HCSS User's Reference Manual

Herschel Data Processing

version 0.2.dev , Document Number: HERSCHEL-HSC-DOC-0935
20 August 2009

HCSS User's Reference Manual: Herschel Data Processing

Table of Contents

Categorized view of Commands

This chapter provides a categorized view of all built-in DPfunctions, tasks and objects.

- **Datasets**
  - NumberedDataset
  - Spectrum1d
  - Spectrum2d
- **Display**
  - ImageAxis
- **Image**
  - AnnotationToolbox
  - Display
  - Flag
  - SimpleCube
  - SimpleImage
  - SimpleStack
  - SlicedCube
  - SlicedImage
  - Wcs
- **PAL**
  - DbFactory
  - DbPool
- **binstruct**
  - PacketSequence
- **class**
  - FitsArchive
  - HduHeaders
  - SerialArchive
- **developer**
  - clear
- **generic task**
  - FitterTask
  - GaussFitTask
- **numeric**
  - REBIN
- **numerics**

- **interp**
  - Bilinear
- **task**
  - bg
  - clear
  - displaylog
  - displayQCLog
  - exportPalToUfDir
  - help
  - importUfDirToPal
  - localStoreWriter
  - openFile
  - openVariable
  - pause
  - restore
  - resume
  - save
  - simpleFitsReader
  - simpleFitsWriter
  - SourceExtractorTask
  - **cube**
    - importCubeTask
    - importCubeTask
  - **general**
    - info
  - **image**
    - clamp
    - crop
    - cutLevels
    - ImageSaverTask
    - importImage
    - rotate
    - scale
    - tiledImage
    - translate
    - transpose

toolbox

- RadialVelocity

utility task

- asciiTableReader
- asciiTableWriter
- pointHistoryDisplay
- poolDataReader
- poolDataWriter

The Herschel Common Science System and Data Processing (DP)

Warning

A number of the examples provided in this document are not rendered correctly when viewed from the JIDE help facility. Please use either the PDF or HTML versions of this document to view the examples correctly.

II.1. Brief Overview

The Herschel Common Science System (HCSS) is being developed by the Herschel Science Center (HSC) and Herschel Instrument Control Centers (ICCs) to provide the complete software system for the Herschel Observatory mission. The intention is to provide a common system that is able to handle test data, observation planning, mission planning and instrument data from observations within one common development. An important element of this common development is Data Processing (DP).

DP handles computed, stored or simulated data and has access to much of the software developed for other purposes within the HCSS (e.g., Quick Look Analysis, which runs on real-time data or replayed data streams from a database).

Branches of the HCSS have also been developed for handling Herschel instrument-specific tasks. So software packages for HIFI, PACS and SPIRE also reside within the HCSS framework and are available within DP.

Since the Herschel DP uses Java programming, it is very flexible and Java programs can be imported into a session. However, the basic DP system is a fully-fledged standalone system that is being developed to specifically deal with data from the Herschel spacecraft.

II.2. Availability of DP and Operating Systems

DP is available free of charge as part of the HCSS and can be downloaded for use on networked or individual desktop/laptop machines. Current operating systems supported by DP include

- Solaris 2.8+
- LINUX (Red Hat 8.0+, SuSE 9.1)
- Windows (2000, XP)

For download and installation instructions see ????.

II.3. Related Documentation

In earlier versions of the DP User's Manual, 'HowTo' documents were available in parallel. Earlier HowTo documents for users are now incorporated into the current User's Manual. Developers of DP packages have also produced Javadocs (at ftp://ftp.rssd.esa.int/pub/HERSCHEL/csdt/releases/doc/api/index.html) which currently provide some basic information on some of the underlying libraries and programs that comprise the DP system.

Note

Users should be aware that these are NOT fully fledged help documents and are probably most useful to system developers or advanced users only.

II.4. Versioning

DP is still very much a system under development and it is intended that this manual will be updated with the regular user release updates of the system. The first version of this manual is associated with User Release v0.3 of the HCSS. Version numbering of the manual will be matched to that of the user release (it is hoped). So the first manual is version 0.3.

Chapter 1. Numeric

1.1. Introduction

This guide is written with Jython users in mind and has a strong focus on Jython syntax. Nevertheless the information found here may be quite useful for Java developers as well.

Quick Start	This may help you starting up.
Import statements	Describes various ways to get access to the functionality in this library.
Arrays	All you need to know about arrays that are defined within this library.
Toolboxes	All the functions and procedures within this library are grouped in toolboxes.

1.2. Quick Start

The following code serves as a quick introduction that may help using the library.

```
# Get everything from the library:
from herschel.ia.numeric.all import *

# Define an 2d double array, and populate it:
x=Double2d(2,3)
x[0,:]=[1,2,3]
x[1,:]=[4,5,6]
print x

# Compute the result of a simple addition:
y=x+1
print y

# Similarly, but do this by altering x itself:
x+=1
print x

# Using a function:
y=SQUARE(x)
print y
```

1.3. Import statements

This section is not applicable where users access the library components from within a JIDE session or a JConsole session as in these cases all the components are automatically available.

A jython session does not give automatic access to numeric arrays and/or functions respectively. For that you have to import them into the name-space of your jython session.

Alternatively you can get finer control over what you import into your session. Because of the layout of the library, you can choose to import arrays definitions only, or even a single array definition.

For those users that are familiar with Java, Jython provides a way to avoid name-space pollution by importing a single name into your session:

```
import herschel
x=herschel.ia.numeric.Double1d([0.,1/3.,0.5,1])
f=herschel.ia.numeric.toolbox.basic.Basic.SIN
y=f(x)
```

In fact, the import statement above gives you access to the complete herschel code without potential name clashes. Alternatively you could do something like:

```
import herschel.ia.numeric
x=numeric.Double1d([0.,1/3.,0.5,1])
f=numeric.toolbox.basic.Basic.SIN
y=f(x)
```

Importing everything

As a user you may want to import all the arrays and functions of the numeric library in one go. This may be rather convenient as it avoids importing all the bits and pieces yourself.

The easiest way to get access to the library is to import every array and function definition into your name-space using the `all` module:

```
from herschel.ia.numeric.all import *
```

1.4. Arrays

This chapter describes how to create and access arrays.

Definitions	Description of all supported array types and dimensions.
Creation	Creation and initialization of arrays.
Properties	General information about your array.
Conversions	Implicit and explicit conversions..
Operators	Unary and binary operators that can be applied to arrays.
Methods	Additional mechanisms to access or manipulate your array, such as `where`, performing functions in-line.

Definitions

The library defines the following arrays:

	rank of the array				
Type	**1**	**2**	**3**	**4**	**5**
String	String1d	-	-		
Boolean	Bool1d	Bool2d	Bool3d	Bool4d	Bool5d
Byte (8-bit)	Byte1d	Byte2d	Byte3d	Byte4d	Byte5d
Short (16-bit)	Short1d	Short2d	Short3d	Short4d	Short5d
Integer (32-bit)	Int1d	Int2d	Int3d	Int4d	Int5d
Long (64-bit)	Long1d	Long2d	Long3d	Long4d	Long5d
Float (32-bit)	Float1d	Float2d	Float3d	Float4d	Float5d
Double (64-bit)	Double1d	Double2d	Double3d	Double4d	Double5d
Complex (128-bit)	Complex1d	Complex2d	Complex3d	Complex4d	Complex5d

Creation

There are various ways to create an array:

Creating a numeric array of a particular shape

You can create an array by specifying the size of each dimension and optionally a scalar value that is used to initialize all the elements. In general the syntax looks like:

```
x=ArrayPrefix1d(length[,value])
x=ArrayPrefix2d(m,n[,value])
x=ArrayPrefix3d(m,n,p[,value])
```

, where `ArrayPrefix1d` is something like `Int1d`.

Examples:

```
x=Bool1d(1)               # [false]
x=Int1d(3)                # [0,0,0]
x=Int1d(3,4)              # [4,4,4]
x=Double2d(3,4)           # [ [0.0,0.0,0.0], [0.0,0.0,0.0], [0.0,0.0,0.0] ]
x=Complex3d(2,3,4,4-3j)
```

Warning

Caveat for users that come from IDL and FORTRAN: The Jython scripting language (and Java for that matter) is using the *last* dimension as the fastest running index! Therefore, the IDL construct `DBLARR(2,3,4)` is equivalent to the Jython construct `Double3d(4,3,2)`.

Creating a 1d array as a range

Using the following syntax:

```
x=ArrayPrefix1d.range(n)
```

, you can create a one-dimensional array of `n` elements, containing the values `[0,1,...,n-1]`.

Examples:

```
x=Int1d.range(3)          # [0,1,2]
x=Double1d.range(4)       # [0.0,1.0,2.0,3.0]
x=Complex1d.range(2)      # [0.0+0.0j,1.0+0.0j]
```

Note, this does not work for `Bool1d` and `String1d`

Creating a numeric array from a jython array

You can create a numeric array from a jython array. Doing so, the library checks whether the jython array was jagged or not.

```
jx=[1,2,3]                # a jython array
x =Int1d(jx)              # a numeric array of rank 1
x =Int1d([1,2,3])         # idem, but in one go

jx=[ [1,2,3], [4,5,6] ]   # a jython array of arrays
x =Double2d(jx)           # a numeric array of rank 2
x =Double2d([ [1,2,3], [4,5,6] ])
```

Making a copy of a numeric array

The assignment `y=x` will usually mean that the variable names `x` and `y`*refer* to the same values.

For example:

```
x=Int1d.range(3)
y=x
print y    # [0,1,2]

x[0]=-1
print y    # [-1,1,2], as x and y refer to the same contents!!
```

If you would like to create a copy, you have to do that explicitly:

```
x=Int1d.range(3)
y=x.copy()
print y      # [0,1,2]

x[0]=-1
print y      # [0,1,2]
```

Note that this behavior is not specific to this library, but it is part of the jython language!

You may want to consult the following link for more insights about sharing data in numeric arrays.

Converting to another type

You can convert a numeric array to another numeric array of the same shape but of different type:

```
x=Int1d([3,4,5])
y=Double1d(x)
z=Complex1d(x)
x=Int1d(y)
```

Properties

Properties (sometimes called attributes or fields, depending on what computer languages you are familiar with) are named items that tell something about your variable.

An array has several *read-only* properties that can be accessed using the following syntax:

```
variable.property
```

This chapter explains the common properties of an array, using the following example variable:

```
x=Int3d(3,5,4)
```

Class

You can ask the array for its Definitions:

```
print x.__class__# herschel.ia.numeric.Int3d
```

Note the underscores. Actually, this property is not specific to arrays. In fact it can be used on any variable in your jython session, as this property is defined as part of the Jython language!

Size

The size gives you the number of all elements within the array

```
print x.size       # 60
```

Rank

The rank is the number of dimensions of an array

```
print x.rank        # 3
```

Dimensions

The dimensions property tells you something about the number of elements along each dimension.

```
print x.dimensions      # [3,5,4]
print x.dimensions[0]   # 3
```

Conversions

Often you are dealing with an array of a specific type, but your function expects another type. In those cases the array needs to be converted.

Below we discuss the conversion behaviors:

Implicit Conversion

An implicit conversion is something the *system* is doing for you. A function may need a Double1d as input, but it can also handle an Int1d. It just silently converts it to what is needed.

Examples:

```
x=Double1d.range(3) # [0.0,1.0,2.0]
x[0]=1              # implicitly convert '1' to '1.0'
y=Byte1d.range(3)   # [0,1,2]
x+=y                # implicitly convert y to the type of x
```

Explicit Conversion

An explicit conversion is something *you* force upon the system. This may be because a particular function is expecting an Float1d array, but your data is stored in an Int1d array.

With an explicit conversions you can also truncate a floating point array into an integer array; this is called narrowing. The opposite is called widening.

Examples:

```
f=Float1d.range(3)+0.5  # [0.5,1.5,2.5]
d=Double1d(f)           # widening!
i=Int1d(d)              # narrowing! [0,1,2]
z=Complex1d(d)          # [0.5+0j,1.5+0j,2.5+0j]
```

Operators

The numeric library provides an extensive set of operators that allow you to manipulate the arrays in various ways.

Access

You can access elements in an array by using the square brackets (`[access-part]`) notation. The *access-part* is a comma-separated list of access items, where each item can be an index or a range/slice:

index — The indices run from `0` to `n-1`, where n is the length of a particular dimension.

selection — An arbitrary number of indices for a particular dimension, e.g.:

```
Selection([1,3,7,200])
```

slice — A slice is specified as: `begin:end[:step]`, where:

- begin — starting index. If not specified, it assumes the first element; if negative, it counts backwards from the **length** of the dimension in question.
- end — ending index. The specified index is *excluded*. If not specified, it takes the length of the dimension in question; if negative, it counts backwards from the **length** of the dimension in question.
- step — step size. Must be a positive number. If not specified, it assumes a default step size of one.

Note, that [:],[::],[:end:],[start:], [::step] are all valid statements, as each field is optional.

To illustrate the above we will use the following variables:

```
vector # array data of rank 1
array  # array data of rank 2
cube   # array data of rank 3
```

Rank 1 Arrays

Syntax:

```
# elemental access
vector[i]=value
value=vector[i]

# slices
vector[i:j]=value     # value is {scalar|data1d|jython-1d-sequence}
value=vector[i:j]     # returns a data1d of the same type as vector
```

Example:

```
x=Int1d([1,2,3,4,5,6])
#        | | | | | |
# index: 0 1 2 3 4 5
x[1]                              # 2 as indexing always starts at 0
x[:]                              # [1,2,3,4,5,6]
x[1:]                             # [2,3,4,5,6]
x[:-1]                            # [1,2,3,4,5]
x[2:5]                            # [3,4,5]
x[::2]                            # [2,4,6]
```

Rank 2 Arrays

Syntax:

```
# elemental access
array[i,j]=value          # value is a scalar
value=array[i,j]

# slices
array[i0:i1,j0:j1]=value  # value is {scalar|data2d|jython-2d-sequence}
value=array[i0:i1,j0:j1]  # returns a data2d of the same type as array

# section
array[i,j0:j1]=value      # value is {scalar|data1d|jython-1d-sequence}
array[i0:i1,j]=value
value=array[i0:i1,j]      # returns a data1d of the same type as array
value=array[i,j0:j1]
```

Example:

```
x=Int2d([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
```

```
x[1,1]              # 6
x[1,:]              # [5,6,7,8]
x[:,1]              # [2,6,10]
x[:,1:-1]           # [ [2,3],[6,7],[10,11] ]
```

Rank 3 Arrays

Syntax:

```
# elemental access
cube[i,j,k]=value             # value is a scalar
value=cube[i,j,k]

# slices
cube[i0:i1,j0:j1,k0:k1]=value # value is {scalar|data3d|jython-3d-sequence}
value=cube[i0:i1,j0:j1,k0:k1] # returns a data3d of the same type as cube

# section
cube[i,j0:j1,k0:k1]=value     # value is {scalar|data2d|jython-2d-sequence}
cube[i0:i1,j,k0:k1]=value
cube[i0:i1,j0:j1,k]=value
cube[i,j0:j1,k0:k1]=value     # returns a data2d of the same type as cube
cube[i0:i1,j,k0:k1]=value
cube[i0:i1,j0:j1,k]=value

cube[i,j,k0:k1]=value         # value is {scalar|data1d|jython-1d-sequence}
cube[i,j0:j1,k]=value
cube[i0:i1,j,k]=value
value=cube[i,j,k0:k1]         # returns a data1d of the same type as cube
value=cube[i,j0:j1,k]
value=cube[i0:i1,j,k]
```

Arithmetic Operators

You can use the following list of operators on arrays that are of numeric type (Byte arrays ... Complex arrays):

OP	Operation	Example	Before	After
*	multiplication	x*2	4	8
/	division	x/2	4	2
+	addition	x+2	4	6
-	subtraction	x-2	4	2
**	exponentiation	x**2	4	16
%	modulus	x%2	5	1

, where **OP** denotes the operator symbol. All arithmetic operators can be used with the following syntax:

```
# classic
arrayNd = arrayNd OP scalar
arrayNd = arrayNd OP arrayNd

# in-line
arrayNd OP= scalar
arrayNd OP= arrayNd
```

, where `arrayNd` is the name of an array variable of `N` dimensions. The first two lines create a new array. The new array has the same type as the widest type of the two variables in the right-hand-side expression:

```
a=Int1d([1,2,3])
b=Float1d([4,5,6])
c=a*1.         # c is a Double1d
```

```
c=a+b        # c is a Float1d
```

The form OP=, we sometimes refer to as an *in-line operator*, as they do not produce a new result, but apply the operation on the left-hand-side. Though this form is functionally equivalent to its counterpart (z=x OP y), there are some subtleties.

The in-line form of the operator does not create an array, but tries to change the left-hand-side. This saves memory and increases speed of the operation.

Note, that the type of the right-hand-side can not be wider than the left-hand-side:

```
x=Int1d([1,2,3])
y=Float1d([4,5,6])
x+=1.       # error: the right-hand-side is a floating point scalar!
x+=y     # error: the right-hand-side is a floating point array!
x*=2     # OK: [2,4,6]
y+=x     # OK: [6.0,9.0,12.0]
```

Logical Operators

Logical operators combine logical expressions, and they are defined for all boolean arrays.

If one defines two variables:

```
x=Bool1d([1,0,1,0])
y=Bool1d([1,0,0,1])
```

then the following logical operators are applicable:

OP	Operator	Example	Result
&	and	x & y	[true,false,false,false]
|	inclusive or	x | y	[true,false,true,true]
^	exclusive or	x ^ y	[false,false,true,true]

Relational Operators

Relational operators compare two arrays on an element-by-element basis, and they are defined for all numeric ordered arrays (thus not for complex arrays).

If one defines two variables:

```
x=Int1d([1,2,3])
y=Double1d([3,2,1])
```

then the following relational operators are applicable:

OP	Operator	Example	Result
<	less than	x < y	[true,true,false]
<=	less than or equals	x <= y	[true,true,false]
==	equals	x == y	[false,true,false]
!=	not equals	x != y	[true,false,true]
>=	greater than or equals	x >= y	[false,true,true]
>	greater than	x > y	[false,false,true]

Methods

This section is under construction.

This section describes various special methods of all defined arrays.

Apply and Perform Methods

In general a Jython User will use the following Jython syntax to apply functions to an array:

```
x=Double1d(...)
y=SIN(SQUARE(x))
```

Method: apply()

Using the Java syntax in Jython provides exactly the same results:

```
x=Double1d(...)
y=x.apply(SQUARE).apply(SIN)
```

Method: perform()

Note that in the examples above two array copies are made. An alternative that mutates the array itself:

```
x=Double1d(...)
x.perform(SQUARE).perform(SIN)
```

Combining

If you want to keep the original data intact, a combination of apply and perform might do the trick:

```
x=Double1d(...)
y=x.apply(SQUARE).perform(SIN)
```

When to use what

In the last example, the SQUARE function is applied to x in order to create a new array; then the SIN function is performed onto the newly created array.

The combination of apply and perform methods allow you to avoid unnecessary creation and initialization of temporary arrays. That is, it allows for optimization of your code

In general, we recommend to prototype your code with the normal syntax. Once satisfied with your algorithm, you may want to tune it using the methods described above.

Where

The `where` method allows you to to query and filter your array data. The mechanisms and how to use it are described in this chapter.

Functionality

`Where` methods return a Selection object that contains an indexing mechanism into an array. The basic syntax is:

```
q=x.where( predicate )
y=x[q]                  # getting selected elements
x[q]=v                  # setting selected elements
```

where:

- **x** is the array this function is operating on,
- **predicate** - the applied logical function,
- **q** is a Selection object containing the subscripts of those elements matching specified predicate;
- **y** - a one dimensional array holding only those elements in `x` that are defined at subsscripts defined in `q`;
- **v** - is a value or an array of values of the same type as `x`. These values are assigned to those elements in `x` that are defined at subsscripts defined in `q`.

Selection object

All `where` methods return a selection. A selection object is a container of *long index* subscripts, where a multi-dimensional array is treated as if it would be a one-dimensional array.

Taking a two-dimensional array as an example, the mapping is as follows:

Table 1.1. Mapping of long indices to normal indices in a m x n array.

	j=0	**1**	**...**	**n-1**
i=0	0	1	...	n-1
1	1*n	1*n+1	...	1*n+n-1
:	:	:	:	:
m-1	(m-1)*n	(m-1)*n+1	...	(m-1)*n+n-1

For example the subscripts `(i=2,j=2)` in a `(m x n)=(3x4)` array correspond to the `long index=2*4+2=10`.

Basic Usage

The `where` function works on arrays of any rank, except for an input argument that is a Jython *lambda* expression: this kind of functionality is limited to arrays of rank one.

Below follows the various ways of using `where` functionality:

Using a mask as your predicate

As described above the predicate specifies the condition(s) that are evaluated for the array the `where` method is operating on.

One way of using this method is by providing a mask as a predicate. You can create this mask manually or as a result of a logical expression. The mask must have the same shape as the array it is operating on:

```
x=Double1d(  [1,2,3,4,5,6,7,8,9,10] )

# Applying a manually created mask
q=x.where(Bool1d( [1,0,1,0,1,0,1,0,1,0] ))
print q              # [0,2,4,6,8]
print x[q]           # [1,3,5,7,9]

# Applying a mask resulting from a logical expression:
q=x.where( (x>2).and(x<8) )            # alternative syntax: (x>2)&(<8)
print q                                # [2,3,4,5,6]
```

```
print x[q]                              # [3,4,5,6,7]

# Same, but now on a 2x5 array:
x=Double2d( [ [1,2,3,4,5],[6,7,8,9,10] ] )
q=x.where( (x>2).and(x<8) )      # alternative syntax: (x>2)&(<8)
print q                                 # [2,3,4,5,6]
print x[q]                              # [3,4,5,6,7]

# Assignment
x[q]=44
print x
# [[1,2,44,44,44],[44,44,8,9,10]]
x[q]=Double1d([33,22,11,22,33])
print x                                 # [[1,2,33,22,11],[22,33,8,9,10]]
```

Using Predicate Functions

The `where` method also works in conjunction with array predicate objects, such as the function IS_FINITE (is finite number).

```
x=Double1d(  [1,2,Double.NaN,3,4] )
q=x.where(IS_FINITE)
print x[q]                              # [1,2,3,4]
```

Lambda Predicate Expressions

Lambda expressions are Jython constructs to apply a certain logic to a list of elements:

```
x=Double1d(  [1,2,3,4,5,6,7,8,9,10] )
q=x.where(lambda i: i>2 and i<8)
print x[q]                              # [3,4,5,6,7]
```

When to use what

Predicate function and boolean arrays have a better performance than lambda expressions.

Advanced Usage

A selection created on one object can be applied to another object as well. This is demonstrated in the sections below.

Applying a selection along a dimension

Suppose a set of numeric arrays are somehow related, for example -using the objects defined below- the value a1d[i] is related to a2d[i,:] and a3d[:,i,:].

Then we create a selection by e.g. applying the `where` function on a1d, and apply that selection to a2d and a3d as well as follows:

```
# --- array definition ---
# 4 elements
a1d=Double1d([1,5,Double.NaN,2])
# 4x3 elements
a2d=Int2d([ [1,2,3],[4,5,6],[7,8,9],[10,11,12] ])

# 2x4x2 elements
a3d=Byte3d([ [[1,2],[2,3],[3,4],[4,5]], [[5,4],[4,3],[3,2],[2,1]] ])

# --- creating a selection manually ---
mask=Bool1d([1,0,0,1])
q=a1d.where(mask)
print "Selected indices:",q # [0,3]

# --- filtering your data ---
print a1d[q]                # [1.0,2.0]
print a2d[q,:]              # [ [1,2,3],[10,11,12] ]
print a3d[:,q,:]            # [ [[1,2],[4,5]], [[5,4],[2,1]] ]
```

```
# --- creating a selection using predicate function ---
q=a1d.where(IS_FINITE)
print "Selected indices:",q # [0,1,3]

# --- filtering your data ---
print a1d[q]                # [1.0,5.0,2.0]
print a2d[q,:]              # [ [1,2,3],[4,5,6],[10,11,12] ]
print a3d[:,q,:]            # [ [[1,2],[2,3],[4,5]], [[5,4],[4,3],[2,1]] ]
```

Applying a selection on an array of different shape

The example above showed how to apply a selection along a particular dimension. This section is making use of the *long indexing* features of arrays as described in section Jython Usage->Arrays->Operators->Access.

In short the idea is that a Selection *can* be considered as a set of long indices. For example the indices `[i,j]=[1,2]` in a 4x3 array corresponds to the long index `li=1*3+2=5`. Thus:

```
# --- definitions ---
a1d=Int1d( [4,1,3,9,2,7,4,5] )
a2d=Double2d([ [1,2,3,4], [4,3,2,1] ])

# --- query ---
q=a1d.where( (a1d > 2).and(a1d < 8).and(a1d != 4) )
print q
# [2,5,7]

# --- query used as long index ---
print a1d[q] # [3,7,5]
print a2d[q] # [3.0,3.0,1.0]
```

Manipulating and creating Selections

The `where` method creates a Selection for you as specified by the predicate argument of this method. It is possible however to create and/or manipulate the indices yourself by converting a selection to and from an Int1d array as follows:

```
# create a selection from an integer array
q=Selection([1,2,6]) # holding indices 1,2 and 6

# extracting the indices from a selection as an Int1d array
x=q.toInt1d()
x[2]=7

# recreate the modified selectionfrom a Int1d array
q=Selection(x)
```

One possible usage is these conversions may be a function that manipulates indices, and the best way to manipulate them is as if they were elements in an integer array (including all the access possibilities and applicable functions). For example if one would like to change the order of of the selection:

```
x=Int1d([1,2,3,4,5])
q=x.where(x > 2) # indices: [2,3,4]
print x[q]          # values:  [3,4,5]

q=Selection(REVERSE(q.toInt1d()))
print x[q]          # values: [5,4,3]
```

Operator Methods

The numeric library provides syntax constructs that allow for the section called “Operators” of writing binary and unary operators, which is not possible in Java. As the library is written for Jython and Java users, these operators are available in the form of methods as well.

There are subtle differences:

- All operators that could be written as inline operators are implemented as methods that mutate the array it operates on.
- All operator methods are explicit, that is no conversions apply. For example adding an array to say an Int1d array only works if the array that is added is an Int1d array as well.

Operator Methods

Arithmetic Operators

All inline operators such as "+=", "-=", "*=" have equivalent methods named "add", "subtract", "divide", etc...

Relational Operators

All operators such as "==", "!=", "&lt", etc... have equivalent methods named "eq", "ne", "lt", etc...

Logical Operators

All logical operators such as "&", "!", "|" and "^" have equivalent methods named "and", "not", "or" and "xor" respectively. The latter methods mutate the logical array it is operating on:

```
a.and(b).or(c)          # contents of 'a' is altered!
r=a.copy().and(b).or(c)          # contents of 'a' is altered!
```

When to use what

Operator methods that work inline are as fast as their abbreviated syntax (e.g. "+="). The real benefit comes from a style of writing:

```
y=SQRT( SQUARE(SIN(x)/n) + 1 )
y=x.apply(SIN).divide(n).perform(SQUARE).add(1).perform(SQRT)
```

The code snippet above has reduced the creation of temporary arrays from four down to zero!

1.5. Toolboxes

Introduction

Toolboxes are add-ons that extend the functionality of numeric arrays. All the functions and procedures within this library are grouped in toolboxes.

Each toolbox has it own set of specialized functions, documentation and example code. If you click on one of the toolboxes on the left, you will get more information about the functionality it provides.

Currently available toolboxes are:

The Basic Toolbox	Basic functions like SIN, SQUARE, MIN, ...
The Fit Toolbox	Provides functionality for fitting.
The Filter Toolbox	Provides filter functions and utilities useful for processing time-series and images.
The Integration Toolbox	Provides numeric integration methods, for both continuous functions and discrete data.
The Interpolator Toolbox	Provides functions and utilities for interpolation and extrapolation.

The Mask Handling Toolbox	Provides facilities for handling masks that are stored in numeric library arrays.
The Matrix Toolbox	Set of functions and utilities to manipulate matrices.
The Random Functions Toolbox	Set of functions for populating arrays with random elements.
The XFORM Toolbox	Provides Transformational functions such as FFT.

The Basic Toolbox

This toolbox provides basic functions in the following categories:

- General Functions
- Trigonometric Functions
- Nearest Integer Functions
- Reduction Functions
- Statistical Functions

```
name-space: herschel.ia.numeric.toolbox.basic
```

General Functions

- IS_FINITE
- MEDIANDEV
- EXP10
- LOG
- IS_INFINITE
- ABS
- AnyPresent
- SORT
- NotPresent
- SQUARE
- IS_NAN
- LogN
- SQRT
- Pow
- SIGCLIP
- UNIQ
- RESHAPE
- Polynomial

- LOG10
- UNIQ_SORTED
- EXP
- CONCATENATE
- Rotate
- SHIFT
- ExpN
- REVERSE

Trigonometric Functions

- COS
- TANH
- ARCCOS
- SIN
- TAN
- ARCTAN
- ARCSIN
- SINH
- COSH

Nearest Integer Functions

- FLOOR
- ROUND
- CEIL

Reduction Functions

This family of functions reduces a sequence of values a single value by combining elements via a supplied function. Additionally, this mechanism can be used to reduce an array of rank N to an N-1 array.

The general syntax of reduction functions is as follows:

```
y=f(x [,dim])
```

where: *x* is the input array of any rank, *dim* an optional dimension and *y* is the result of the computation. If the optional dimension is not specified, the array is reduced to a single scalar value.

If the dimension is specified the function computes the result by reducing the array elements along that dimension into a single value; the resulting array has a rank which is equivalent to x.rank-1.

For example, consider an arbitrary reduction function *f* and the following two-dimensional array

```
  | 1 2 |
```

```
x=| 3 4 |
```

Then:

```
f(x)   = f([1,2,3,4])
f(x,0) = [ f(1,3), f(2,4) ]
f(x,1) = [ f(1,2), f(3,4) ]
```

- ANY
- MIN
- AllPresent
- SUM
- PRODUCT
- ALL
- MAX

Statistical Functions

The following statistical functions accept integral as well floating-point arrays; the former implicitly transformed to a double array.

You have to filter out non-finite elements (such as NaN elements) as these functions produce sensible results only if all elements have finite values:

```
x=Double1d([Double.NaN,1,Double.NaN,3,2,3,4,Double.NaN,4])
print MEDIAN(x)                       # ERROR: the median is not NaN
print MEDIAN(x[x.where(IS_FINITE)])   # 3.0
```

- KURTOSIS
- MEAN
- Erf
- Correlate
- STDDEV
- GammaQ
- Erfc
- GAMMALN
- CorrelateMatrix
- GammaP
- SKEWNESS
- MEDIAN
- VARIANCE

The Filter Toolbox

This toolbox provides filter functions.

```
name-space: herschel.ia.numeric.toolbox.filter
```

- BoxCarFilter
- GaussianFilter
- Convolution

The Fit Toolbox

The fitter toolbox provides a generalized way to fit models to data and to estimate the best values of the model parameters.

Before proceeding it is wise to familiarize yourself with the background information about the Fitter classes.

```
name-space: herschel.ia.numeric.toolbox.fit
```

The Integration Toolbox

The integration toolbox provides a generalized way to compute integrals, through different numeric methods.

```
name-space: herschel.ia.numeric.toolbox.integr
```

- Integrator
- FitterFunction

The Interpolator Toolbox

This toolbox provides interpolation functions.

```
name-space: herschel.ia.numeric.toolbox.interp
```

- LinearInterpolator
- CubicSplineInterpolator
- NearestNeighborInterpolator

The Mask Handling Toolbox

This toolbox provides facilities for mask handling.

```
name-space: herschel.ia.numeric.toolbox.mask
```

- FixedMask
- PackedMask

The Matrix Toolbox

This toolbox provides matrix functions.

```
name-space: herschel.ia.numeric.toolbox.matrix
```

- SOLVE
- INVERSE
- MatrixSolve
- MatrixMultiply
- LUDecomposition
- TRANSPOSE
- DETERMINANT
- MATMUL
- EigenvalueDecomposition

The Random Functions Toolbox

This toolbox provides random functions.

```
name-space: herschel.ia.numeric.toolbox.random
```

- RandomPoisson
- RandomUniform
- RandomGauss

The XFORM Toolbox

This toolbox provides transformation functions and utilities.

```
name-space: herschel.ia.numeric.toolbox.xform
```

- IFFT
- FFT
- HANNING
- HAMMING
- PowerSpectrum

Chapter 2. Product Access Layer

2.1. Introduction

The Product Access Layer consists of several elements that allow you to save, retrieve and query Products that are stored in different locations whether that is on your local file system or remotely.

The **ProductStorage** class provides the user front-end interface that allows you to communicate with **Products** stored in multiple locations.Note that this is where the Product Access Layer fundamentally differs from the (versant) ObjectStore, that only allows to connect to one database in your session.

The locations mentioned above are what we call **Product Pools**; all Product pools adhere to the **ProductPool** interface. Example implementations of the ProductPool interface are local storage (on your laptop), and remote pools, such as the future Herschel Archive, accessing Products within a Versant database, a local storage made available by a fellow astronomer etcetera.

Simply by *registering* a pool to your storage, you can access the Products in that pool.

The ProductStorage provides mechanisms to load, save and query Products in the registered pools. Doing so you receive a reference to a Product (returned by the load() and save() commands) or a set of Product references (when querying). This functionality of a Product reference is provided by the **ProductRef** class; it allows to fetch information of the Product -such as meta data- without loading the Product in question in your memory completely.

The ProductStorage currently distincts three types of queries: 1) query on attributes that are available to all Products, 2) queries on meta data of the Product and 3) full data mining queries!

2.2. Example Usage

Creation and Registry

You can create a storage as follows:

```
storage=ProductStorage()
```

Then you have to assign the Product pools that you want to access. You have to register at least one pool:

```
storage.register(SimplePool.getInstance()) # simple storage on local file system
storage.register(SerialPoolClient("abc.xyz.org",123,"dummy"))
:
storage.register(poolN)
```

Save and Restoring Products

Saving a Product:

```
# creation of a dummy product
product=Product(creator="Me")
product["array"]=ArrayDataset(data=Int1d.range(5))

# save! returns reference to the saved product
reference=storage.save(product)
print reference.urn
# urn:simple.default:herschel.ia.dataset.Product:0
```

Loading a Product:

```
reference=storage.load("urn:simple.default:herschel.ia.dataset.Product:0")
```

A reference provides access parts of the product as well as access to the product itself:

```
print reference.urn
# urn:simple.default:herschel.ia.dataset.Product:0

print reference.type
# herschel.ia.dataset.Product

meta=reference.meta
print meta["creator"]
# Me

product=reference.product
print product.creator
# Me
```

Querying

To be written

2.3. Querying

The Product Access Layer allows to select product references from a ProductStorage by specifying a query.

Query types

We distinct three types of queries:

- AttribQuery

  Allows searching on meta data values -so called attributes- that appear in any product. The attribute names are:

  - creationDate : Date
  - creator : String
  - endDate : Date
  - instrument : String
  - modelName : String
  - startDate : Date
  - type : String

- MetaQuery

  Allows searching on all meta data values and descriptions

- FullQuery

  Allows searching on all aspects of Products exploiting the full interface of the Product family in question

Creation of a query

All queries are setup and executed on a ProductStorage as follows:

```
query=.....Query(product-class,variable,where-expression)
results=storage.select(query)
```

,where:

- *product-class* is the product family. All products are derived from the Product class, but products may be categorized in sub-families such as PacsProduct.
- *variable* is a string holding the name of the product variable that is used in the *where-expression*.
- *where-expression* is a string that specifies the actual expression for which products of the *product-class* should be matched against. Note: the syntax is identical to the syntax of the JIDE command-line.

An example:

```
query=AttribQuery(SomeProduct,'prod',
                  'prod.creator=="Me" and prod.modelName=="Fake"')
```

Query strategy

Typically an AttribQuery is faster than a MetaQuery which is in turn faster than a FullQuery. Depending on the product pools that are registered, a query can take some time; to avoid unnecessary waiting time one can adopt a strategy of staged queries.

For example, a query on attributes is executed first. If too many hits are found, you can refine your query by executing another query *using the hits returned from the previous query*. This process can be repeated until the number of hits have been reduced to a digestable amount:

```
results=storage.select(AttribQuery(...))       # 1000 hits
results=storage.select(MetaQuery(...),results) # 100 hits
results=storage.select(MetaQuery(...),results) # 50 hits
results=storage.select(FullQuery(...),results) # 3 hits
```

Querying for metadata in products

One thing you need to watch out when performing a meta or full query, is when you try to query for a metadata that does not exist in one or more products that you are applying the query to.

For example, consider the following MetaQuery:

```
    query =MetaQuery(Product, 'p', 'p.meta["temperature"].value==10)
    resultset=storage.select(query)
```

The query first starts creating a shortlist of all products in the storage matching type 'Product'. It then runs the query string on each product in that shortlist. If any of those products don't contain the information referenced in the query string, an error is raised.

There are two ways to avoid this:

- Be as specific as you can when it comes to specifying the product type in a query. If you know the product type you want to query is of type 'CalHrsQDCFull', then specify that. Running queries

using the most general product type of 'Product' is not recommended, unless the products you have sav ed are of this type only.

- Run a two-stage query, using the containsKey() operator to check whether a component exists first, eg

  Get a sub-set of products that contain the metadata 'temperature':

```
queryOne= MetaQuery(Product, 'p', 'p.meta.containsKey("temperature")')
resultsetOne = storage.select(queryOne)
```

  Run the original query on this subset:

```
queryTwo =MetaQuery(Product, 'p', 'p.meta["temperature"].value==10)
resultsetTwo = storage.select(queryTwo, resultSetOne)
```

2.4. Product Pools

Before you can do something useful with a created ProductStorage, you have to register one or more pools to that store.

Product pools are implementations that can load, save and query simple Products. All pools share the same features (the so-called ProductPool interface) such that they can be registered to a ProductStorage.

Typically a User sets up one ProductStorage and registers one or more Product pools to it. However the design permits to create multiple ProductStorage devices with a different registry of Product pools. Product pools can also be shared between two ProductStorage devices.

Depending of the Product Pool implementation you have selected, the products in such a pool can be stored on your local file system, talking to a object-oriented database, a relational database etcetera. You can even create your own ProductPool implementation (what about a memory pool, such that you can query all the Products available in memory of your JIDE session?).

Product Pool implementations

The following ProductPool implementations are available in the *Product Access Layer*:

SimplePool	This simple implementation stores Products on your local file system with reasonable query performance up to a few thousand Products.
DbPool	The DbPool stores Products in the Object (Versant) database.
CachedPool	A prototype implementation to cache communications with a ProductPool on your local file system. Useful when you wish to work in a disconnected environment.
SerialClientPool	A prototype implementation to set up a ProductPool server and an accompanying client.

2.5. What is a URN ?

'URN' stands for Uniform Resource Name, and is simply a string identifier to uniquely identify any product stored in any pool.

URNs are automatically generated and assigned to a product when that product is written to a storage. You can find out what the URN is for a product through it's ProductRef. That ProductRef can in turn

be retrieved from the return value of the ProductStorage.save() operation that had initially saved that product:

```
ref = storage.save(myproduct) # Save the product and get the ref
print ref.urn                 # Get the URN
```

2.6. Context Products

Contexts are special types of products that contain references to other products stored.

This enables a means of building complex data structures in a storage.

There are two 'standard' types of context products provided: ListContext (for grouping products into sequences or lists) and MapContexts (for grouping products into containers with access to each by key).

2.7. Deep Copy or Cloning of Products

Say you had a context in one storage that referenced another product, and you wanted to copy that data tree to a different storage. How would you do that?

It is possible to do this using the usual ProductStorage.save() method. If you pass as an argument the context pointing to the 'head' of the data tree you want to clone, the whole data tree is cloned.

So for example, we have create a context with a child and store it in storage a:

```
l=ListContext()
p=Product()
l.refs.add(ProductRef(p))

storageA.save(l)
```

then we want to copy the context and child to a new storage, say storageB, all we do is as follows:

```
storageB.save(l)
```

The above cloning operation has one proviso: if a product within the data tree already exists in the destination product storage, it is not copied. A product can exist in the destination storage if for example, the original and destination storage happen to share a pool, and one of the products in the data tree being copied is in that common pool.

Note that a context may have older versions of it stored in a storage (a older version of a context may be saved when a context is saved, modified, then saved again). The older versions of the context specified in the ProductStorage.save() argument are also cloned (if that context has any decendents that are contexts, the local versions of those descendent contexts are not cloned, however).

2.8. Interface Definitions

Product Storage:

- ProductRef
- MapContext

- ListContext
- ProductStorage

ProductPool implementations:

- SerialClientPool
- CachePool

Query types:

- FullQuery
- MetaQuery
- AttribQuery

Context Products:

Product Browser:

Chapter 3. DP Commands

3.1. Introduction

This chapter is a complete listing of all built-in DP functions, task and objects, collectively referred to as commands.

How to use this chapter

All of Dp's commands are documented alphabetically in this chapter. A description of each command follows its name. Beneath the general description of the command are a number of sections that describe the command in detail, its synopsis, its arguments, its limitations and examples.

Synopsis

The "Synopsis" section shows the proper syntax for calling the command.

Examples

Most commands include one or more examples that demonstrates how the command is used. Most of the examples are just one or two lines of DP code that can be entered at the JIDE prompt. Others are code fragments or routines designed to serve as an examples for your own programs.

Arguments

The "Arguments" section describes each valid argument to the command. Note that most of these arguments are positional parameters that must be supplied in the order indicated by the command's syntax.

However arguments can be often supplied by specifying their names. If its the case then they can provided in any order.

Limitations

The "Limitations" section describes the boundaries of applicability of the command.

Reference

The "Reference" section lists the names of related commands.

History

The "History" section describes the changes occurred to the command.

3.2. ABS

Full Name:	herschel.ia.numeric.toolbox.basic.Abs
Alias:	ABS
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Abs

Description

Gives the absolute value of a number or a numeric array. For complex numbers, ABS(x) gives the modulus.

Example

Example 1: Apply ABS on a Int1d

```
x=Int1d( [-1,0,1] )
print ABS(-123), ABS(x) # 123 [1,0,1]
```

API Summary

Jython Syntax

```
<y>=ABS(<x>)
```

Properties

```
any number and numeric type x [INPUT, MANDATORY, default=no default value]
```

```
any number and numeric type y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any number and numeric type x [INPUT, MANDATORY, default=no default value]

Accepts any number and numeric type.

any number and numeric type y [OUTPUT, MANDATORY, default=no default value]

For complex numbers, ABS(z) gives the modulus.

3.3. AbstractMappingTask

Full Name:	herschel.ia.toolbox.image.AbstractMappingTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import AbstractMappingTask

Description

An abstract Task for mapping.

An abstract Task where the overlap of an input pixel with (smaller) output pixels must be determined.

API Summary

Property
`boolean` **`oversample`** `[INPUT, OPTIONAL, default=Default value : true]`

API details

Property

`boolean oversample [INPUT, OPTIONAL, default=Default value : true]`

Indication whether oversampling should be done.

3.4. AddSpectrum

Full Name:	herschel.ia.toolbox.spectrum.AddSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import AddSpectrum

Description

Task for adding a scalar to the flux data included in a spectrum container or for adding two spectrum containers on a spectrum-by-spectrum basis (computed as an average).

For the scalar mode, 'ds' and 'param' need to specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1,3,4,2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1,3,4]` and for the pair operation `segments1=[1,3,4], segments1=[3,6,5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the orginal datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import AddSpectrum
add = AddSpectrum()
spectraOut = add(ds1=spectra1, ds2=spectra2)
spectraOut = add(ds=spectra, param=2.1)
spectraOut = add(ds=spectra, selection={"bbtype":[6031]}, param=2.1)
spectraOut = add(ds=spectra, selection=[0,1,2,3], param=2.1)
spectraOut = add(ds=spectra, selection={"Chopper":([-4.4,5.9],0.2), "bbtype":
[6031]}, param=2.1)
spectraOut = add(ds=spectra, selection={"LoFrequency":(4000.0,5000.0)},
 param=2.1)
spectraOut = add(ds=spectra, selection={"bbtype":[6031]}, segments=[2,3],
 param=2.1)
```

API Summary

Properties
SpectrumContainer **ds** [INPUT, OPTIONAL, default=no default value.]
Double **param** [INPUT, OPTIONAL, default=no default value.]
Object **selection** [INPUT, OPTIONAL, default=None.]
PyDictionary|Map> **selection_lookup** [INPUT, OPTIONAL, default=no default value.]
PyList **selection_index** [INPUT, OPTIONAL, default=No default value.]
Boolean **overwrite** [INPUT, OPTIONAL, default=False.]
SpectrumContainer **ds1** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **ds2** [INPUT, OPTIONAL, default=no default value.]
Object **segments** [INPUT, OPTIONAL, default=no default value.]
Object **segments1** [INPUT, OPTIONAL, default=no default value.]
Object **segments2** [INPUT, OPTIONAL, default=no default value.]
String **variant** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]

Input container to be processed by the task in case the task is used as a scalar operation.

Double param [INPUT, OPTIONAL, default=no default value.]

The scalar parameter to be considered when the task is used as scalar operation.

Object selection [INPUT, OPTIONAL, default=None.]

Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra for which the add should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also

- ArithmeticSpectrumTask

History

- `2007-08-17 - meli`: Initial.
- `2008-03-16 - meli`: major refactoring of the toolbox.
- `2008-10-03 - meli`: segments, javadoc

3.5. ALL

Full Name:	herschel.ia.numeric.toolbox.basic.All
Alias:	ALL
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import All

Description

Determines whether all elements in the array evaluate to true.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply ALL on a Int1d: check if all the array items, in the specified dimension, are lower than '3'

```
x=Int2d( [ [1,2], [1,4], [1,6] ])
print ALL(x<3)    #all items => 0 (false)
print ALL(x<3, 0) #dim 0 => [ ALL(Int1d([1,1,1]) < 3) , ALL(Int1d([2,4,6]) <
 3) ] = [true, false]
print ALL(x<3, 1) #dim 1 => [ ALL(Int1d([1,2]) < 3) , ALL(Int1d([1,4]) < 3),
 ALL(Int1d([1,6]) < 3) = [true, false, false]
```

API Summary

Jython Syntax
<y>=ALL(<array_expression>[,<dimension>])

Properties
any expression with an integral type **x** [INPUT, MANDATORY, default=no default value]
integer **dimesion** [INPUT, OPTIONAL, default=all the array items]
a boolean array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any expression with an integral type x [INPUT, MANDATORY, default=no default value]

The input array must be of integral type (e.g. bytes,integers) See example.

integer dimesion [INPUT, OPTIONAL, default=all the array items]

The dimension where to apply the expression.

a boolean array y [OUTPUT, MANDATORY, default=no default value]

The result is a boolean array

3.6. AllPresent

Full Name:	herschel.ia.numeric.toolbox.basic.AllPresent
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import AllPresent

Description

Tests whether all of the bits in specified mask are present in the elements of the input array.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply AllPresent on a Int1d: test if a binary 'and' comparison, item by item, with '3' (as mask) is equals to '3'

```
x=Int1d([0,1,2,3])
print AllPresent(3)(x)
#=> [  (0 & 3) == 3,   (1 & 3) == 3,   (2 & 3) == 3,   (3 & 3) == 3  ] =
 [false,false,false,true]
print x.apply(AllPresent(3)) #Another way for using AllPresent
```

API Summary

Jython Syntax
`<y>=AllPresent(<bitmask>)(<x>)`

Properties
`any integral type x [INPUT, MANDATORY, default=no default value]`
`any number bitmask [INPUT, MANDATORY, default=no default value]`
`a boolean array y [OUTPUT, MANDATORY, default=no default value]`

API details

Properties

`any integral type x [INPUT, MANDATORY, default=no default value]`

The input array must be of integral type (e.g. bytes,integers)

`any number bitmask [INPUT, MANDATORY, default=no default value]`

Accepts any number.

`a boolean array y [OUTPUT, MANDATORY, default=no default value]`

The result is a boolean array

See also

- AnyPresent
- NotPresent

3.7. AmoebaFitter

Full Name:	herschel.ia.numeric.toolbox.fit.AmoebaFitter
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.fit import AmoebaFitter

Description

An introduction to the use of fit package can be found

here. At least once have a look at the background on the organization and structure of the package.

Example

Example 1: The fit/demo directory contains worked

```
<a href="../../../ia/numeric/toolbox/fit/demo/jython.html">examples</a>
on almost all aspects of of the package.
```

Limitations

1. AmoebaFitter is **not** guarateed to find the global minimum.

2. The calculation of the evidence is an Gaussian approximation which is only exact for linear models with a fixed scale.

3.8. AnnotationToolbox

Full Name:	herschel.ia.gui.image.gui.AnnotationToolbox
Type:	Java Class -
Import:	from herschel.ia.gui.image.gui import AnnotationToolbox
Category:	Image

Description

A toolbox to draw annotations.

AnnotationToolbox construct a toolbox where the use can select which annotation to display. Using the mouse, the annotation can be put on the image. The jython command to reconstruct the annotation is also shown.

API Summary

Constructors
AnnotationToolbox(Display d) Constructor for the annotation toolbox.
AnnotationToolbox(Display d, boolean useAsComponent) Constructs the annotation toolbox.

Methods
getComponent() Returns the component that contains the annotation toolbox.
close() Closes the annotation toolbox.
saveJythonCode() Saves the jython code to a python script.

API details

Constructors

AnnotationToolbox(Display d)

Constructs the annotation toolbox.

Argument

Display **d** [INPUT, MANDATORY]

AnnotationToolbox(Display d, boolean useAsComponent)

Constructs the annotation toolbox. If useAsComponent is true, the annotation toolbox is component based.

Arguments

Display **d** [INPUT, MANDATORY]

boolean **useAsComponent** [INPUT, MANDATORY]

Methods

getComponent()
Returns the component that contains the annotation toolbox.

close()
Closes the annotation toolbox.

saveJythonCode()
Saves the jython code to reconstruct the annotations to a python script.

3.9. AnnularSkyAperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.AnnularSkyAperturePhotometryExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import AnnularSkyAperturePhotometryExplorer

Description

An explorer for AnnularSkyAperturePhotometryProducts.

API Summary

Constructors
`AnnularSkyAperturePhotometryExplorer()`
The constructor of a new AnnularSkyAperturePhotometryExplorer.
`AnnularSkyAperturePhotometryExplorer(Object object)`
The constructor of a new AnnularSkyAperturePhotometryExplorer

Methods
`boolean` **`canHandle`**`(Class className)`
Checks whether this AnnularSkyAperturePhotometryExplorer can
`AnnularSkyAperturePhotometryProduct` **`getObject`**`()`
Returns the object for this AnnularSkyAperturePhotometyrExplorer.
`Class` **`getVariableType`**`()`
Returns the expected variable type for this AnnularSkyAperturePhotometryExplorer.
`JTable` **`getParameterTable`**`()`
Constructs and returns the parameter table for this
`JPanel` **`getPlotPanel`**`()`
Constructs and returns the plot panel for this
`JPanel` **`getSkyIntensityPlotPanel`**`()`
Constructs and returns the plot panel for this
`JComponent` **`getSkyIntensityPlot`**`()`
Constructs and returns the sky intensity plot for this

API details

Constructors

`AnnularSkyAperturePhotometryExplorer()`

`AnnularSkyAperturePhotometryExplorer(Object object)`

associated with the given object.

Argument

`Object` **`object`** `[INPUT, MANDATORY]`

Methods

boolean canHandle(Class className)

handle objects of the given class.

Argument

`Class` **className** [INPUT, MANDATORY]

Return

boolean

Returns true if this AnnularSkyAperturePhotometry can handle objects of the given class; false otherwise.

AnnularSkyAperturePhotometryProduct getObject()

Return

AnnularSkyAperturePhotometryProduct

Returns the object for this AnnularSkyAperturePhotometryExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this AnnularSkyAperturePhotometryExplorer.

JTable getParameterTable()

AnnularSkyAperturePhotometryExplorer.

Return

JTable

Returns the parameter table for this AnnularSkyAperturePhotometryExplorer.

JPanel getPlotPanel()

AnnularSkyAperturePhotometryExplorer.

Return

JPanel

Returns the plot panel for this AnnularSkyAperturePhotometryExplorer.

JPanel getSkyIntensityPlotPanel()

AnnularSkyAperturePhotometryExplorer.

Return

JPanel

Returns the plot panel for this AnnularSkyAperturePhotometryExplorer.

JComponent getSkyIntensityPlot()

AnnularSkyAperturePhotometryExplorer.

JComponent getSkyIntensityPlot()

Return

JComponent

Returns the sky intensity plot for this AnnularSkyAperturePhotometryExplorer.

3.10. AnnularSkyAperturePhotometryPanel

Full Name:	herschel.ia.toolbox.image.gui.AnnularSkyAperturePhotometryPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import AnnularSkyAperturePhotometryPanel

Description

A panel for the AnnularSkyAperturePhotometryTask.

API Summary

Constructor
AnnularSkyAperturePhotometryPanel() The constructor of a new AnnularSkyAperturePhotometryPanel.

Methods
JPanel **getAperturesPanel()** Returns the panel where to specify the radii for the
JComponent **getSkyApertureComponent()** Returns the component where to specify the inner and outer
drawFigures() Draws the circles on the image for this
moveConfirmedFigures() Allows the user to move the circles bounding the
clearSkyAperture() Clears the annular sky aperture for this

API details

Constructor

AnnularSkyAperturePhotometryPanel()

Methods

JPanel getAperturesPanel()
apertures for this AnnularSkyAperturePhotometryPanel. **Return** **JPanel** Returns the panel where to specify the radii for the apertures for this AnnularSkyAoerturePhotometryPanel.

JComponent getSkyApertureComponent()
radius of the annular sky aperture for this AnnularSkyAperturePhotometryPanel.

JComponent getSkyApertureComponent()

Return

JComponent

Returns the component where to specify the inner and outer radius of the annular sky aperture for this AnnularSkyAperturePhotometryPanel.

drawFigures()

AnnularSkyAperturePhotometryPanel.

moveConfirmedFigures()

apertures for this AnnularSkyAperturePhotometryPanel.

clearSkyAperture()

AnnularSkyAperturePhotometryPanel.

3.11. AnnularSkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.AnnularSkyAperturePhotometryProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import AnnularSkyAperturePhotometryProduct

Description

A class to deal with the results of aperture photometry with a circular

target aperture and an annular sky aperture.

API Summary

Constructor
`AnnularSkyAperturePhotometryProduct()` The constructor of a new AnnularSkyAperturePhotometryProduct.

Methods
`setRadii(double innerPixels, double outerPixels)` Sets the inner and outer radius in pixels for this
`setRadii(double innerPixels, double innerArcsec, double outerPixels, double outerArcsec)` Sets the inner and outer radius for this
`double` **`getInnerRadiusPixels()`** Returns the inner radius for this AnnularSkyAperturePhotometryProduct in pixels.
`double` **`getInnerRadiusArcsec()`** Returns the inner radius for this AnnularSkyAperturePhotometryProduct
`double` **`getOuterRadiusPixels()`** Returns the outer radius for this AnnularSkyAperturePhotometryProduct in pixels.
`double` **`getOuterRadiusArcsec()`** Returns the outer radius for this AnnularSkyAperturePhotometryProduct in arcsec.
`TableDataset` **`getSkyIntensityPlot()`** Returns the sky intensity plot for this
`Double1d` **`getSkyRadius()`** Returns the sky radius in pixels for the sky intensity
`Double1d` **`getSkyIntensity()`** Returns the sky intensity for the sky intensity plot for this

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

AnnularSkyAperturePhotometryProduct()

Methods

setRadii(double innerPixels, double outerPixels)

AnnularSkyAperturePhotometryProduct to the given numbers of pixels.

Arguments

double **innerPixels** [INPUT, MANDATORY]

double **outerPixels** [INPUT, MANDATORY]

setRadii(double innerPixels, double innerArcsec, double outerPixels, double outerArcsec)

AnnularSkyAperturePhotometryProduct to the given numbers of pixels and arcsec.

Arguments

double **innerPixels** [INPUT, MANDATORY]

double **innerArcsec** [INPUT, MANDATORY]

double **outerPixels** [INPUT, MANDATORY]

double **outerArcsec** [INPUT, MANDATORY]

double getInnerRadiusPixels()

Return

double

Returns the inner radius for this AnnularSkyAperturePhotometryProduct in pixels.

double getInnerRadiusArcsec()

in arcsec.

Return

double

Returns the inner radius for this AnnularSkyAperturePhotometryProduct in arcsec.

double getOuterRadiusPixels()

Return

double

Returns the outer radius for this AnnularSkyAperturePhotometryProduct in pixels.

double getOuterRadiusArcsec()

Return

double

Returns the outer radius for this AnnularSkyAperturePhotometryProduct in arcsec.

TableDataset getSkyIntensityPlot()

AnnularSkyAperturePhotometryProduct.

Return

TableDataset

Returns the sky intensity plot for this AnnularSkyAperturePhotometryProduct.

Double1d getSkyRadius()

plot for this AnnularSkyAperturePhotometryProduct.

Return

Double1d

Returns the sky radius in pixels for the sky intensity plot for this AnnularSkyAperturePhotometryProduct.

Double1d getSkyIntensity()

AnnularSkyAperturePhotometryProduct.

Return

Double1d

Returns the sky intensity for the sky intensity plot for this AnnularSkyAperturePhotometryProduct.

3.12. AnnularSkyAperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.AnnularSkyAperturePhotometryTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import AnnularSkyAperturePhotometryTask

Description

A Task for aperture photometry.

A Task for aperture photometry, using a circular target aperture and an annular sky aperture.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`centerX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`centerY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`centerRA`** `[INPUT, OPTIONAL, default=Default value : "centerRA"]`
`String` **`centerDEC`** `[INPUT, OPTIONAL, default=Default value : "centerDec"]`
`Double` **`radiusPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`radiusArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Integer` **`algorithm`** `[INPUT, MANDATORY, default=Default value : 4]`
`AnnularSkyAperturePhotometryProduct` **`result`** `[OUTPUT, MANDATORY, default=No default value]`
`Double` **`innerPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`innerArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`outerPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`outerArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the target center.

`Double centerY [INPUT, OPTIONAL, default=Default value : NaN]`

The y-pixel-coordinate of the target center.

`String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]`

The right ascension of the target center.

String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the target center.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The target radius (i.e. the radius of the circular target aperture) in pixels.

Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

Integer algorithm [INPUT, MANDATORY, default=Default value : 4]

The sky estimation algorithm.

AnnularSkyAperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

The result.

Double innerPixels [INPUT, OPTIONAL, default=Default value : NaN]

The inner radius of the annular sky aperture in pixels.

Double innerArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The inner radius of the annular sky aperture in arcsec.

Double outerPixels [INPUT, OPTIONAL, default=Default value : NaN]

The outer radius of the annular sky aperture in pixels.

Double outerArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The outer radius of the annular sky aperture in arcsec.

3.13. ANY

Full Name:	herschel.ia.numeric.toolbox.basic.Any
Alias:	ANY
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Any

Description

Determines whether any element in the array evaluates to true.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply ANY on a Int2d: check if any array item, in the specified dimension, is lower than '3'

```
x=Int2d( [ [1,2], [3,4], [5,6] ])
print ANY(x<3)    #any item => 1 (true)
print ANY(x<3, 0) #dim 0 => [ ANY(Int1d([1,3,5]) < 3) , ANY(Int1d([2,4,6]) <
 3) ] = [true, true]
print ANY(x<3, 1) #dim 1 => [ ANY(Int1d([1,2]) < 3) , ANY(Int1d([3,4]) < 3),
 ANY(Int1d([5,6]) < 3) = [true, false, false]
```

API Summary

Jython Syntax
<y>=ALL(<array_expression>[,<dimension>])

Properties
any expression with an integral type **x** [INPUT, MANDATORY, default=no default value]
integer **dimesion** [INPUT, OPTIONAL, default=all the array items]
a boolean array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any expression with an integral type x [INPUT, MANDATORY, default=no default value]

The input array must be of integral type (e.g. bytes,integers) See example.

integer dimesion [INPUT, OPTIONAL, default=all the array items]

The dimension where to apply the expression.

a boolean array y [OUTPUT, MANDATORY, default=no default value]

The result is a boolean array

See also

- ALL

3.14. AnyPresent

Full Name:	herschel.ia.numeric.toolbox.basic.AnyPresent
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import AnyPresent

Description

Tests whether any of the bits in specified mask are present in the elements of the input array.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply AnyPresent on a Int1d: test if a binary 'and' comparison, item by item, with '3' (as mask) is not equals to '0'

```
x=Int1d([0,1,2,3])
print AnyPresent(3)(x)
#=> [  (0 & 3) != 0,   (1 & 3) != 0,   (2 & 3) != 0,   (3 & 3) != 0  ] =
 [false,true,true,true]
print x.apply(AnyPresent(3)) #Another way for using AnyPresent
```

API Summary

Jython Syntax
`<y>=AnyPresent(<bitmask>)(<x>)`

Properties
`any integral type x [INPUT, MANDATORY, default=no default value]`
`any number bitmask [INPUT, MANDATORY, default=no default value]`
`a boolean array y [OUTPUT, MANDATORY, default=no default value]`

API details

Properties

`any integral type x [INPUT, MANDATORY, default=no default value]`

The input array must be of integral type (e.g. bytes,integers)

`any number bitmask [INPUT, MANDATORY, default=no default value]`

Accepts any number.

`a boolean array y [OUTPUT, MANDATORY, default=no default value]`

The result is a boolean array

See also

- AllPresent
- NotPresent

3.15. ApperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.ApperturePhotometryExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import ApperturePhotometryExplorer

Description

An explorer for ApperturePhotometryProducts.

API Summary

Constructors
ApperturePhotometryExplorer() The constructor of a new ApperturePhotometryExplorer.
ApperturePhotometryExplorer(Object object) The constructor of a new ApperturePhotometryExplorer associated

Methods
String **getName()** Returns the name for this ApperturePhotometryExplorer.
String **getDescription()** Returns the description for this ApperturePhotometryExplorer.
boolean **canHandle(Class className)** Checks whether this ApperturePhotometryExplorer can
setObject(Object object) Sets the object for this ApperturePhotometryExplorer to the given object.
ApperturePhotometryProduct **getObject()** Returns the object for this ApperturePhotometryExplorer.
Class **getVariableType()** Returns the expected variable type for this ApperturePhotometryExplorer.
JComponent **getComponent()** Returns the component that is responsible for displaying the
JTable **getParameterTable()** Constructs and returns the parameter table for this
JTable **getResultsTable()** Constructs and returns the results table for this ApperturePhotometryExplorer.
JPanel **getPlotPanel()** Constructs and returns the plot panel for this
JPanel **getCurveOfGrowthPanel()** Constructs and returns the curve of growth for this
JComponent **getCurveOfGrowth()** Constructs and returns the curve of growth for this ApperturePhotometryExplorer.
addDataObjectListener(DataObjectListener listener)

Methods
Add the given listener to this ApperturePhotometryExplorer
removeDataObjectListener(DataObjectListener listener) Removes the given listener for this ApperturePhotometryExplorer,

API details

Constructors

ApperturePhotometryExplorer()

ApperturePhotometryExplorer(Object object)

with the given object.

Argument

Object **object** [INPUT, MANDATORY]

Methods

String getName()

Return

String

Returns the name for this ApperturePhotometryExplorer.

String getDescription()

Return

String

Returns the description for this ApperturePhotometryExplorer.

boolean canHandle(Class className)

handle objects of the given class.

Argument

Class **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this ApperturePhotometryExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

Object **object** [INPUT, MANDATORY]

ApperturePhotometryProduct getObject()

Return

ApperturePhotometryProduct

ApturePhotometryProduct getObject()

Returns the object for this ApturePhotometryExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this ApturePhotometryExplorer.

JComponent getComponent()

data object for this ApturePhotometryExplorer.

Return

JComponent

Returns the component that is responsible for displaying the data object for this ApturePhotometryExplorer.

JTable getParameterTable()

ApturePhotometryExplorer.

Return

JTable

Returns the parameter table for this ApturePhotometryExplorer.

JTable getResultsTable()

Return

JTable

Returns the results table for this ApturePhotometryExplorer.

JPanel getPlotPanel()

ApturePhotometryExplorer.

Return

JPanel

Returns the plot panel for this ApturePhotometryExplorer.

JPanel getCurveOfGrowthPanel()

ApturePhotometryExplorer.

Return

JPanel

Returns the curve of growth for this ApturePhotometryExplorer.

JComponent getCurveOfGrowth()

Return

JComponent getCurveOfGrowth()

JComponent

Returns the curve of growth for this AperturePhotometryExplorer.

addDataObjectListener(DataObjectListener listener)

to receive data object events from it.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

removeDataObjectListener(DataObjectListener listener)

so that it no longer receives data object events by this explorer.

Argument

DataObjectListener **listener** [INPUT, MANDATORY]

3.16. AperturePhotometryPanel

Full Name:	herschel.ia.toolbox.image.gui.AperturePhotometryPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import AperturePhotometryPanel

Description

A panel for aperture photometry.

API Summary

Constructor
`ApperturePhotometryPanel()`
The construction of a new AperturePhotometryPanel.

Methods
`JPanel` **`getImagePanel()`**
Returns a panel that displays the image for this
`JPanel` **`getTargetCenterPanel()`**
Returns the panel where to specify the target center
`adjustTargetCenter()`
Allows the user to drag the circle indicating the target
`boolean` **`isInImage`**`(MouseEvent me)`
Checks whether the given mouse event occurred inside the
`boolean` **`isInImage`**`(double pixX, double pixY)`
Checks whether the given pixel coordinates lies
`JPanel` **`getAperturesPanel()`**
Returns the panel where to specify the radius/radii for
`JPanel` **`getTargetAperturePanel()`**
Returns the panel where to specify the target radius
`JComponent` **`getSkyApertureComponent()`**
Returns the panel where to specify the parameters for
`JPanel` **`getAlgorithmPanel()`**
Returns the panel where to specify the kind of pixels
`JPanel` **`getButtonPanel()`**
Returns the button panel for this AperturePhotometryPanel
`JLabel` **`newLabel`**`(String title, String tooltip)`
Returns a label with the given title and tooltip.
`Border` **`newBorder`**`(String title)`
Returns a border with the given title.
`setVariableSelection``(VariableSelection selection)`
Sets the variable selection for this AperturePhotometryPanel

Methods
setSiteEventHandler(SiteEventHandler handler) Sets the site event handler for this ApperturePhotometryPanel
setTask(TaskApi task) Associates the given task with this ApperturePhotometryPanel.
Display **getDisplay**() Returns the display for this ApperturePhotometryPanel.
Image **getImage**() Returns the image associated with this
TaskApi **getTask**() Returns the task associated with this
Map **getSelectionMap**() Returns the map associated with this
Map **getModifierMap**() Returns the map with the modifiers associated
SiteEventHandler **getHandler**() Returns the site event handler associated with this
double **getTargetRadius**() Returns the target radius for this
trigger() Triggers the execution of the task associated
transferFromModifierToSelectionMap() Transfers the information for this ApperturePhotometryPanel
drawFigures() Draws the figures bounding the apertures for this
moveConfirmedFigures() Allows to move/resize the figures bounding the apertures
clearSkyAperture() Clears the sky aperture for this ApperturePhotometryPanel.

API details

Constructor

ApperturePhotometryPanel()

Methods

JPanel getImagePanel()
PhotometryPanel. **Return** **JPanel** Returns a panel that displays the image for this ApperturePhotometryPanel.

JPanel getTargetCenterPanel()

for this ApperturePhotometryPanel.

Return

JPanel

Returns the panel where to specify the target center for this ApperturePhotometryPanel.

adjustTargetCenter()

center for this ApperturePhotometryPanel.

boolean isInImage(MouseEvent me)

image for this ApperturePhotometryPanel.

Argument

MouseEvent **me** [INPUT, MANDATORY]

Return

boolean

Returns true if the given mouse event occurred inside the image for this ApperturePhotometryPanel.

boolean isInImage(double pixX, double pixY)

inside the image for this ApperturePhotometryPanel.

Arguments

double **pixX** [INPUT, MANDATORY]

double **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the given pixel coordinates lie inside the image for this ApperturePhotometryPanel; false otherwise.

JPanel getAperturesPanel()

the apertures for this ApperturePhotometryPanel.

Return

JPanel

Returns the panel where to specify the radius/radii for the apertures for this ApperturePhotometryPanel.

JPanel getTargetAperturePanel()

for this ApperturePhotometryPanel.

Return

JPanel

Returns the panel where to specify the target radius for this ApperturePhotometryPanel.

JComponent getSkyApertureComponent()

the annular/rectangular sky aperture for this ApperturePhotometryPanel.

JComponent getSkyApertureComponent()

Return

JComponent

Returns the panel where to specify the parameter for the annular/rectangular sky aperture for this AperturePhotometryPanel.

JPanel getAlgorithmPanel()

to use and the sky estimation algorithm for this AperturePhotometryPanel.

Return

JPanel

Returns the panel where to specify the kind of pixels to use and the sky estimation algorithm for this AperturePhotometryPanel.

JPanel getButtonPanel()

Return

JPanel

Returns the button panel for this AperturePhotometryPanel.

JLabel newLabel(String title, String tooltip)

Arguments

`String` **`title`** `[INPUT, MANDATORY]`

`String` **`tooltip`** `[INPUT, MANDATORY]`

Return

JLabel

Returns a label with the given title and tooltip.

Border newBorder(String title)

Argument

`String` **`title`** `[INPUT, MANDATORY]`

Return

Border

Returns a border with the given title.

setVariableSelection(VariableSelection selection)

to the given variable selection.

Argument

`VariableSelection` **`selection`** `[INPUT, MANDATORY]`

setSiteEventHandler(SiteEventHandler handler)

to the given site event handler.

Argument

setSiteEventHandler(SiteEventHandler handler)

SiteEventHandler **handler** [INPUT, MANDATORY]

setTask(TaskApi task)

Argument

TaskApi **task** [INPUT, MANDATORY]

Display getDisplay()

Return

Display

Returns the display for this AperturePhotometryPanel.

Image getImage()

AperturePhotometryPanel.

Return

Image

Returns the image associated with this AperturePhotometryPanel.

TaskApi getTask()

AperturePhotometryPanel.

Return

TaskApi

Returns the task associated with this AperturePhotometryPanel.

Map getSelectionMap()

AperturePhotometryPanel.

Return

Map

Returns the map associated with this AperturePhotometryPanel.

Map getModifierMap()

with this AperturePhotometryPanel.

Return

Map

Returns the map with the modifiers associated with this AperturePhotometryPanel.

SiteEventHandler getHandler()

AperturePhotometryPanel.

Return

SiteEventHandler

Returns the site event handler associated with this AperturePhotometryPanel.

`double getTargetRadius()`

ApperturePhotometryPanel in pixels.

Return

double

Returns the target radius for this ApperturePhotometryPanel in pixels.

`trigger()`

with this ApertrurePhotometryPanel.

`transferFromModifierToSelectionMap()`

from the modifier to the selection map.

`drawFigures()`

ApperturePhotometryPanel on the image.

`moveConfirmedFigures()`

for this ApperturePhotometryPanel.

`clearSkyAperture()`

3.17. ApperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.ApperturePhotometryProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import ApperturePhotometryProduct

Description

A class to deal with the results of aperture photometry.

API Summary

Constructor
`ApperturePhotometryProduct()` The constructor of a new ApperturePhotometryProduct.

Methods
`setTargetCenter(double centerX, double centerY)` Sets the target center for this ApperturePhotometryProduct to
`setTargetCenter(double centerX, double centerY, String centerRA, String centerDec)` Sets the target center for this ApperturePhotometryproduct
`setTargetRadius(double pixels)` Sets the target radius for this ApperturePhotometryProduct
`setTargetRadius(double pixels, double arcsec)` Sets the target radius for this ApperturePhotometryProduct
`setPixels(boolean fractional)` Sets the type of pixels (entire / fractional) used for this
`setUnit(Unit unit)` Sets the unit for this ApperturePhotometryProduct to the given unit.
`setResultsTable(Double2d resultsTable)` Sets the results table for this ApperturePhotometryProduct to the
`setCurveOfGrowth(Double1d growthRadius, Double1d growthFlux)` Sets the curve of growth for this ApperturePhotometryProduct for the given
`Double1d` **`getTargetCenterPixelCoordinates()`** Returns the target center for this ApperturePhotometryProduct in
`String1d` **`getTargetCenterSkyCoordinates()`** Returns the target center for this ApperturePhotometryProduct in
`double` **`getTargetRadiusPixels()`** Returns the target radius for this ApperturePhotometryProduct in pixels.
`double` **`getTargetRadiusArcsec()`** Returns the target radius for this ApperturePhotometryProduct in arcsec.
`String` **`getPixels()`** Returns the String representation of the type of pixels (fractional / entire) used

Methods
`String` **`getUnit()`** Returns the unit for this AperturePhotometryProduct.
`TableDataset` **`getTable()`** Returns the results table for this AperturePhotometryProduct.
`Double2d` **`getDouble2dTable()`** Returns the results table for this AperturePhotometryProduct as a Double2d.
`Double1d` **`getTotal()`** Returns the total flux for the target, the sky and the target from
`Double1d` **`getNbOfPixels()`** Returns the number of pixels for the target, the sky and the target from
`Double1d` **`getIntensityPerPixel()`** Returns the intensity per pixel for the target, the sky and the
`Double1d` **`getError()`** Returns the error for the target, the sky and the target from which the
`double` **`getTargetPlusSkyTotal()`** Returns the total flux for the target (sky included) for this
`double` **`getNbOfTargetPlusSkyPixels()`** Returns the number of pixels for the target (sky included) for this
`double` **`getIntensityPerTargetPlusSkyPixel()`** Returns the intensity per pixel for the target (sky included) for this
`double` **`getTargetPlusSkyError()`** Returns the error for the target (sky included) for this AperturePhotometryProduct.
`double` **`getIntensityPerSkyPixel()`** Returns the intensity per pixel for the sky for this AperturePhotometryProduct.
`double` **`getTargetTotal()`** Returns the total flux for the target (sky subtracted) for this
`double` **`getNbOfTargetPixels()`** Returns the number of pixels for the target (sky subtracted) for this
`double` **`getIntensityPerTargetPixel()`** Returns the intensity per pixel for the target (sky subtracted) for this
`double` **`getTargetError()`** Returns the error for the target (sky subtracted) for this
`TableDataset` **`getCurveOfGrowth()`** Returns the curve of growth for this AperturePhotometryProduct.
`Double1d` **`getGrowthRadius()`** Returns the growth radius for this AperturePhotometryProduct.
`Double1d` **`getGrowthFlux()`** Returns the growth flux for this AperturePhotometryProduct,

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

ApertureFotometryProduct()

Methods

setTargetCenter(double centerX, double centerY)

the given pixel coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

setTargetCenter(double centerX, double centerY, String centerRA, String centerDec)

to the given pixel and sky coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

String **centerRA** [INPUT, MANDATORY]

String **centerDec** [INPUT, MANDATORY]

setTargetRadius(double pixels)

to the given number of pixels.

Argument

double **pixels** [INPUT, MANDATORY]

setTargetRadius(double pixels, double arcsec)

to the given number of pixels and arcsec.

Arguments

double **pixels** [INPUT, MANDATORY]

double **arcsec** [INPUT, MANDATORY]

setPixels(boolean fractional)

ApertureFotometryProduct.

Argument

boolean **fractional** [INPUT, MANDATORY]

setUnit(Unit unit)

Argument

Unit **unit** [INPUT, MANDATORY]

setResultsTable(Double2d resultsTable)

given table.

setResultsTable(Double2d resultsTable)

Argument

Double2d **resultsTable** [INPUT, MANDATORY]

setCurveOfGrowth(Double1d growthRadius, Double1d growthFlux)

growth radius and growth flux.

Arguments

Double1d **growthRadius** [INPUT, MANDATORY]

Double1d **growthFlux** [INPUT, MANDATORY]

Double1d getTargetCenterPixelCoordinates()

pixel coordinates.

Return

Double1d

Returns the target center for this AperturePhotometryProduct in pixel coordinates.

String1d getTargetCenterSkyCoordinates()

sky coordinates.

Return

String1d

Returns the target center for this AperturePhotometryProduct in sky coordinates.

double getTargetRadiusPixels()

Return

double

Returns the target radius for this AperturePhotometryProduct in pixels.

double getTargetRadiusArcsec()

Return

double

Returns the target radius for this AperturePhotometryProduct in arcsec.

String getPixels()

for this AperturePhotometryProduct.

Return

String

Returns the String representation of the type of pixels (fractional / entire) used for this AperturePhotometryProduct.

String getUnit()

Return

String getUnit()

String

Returns the unit for this AperturePhotometryProduct.

TableDataset getTable()

Return

TableDataset

Returns the results table for this AperturePhotometryProduct.

Double2d getDouble2dTable()

Return

Double2d

Returns the results table for this AperturePhotometryProduct as a Double2d.

Double1d getTotal()

which the sky has been subtracted for this AperturePhotometryProduct.

Return

Double1d

Returns the total flux for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

Double1d getNbOfPixels()

which the sky has been subtracted.

Return

Double1d

Returns the number of pixels for the target, the sky and the target from which the sky has been subtracted.

Double1d getIntensityPerPixel()

target from which the sky has been subtracted for this AperturePhotometryProduct.

Return

Double1d

Returns the intensity per pixel for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

Double1d getError()

sky has been subtracted for this AperturePhotometryProduct.

Return

Double1d

Returns the error for the target, the sky and the target from which the sky has been subtracted for this AperturePhotometryProduct.

`double getTargetPlusSkyTotal()`

ApperturePhotometryProduct.

Return

double

Returns the total flux for the target (sky included) for this ApperturePhotometryProduct.

`double getNbOfTargetPlusSkyPixels()`

ApperturePhotometryProduct.

Return

double

Returns the number of pixels for the target (sky included) for this ApperturePhotometryProduct.

`double getIntensityPerTargetPlusSkyPixel()`

ApperturePhotometryProduct.

Return

double

Returns the intensity per pixel for the target (sky included) for this ApperturePhotometryProduct.

`double getTargetPlusSkyError()`

Return

double

Returns the error for the target (sky included) for this ApperturePhotometryProduct.

`double getIntensityPerSkyPixel()`

Return

double

Returns the intensity per pixel for the sky for this ApperturePhotometryProduct.

`double getTargetTotal()`

ApperturePhotometryProduct.

Return

double

Returns the total flux for the target (sky subtracted) for this ApperturePhotometryProduct.

`double getNbOfTargetPixels()`

ApperturePhotometryProduct.

Return

double

Returns the number of pixels for the target (sky subtracted) for this ApperturePhotometryProduct.

`double getIntensityPerTargetPixel()`

ApperturePhotometryProduct.

Return

double

Returns the intensity per pixel for the target (sky subtracted) for this ApperturePhotometryProduct.

`double getTargetError()`

ApperturePhotometryProduct.

Return

double

Returns the error for the target (sky subtracted) for this ApperturePhotometryProduct.

`TableDataset getCurveOfGrowth()`

Return

TableDataset

Returns the curve of growth for this ApperturePhotometryProduct.

`Double1d getGrowthRadius()`

Return

Double1d

Returns the growth radius for this ApperturePhotometryProduct.

`Double1d getGrowthFlux()`

Return

Double1d

Returns the growth flux for this ApperturePhotometryProduct.

3.18. AperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.AperturePhotometryTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import AperturePhotometryTask

Description

An abstract Task for aperture photometry.

Three Tasks implement this interface. They all use a circular target aperture. Two use a sky aperture (annular/rectangular) to estimate the sky with one of the five sky estimation algorithms. The third one uses a fixed value for the sky.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`centerX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`centerY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`centerRA`** `[INPUT, OPTIONAL, default=Default value : "centerRA"]`
`String` **`centerDec`** `[INPUT, OPTIONAL, default=Default value : "centerDec"]`
`Double` **`radiusPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Integer` **`radiusArcsec`** `[INPUT, OPTIONAL, default=Default value : 4]`
`boolean` **`fractional`** `[INPUT, OPTIONAL, default=Default value : true]`
`AperturePhotometryProduct` **`result`** `[OUTPUT, MANDATORY, default=No default value]`

Miscellaneous

In the current version two subclasses extend this class : - an abstract class for aperture photometry for which the sky is estimated through a sky aperture (annular/rectangular) and a sky estimation algorithm - a class for aperture photometry for which to sky has a fixed value, given by the user In both cases, a circular target aperture is used.

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the target center.

`Double centerY [INPUT, OPTIONAL, default=Default value : NaN]`

The y-pixel-coordinate of the target center.

String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]

The right ascension of the target center.

String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the target center.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The radius of the circular target aperture in pixels.

Integer radiusArcsec [INPUT, OPTIONAL, default=Default value : 4]

The radius of the circular target aperture in arcsec.

boolean fractional [INPUT, OPTIONAL, default=Default value : true]

The type of pixels.

AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

The result.

3.19. ARCCOS

Full Name:	herschel.ia.numeric.toolbox.basic.ArcCos
Alias:	ARCCOS
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import ArcCos

Description

Computes the inverse cosine of a number or array, and may be of any type.

Gives the inverse cosine of a number or array. The input must be in the range [-1,1], and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

Example

Example 1: Apply ARCCOS on a Float1d

```
x=Float1d([0,0.5])
print  ARCCOS(x) # [1.5707964,1.0471976]
```

API Summary

Jython Syntax

```
<y>=ARCCOS(<x>)
```

Properties

```
any type x [INPUT, MANDATORY, default=no default value]
radians y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

must be in the range [-1,1], and may be of any type.

radians y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- COS
- COSH

3.20. ARCSIN

Full Name:	herschel.ia.numeric.toolbox.basic.ArcSin
Alias:	ARCSIN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import ArcSin

Description

Computes the inverse sine of a number or array, and may be of any type.

Gives the inverse sine of a number or array. The input must be in the range [-1,1], and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

Example

Example 1: Apply ARCSIN on a Float1d

```
x=Float1d([0,0.5])
print  ARCSIN(x) # [0.0,0.5235988]
```

API Summary

Jython Syntax
<y>=ARCSIN(<x>)

Properties
any type **x** [INPUT, MANDATORY, default=no default value]
radians **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

must be in the range [-1,1], and may be of any type.

radians y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- SIN
- SINH

3.21. ARCTAN

Full Name:	herschel.ia.numeric.toolbox.basic.ArcTan
Alias:	ARCTAN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import ArcTan

Description

Computes the inverse tangent of a number or array, and may be of any type.

Gives the inverse tangent of a number or array. The input must be in the range [-1,1], and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

Example

Example 1: Apply ARCTAN on a Float1d

```
x=Float1d([0,0.5])
print  ARCTAN(x) # [0.0,0.4636476]
```

API Summary

Jython Syntax
<y>=ARCTAN(<x>)

Properties
any type **x** [INPUT, MANDATORY, default=no default value]
radians **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

must be in the range [-1,1], and may be of any type.

radians y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- TAN
- TANH

3.22. ArithmeticSpectrumTask

Full Name:	herschel.ia.toolbox.spectrum.ArithmeticSpectrumTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import ArithmeticSpectrumTask

Description

Base task for applying a scalar operation to the flux data in a spectrum container or processing two spectrum containers on a spectrum-by-spectrum basis.

For the scalar mode, 'ds' and 'param' need to specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1,3,4,2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1,3,4]` and for the pair operation `segments1=[1,3,4], segments1=[3,6,5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the orginal datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

API Summary

Properties
`SpectrumContainer` **`ds`** `[INPUT, OPTIONAL, default=no default value.]`
`Double` **`param`** `[INPUT, OPTIONAL, default=no default value.]`
`Object` **`selection`** `[INPUT, OPTIONAL, default=None.]`
`PyDictionary|Map>` **`selection_lookup`** `[INPUT, OPTIONAL, default=no default value.]`
`PyList` **`selection_index`** `[INPUT, OPTIONAL, default=No default value.]`
`Boolean` **`overwrite`** `[INPUT, OPTIONAL, default=False.]`
`SpectrumContainer` **`ds1`** `[INPUT, OPTIONAL, default=no default value.]`

Properties
SpectrumContainer **ds2** [INPUT, OPTIONAL, default=no default value.]
Object **segments** [INPUT, OPTIONAL, default=no default value.]
Object **segments1** [INPUT, OPTIONAL, default=no default value.]
Object **segments2** [INPUT, OPTIONAL, default=no default value.]
String **variant** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]

Input container to be processed by the task in case the task is used as a scalar operation.

Double param [INPUT, OPTIONAL, default=no default value.]

The scalar parameter to be considered when the task is used as scalar operation.

Object selection [INPUT, OPTIONAL, default=None.]

Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra for which the add should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

History

- `2009-05-20 - meli`: Initial.

3.23. asciiTableReader

Full Name:	herschel.ia.toolbox.util.AsciiTableReaderTask
Alias:	asciiTableReader
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import AsciiTableReaderTask
Category:	utility task

Description

AsciiTableReaderTask

Task for reading ascii files.

Examples

Example 1: Read a raw space-separated ascii table

```
# test_space.dat contents
# start
1 2 3
2 3 4
5 6 7
# end
from herschel.ia.io.ascii import AsciiParser
atrt = AsciiTableReaderTask()
tds = atrt(file="test_space.dat", \
    parserDelim=" ", \
    #template=TableTemplate(3,names=["a","b","c"]), \
    parserIgnore= '^\s*$|^\s*#', \
    parserIgnoreWarn= 1, \
    parserGuess=AsciiParser.GUESS_TRY)
# tds will hold a TableDataset with the data, blank lines will be ignored, and
 warnings will be issued for ignored lines
```

Example 2: Read a file

```
atrt = AsciiTableReaderTask()
tableDataset = atrt(file="file.ascii")
print tableDataset
```

Example 3: Read a file without HCSS header (first row does not contain names), delimiter is tab

```
atrt = AsciiTableReaderTask()
tableDataset = atrt(file=&quot;file.ascii&quot;, parserDelim='\t',
 parserGuess=AsciiParser.GUESS_TRY)
```

Example 4: Read a file without HCSS header (first row contains names), delimiter is tab.

```
tableDataset = atrt(file=&quot;file.ascii&quot;, parserDelim='\t',
 parserGuess=AsciiParser.GUESS_TRY, parseNames=1)
```

API Summary

Jython Syntax

```
tableDataset=AsciiTableReaderTask()(file=&quot;file.ascii&quot;)
```

Jython Syntax
`see examples`

Properties
`String` **`file`** `[INPUT, MANDATORY, default=null]`
`String` **`table`** `[OUTPUT, MANDATORY, default=null]`
`String` **`configFile`** `[INPUT, OPTIONAL, default=null]`
`String` **`configFileOutput`** `[INPUT, OPTIONAL, default=null]`
`AsciiParser` **`parser`** `[INPUT, OPTIONAL, default=default AsciiTableTool parser]`
`String` **`parserIgnore`** `[INPUT, OPTIONAL, default=default AsciiParser ignore value.]`
`Boolean` **`parserIgnoreWarn`** `[INPUT, OPTIONAL, default=default AsciiTableTool warnWhenIgnore value (false).]`
`Integer` **`parserSkip`** `[INPUT, OPTIONAL, default=default AsciiParser skipping rows value.]`
`Boolean` **`parserTrim`** `[INPUT, OPTIONAL, default=default AsciiParser trim rows value.]`
`Integer` **`parserGuess`** `[INPUT, OPTIONAL, default=GUESS_NONE Guess behavior.]`
`String` **`parserDelim`** `[INPUT, OPTIONAL, default=comma separator.]`
`Boolean` **`parseNames`** `[INPUT, OPTIONAL, default=default AsciiParser parseNames value.]`
`TableTemplate` **`template`** `[INPUT, OPTIONAL, default=extracted from the first file rows.]`

API details

Properties

`String file [INPUT, MANDATORY, default=null]`
Input file.

`String table [OUTPUT, MANDATORY, default=null]`
TableDataset object loaded.

`String configFile [INPUT, OPTIONAL, default=null]`
Configuration file where the formatter (AsciiFormatter), parser (AsciiParser) and table template (TableTemplate) must be specified. When configFile parameter is specified, any parameter related to parser or to table template are not allowed.

`String configFileOutput [INPUT, OPTIONAL, default=null]`
If a file is specified, an output configuration file will be created.

`AsciiParser parser [INPUT, OPTIONAL, default=default AsciiTableTool parser]`
AsciiParser object

String parserIgnore [INPUT, OPTIONAL, default=default AsciiParser ignore value.]

String expression to ignore when parsing a file. AsciiParser.setIgnore(expression) is called.

Boolean parserIgnoreWarn [INPUT, OPTIONAL, default=default AsciiTableTool warnWhenIgnore value (false).]

Specifies if the parser must issue a warning if a line is ignored (emit only if true). AsciiTableTool.setWarnWhenIgnore(expression) is called.

Integer parserSkip [INPUT, OPTIONAL, default=default AsciiParser skipping rows value.]

Number of rows to skip when reading a file. AsciiParser.setSkip(number of lines) is called.

Boolean parserTrim [INPUT, OPTIONAL, default=default AsciiParser trim rows value.]

Specifies if the parser must trim each row when reading a file. AsciiParser.setTrim(strip lines) is called.

Integer parserGuess [INPUT, OPTIONAL, default=GUESS_NONE Guess behavior.]

Specifies if the parser should guess column types. Files should not contain HCSS header (use skip=AsciiReader.HCSS_HEADER for skipping HCSS header or comment these lines) AsciiParser.setGuess(guessType) is called. Valid options: -AsciiParser.GUESS_NONE: (default) file must contains template or template must be provided (no guess) -AsciiParser.GUESS_TRY: guess types based on the first 100 records -AsciiParser.GUESS_ALL: guess types based on all records -AsciiParser.ALL_STRING: each record is a string (no guess required) -AsciiParser.ALL_BOOLEAN: each record is a boolean (no guess required) -AsciiParser.ALL_BYTE: each record is a byte (no guess required) -AsciiParser.ALL_INTEGER: each record is an integer (no guess required) -AsciiParser.ALL_LONG: each record is a long (no guess required) -AsciiParser.ALL_FLOAT: each record is a float (no guess required) -AsciiParser.ALL_DOUBLE: each record is a double (no guess required) -AsciiParser.ALL_COMPLEX: each record is a complex (no guess required)

String parserDelim [INPUT, OPTIONAL, default=comma separator.]

Specifies the field separator. If it is one character, a csvParser is selected. If it is an expression, a RegExpParser is selected.

Boolean parseNames [INPUT, OPTIONAL, default=default AsciiParser parseNames value.]

Specifies if the parser must read column names when reading a file. AsciiParser.firstRowAreColumnNames(expression) is called.

TableTemplate template [INPUT, OPTIONAL, default=extracted from the first file rows.]

Specifies the data structure.

History

- `2007-11-22 - JCS`: initial version
- `2008-08-22 - JDS`: added optional parameter parserIgnoreWarn (SCR #3674), serialVersionUID, parameter descriptions, better jython doc

3.24. asciiTableWriter

Full Name:	herschel.ia.toolbox.util.AsciiTableWriterTask
Alias:	asciiTableWriter
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import AsciiTableWriterTask
Category:	utility task

Description

Task for writing ASCII files.

Example

Example 1: write a table in an text file

```
...
atwt = AsciiTableWriterTask()
atwt(file="file.ascii", table=TableDataset())
...
```

API Summary

Jython Syntax

```
asciiTableWriter(<file>, <table> [, <configFile>,
<configFileOutput>, <formatter>, \
<formatterHeader>, <formatterCommented>, <formatterCommentPrefix>,
<template>])
```

Properties

```
String file [INPUT, true, default=null]
TableDataset table [INPUT, true, default=null]
String configFile [INPUT, false, default=null]
String configFileOutput [INPUT, false, default=null]
AsciiFormatter formatter [INPUT, false, default=default
AsciiTableTool formatter.]
Boolean formatterHeader [INPUT, false, default=default
AsciiFormatter header allowance value.]
Boolean formatterCommented [INPUT, false, default=default
AsciiFormatter comments allowance value.]
String formatterCommentPrefix [INPUT, false, default=default
AsciiFormatter comments prefix value.]
TableTemplate template [INPUT, false, default=extracted from the
first file rows.]
```

API details

Properties

String file [INPUT, true, default=null]

Output file.

TableDataset table [INPUT, true, default=null]

TableDataset object to be saved.

String configFile [INPUT, false, default=null]

Configuration file where the formatter (AsciiFormatter), parser (AsciiParser) and table template (TableTemplate) must be specified. When configFile parameter is specified, any parameter related to parser or to table template are not allowed.

String configFileOutput [INPUT, false, default=null]

If a file is specified, an output configuration file will be created.

AsciiFormatter formatter [INPUT, false, default=default AsciiTableTool formatter.]

AsciiFormatter object

Boolean formatterHeader [INPUT, false, default=default AsciiFormatter header allowance value.]

Specifies allowance of header information AsciiFormatter.setHeader(true/false) is called.

Boolean formatterCommented [INPUT, false, default=default AsciiFormatter comments allowance value.]

Specifies allowance of comments when writing a file. AsciiFormatter.setCommented(true/false) is called.

String formatterCommentPrefix [INPUT, false, default=default AsciiFormatter comments prefix value.]

Specifies the prefix of all comments. AsciiFormatter.setCommentPrefix(expression) is called.

TableTemplate template [INPUT, false, default=extracted from the first file rows.]

Specifies the data structure.

History

- 22-11-2007 JCS

3.25. AttribQuery

Full Name:	herschel.ia.pal.query.AttribQuery
Type:	Java Class -
Import:	from herschel.ia.pal.query import AttribQuery

Description

Attribute Query formulates a query on the attributes of a Product.

In principle this can be the fastest query on the Product Access Layer. Known attributes are:

- type : String
- creator : String
- creationDate: FineTime
- startDate : FineTime
- endDate : FineTime
- modelName : String
- instrument : String
- description : String

Example

Example 1: Example of a query on attributes

```
date = SimpleTimeFormat().parse("2008-10-31T12:00:00
 TAI").microsecondsSince1958()
q = AttribQuery(Product, "p", "p.creationDate < " + str(date) + "L and
 p.creator = 'Me'")
```

See also

- Querying

3.26. AutomaticContourTask

Full Name:	herschel.ia.toolbox.image.AutomaticContourTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import AutomaticContourTask

Description

A Task for making contours.

A Task for making contours for a given number of contour levels aith a given distribution between the given extreme contour values.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Integer` **`levels`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`min`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`max`** `[INPUT, MANDATORY, default=No default value]`
`String` **`distribution`** `[INPUT, MANDATORY, default=No default value]`
`ImageContour` **`contours`** `[OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Integer levels [INPUT, MANDATORY, default=No default value]`

The number of contour levels.

`Double min [INPUT, MANDATORY, default=No default value]`

The minimum contour value.

`Double max [INPUT, MANDATORY, default=No default value]`

The maximum contour value.

`String distribution [INPUT, MANDATORY, default=No default value]`

The distribution of the contour values.

`ImageContour contours [OUTPUT, MANDATORY, default=No default value]`

The ImageContour.

3.27. AverageSpectrum

Full Name:	herschel.ia.toolbox.spectrum.AverageSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import AverageSpectrum

Description

Task for averaging the individual spectra included in one or several `SpectrumContainer` (see in `herschel.ia.dataset.spectrum`).

The averaging operation is applied to the spectral segments included in the different spectra. For other attributes characterizing the spectrum information and included in the same spectrum container, suitable defaults can be defined (depending on the runtime types or the 'instrument' metadata element; typically, these defaults are defined by the ICC's). A further option is that the user can register such operations from the command line:

```
avg.register("integrations", "ADD")
```

The input data with the spectra to be averaged can be provided in different forms: Either any data object that implements `SpectrumContainer` or any array of such. However, for a successful processing of the task, the data included in the containers should all be consistent. This means that the number of segments found in all the spectra in the containers should be the same and the lengths of these segments should be consistent. Otherwise an exception should be thrown.

Different selection schemes are available for selecting the point spectra or the segments to be averaged. The most simple scheme is to specify lists of point spectrum indices (`selection=[1,3,4,2]`). In this case, the average is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider in the averaging. In the most simple case, you can just specify the indices of segments to be considered `segments=[1,3,4]`. In more advanced situations you can use a `SegmentSelection` object.

Sometimes, the selection model also provides a natural segmentation of the data into groups. If you want to calculate the average on a per group basis then you can set the per_group flag to true.

Using the keyword grouping the user may specify a grouping model which allows to partition the spectra included in the container into suitable sub-groups. Here, when a grouping model is specified, the average is calculated for each sub-group separately. Grouping and selections can be combined such that the selection model(s) impose(s) constraints on the spectra to be included in the groups.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import AverageSpectrum
avg = AverageSpectrum()
avg.register("integrations","ADD")
averagedSpectra = avg(ds=spectra)
spectraOut = avg(ds=spectra, selection={"bbtype":[6031, 6032]})
spectraOut = avg(ds=spectra, selection=[0,1,2,3])
spectraOut = avg(ds=spectra, selection={"LoFrequency":(4000.0,5000.0)})
spectraOut = avg(ds=spectra, selection={"Chopper":([-4.4,5.9],0.1))
```

Example 1: from Jide:

```
spectraOut = avg(ds=spectra, selection={"Chopper":([-4.4,5.9],0.1),
 segments=[1,2])
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
spectraOut = avg(ds=spectra, selection=RangesSelectionModel("Chopper",
 [-4.4,5.9], 0.1)) # the same as above
# or all combined:
spectraOut = avg(ds=spectra, selection={"bbtype":[6031, 6032], "Chopper":
([-4.4,5.9],0.1), "LoFrequency":(4000.0,5000.0))
from herschel.ia.toolbox.spectrum.selections.models import RangesGroupingModel
averagedSpectraPerGroup = avg(ds=spectra,
 grouping=RangesGroupingModel("Chopper", 0.1})
```

API Summary

Properties
Object **ds** [INPUT, OPTIONAL, default=no default value.]
Object **selection** [INPUT, OPTIONAL, default=None.]
PyDictionary|Map> **selection_lookup** [INPUT, OPTIONAL, default=no default value.]
PyList **selection_index** [INPUT, OPTIONAL, default=No default value.]
Object **segments** [INPUT, OPTIONAL, default=no default value.]
String **variant** [INPUT, OPTIONAL, default=no default value.]
Boolean **per_group** [INPUT, OPTIONAL, default=no default value.]
GroupingModel **grouping** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

Object ds [INPUT, OPTIONAL, default=no default value.]

Input data to be processed by the task. Several types are possible:

- SpectrumContainer
- Array of SpectrumContainer, i.e. SpectrumContainer[]
- List of SpectrumContainer, i.e. List
- Any product with implementations of `SpectrumContainer` inside.

Object selection [INPUT, OPTIONAL, default=None.]

Specification of what point spectra the average should be restricted to. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra for which the average should be taken.

Object selection [INPUT, OPTIONAL, default=None.]

- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

Boolean per_group [INPUT, OPTIONAL, default=no default value.]

Specify that the average should be calculated on a per group basis - once a selection model is specified that provides a natural grouping of the data.

GroupingModel grouping [INPUT, OPTIONAL, default=no default value.]

Grouping model that can be used to partition the point spectra into groups and calculate the average on a per group basis. When a grouping model is specified, the 'per_group'-flag is obsolete.

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also

- ManyToOneSpectrumTask

3.28. bg

Full Name:	herschel.ia.toolbox.util.BackgroundTask
Alias:	bg
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import BackgroundTask
Category:	task

Description

Execute jython commands in the background

The bg task is used to execute jython commands in a background Thread. The Thread is returned to allow Thread operations.

Example

Example 1: save specified variable names and values

```
bg("myfunc()")
```

API Summary

Jython Syntax
`bg(<command>)`

Properties
`String command [INPUT, YES, default=No default value]`
`String command [OUPUT, YES, default=The created Tread]`

Limitations

Only applicable from a jide session

Miscellaneous

No miscellaneous

API details

Properties

`String command [INPUT, YES, default=No default value]`

The name of command to be executed

`String command [OUPUT, YES, default=The created Tread]`

Return the Thread created internally to allow Thread operations

See also

- ???

History

- `2004-07-13 - NdC`: first release.

3.29. Bilinear

Full Name:	herschel.ia.numeric.toolbox.interp.Bilinear
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.interp import Bilinear
Category:	numerics/interp

Description

Bilinear Algorithm

The Bilinear class uses bilinear interpolation algorithm to compute the value at the point (x,y) on a regular grid where the images[i, j] i=0, m-1 and j=0, n-1 are known. The nearest neighbor extrapolation is used when the point (x,y) falls outside the boundary of the regular grid if the extrapolation is requested.

The Bilinear has the following arguments:

images: Input, a double2d data array

x: Input, a location array

y: Input, a location array

grid:The grid keyword controls how the location arrays specify where interpolates are desired.

If grid is not set, the location arrays, x, y must have the same number of elements. The result has the same structure and number of elements as x. Let n be the number of elements in x and y and the data array is images[n, n], and the result is also an one dimension array of size n.

If grid is set, let m be the number of elements in x, let n be the number of elements in y. The result has dimensions [m, n]. The element [i,j] of the result contains the value of images interpolated at position (xi, yj).

missing: The value to return for elements outside the bounds of images.

If this value is not specified, interpolated positions that fall outside the bounds of the array images - that is, elements of the x or y arguments that are either less than zero or greater than the largest subscript in the corresponding dimension of images - are set equal to the value of the nearest element of the images.

Examples

Example 1: All data points are inside the boundary

```
from herschel.ia.numeric.toolbox.interp import  Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic  import *
images=RESHAPE(Double1d.range(16), [4,4])
x=[0.0, 1.5]
y=[0.5, 2.5]
bi_imag =Bilinear(Bilinear.GRID, x,y)(images)
print bi_imag
  [
    [0.5,2.5],
    [6.5,8.5]
   ]
```

Example 2: Some data points are on the boundary without GRID is set (the default)

```
from herschel.ia.numeric.toolbox.interp import  Bilinear
```

Example 2: Some data points are on the boundary without GRID is set (the default)

```
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic  import *
images=RESHAPE(Double1d.range(16), [4,4])
x=[0.0, 1.5, 3]
y=[0.5, 1.0, 2.5]
bi_imag =Bilinear(x,y)(images)
print bi_imag
[0.5,7.0,14.0]
```

Example 3: Some data points are on the boundary with GRID is set

```
from herschel.ia.numeric.toolbox.interp import  Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic  import *
images=RESHAPE(Double1d.range(16), [4,4])
x=[0.0, 1.5, 3]
y=[0.5, 1.0, 2.5]
bi_imag =Bilinear(Bilinear.GRID,x,y)(images)
print bi_imag
  [
  [0.5,1.0,2.5],
  [6.5,7.0,8.5],
  [12.5,13.0,14.5]
  ]
```

Example 4: Input x and y coordinate have different array size when GRID is set.

```
images=RESHAPE(Double1d.range(16), [4,4])
x=[0.0, 1.0]
y=[0.5, 1.0, 2.5, 3.5]
bi_imag=Bilinear(Bilinear.GRID,x,y)(images)
print bi_imag
[
    [0.5,1.0,2.5,3.0],
    [4.5,5.0,6.5,7.0]
]
```

Example 5: Some data points are outside boundary with missing=-1 and GRID is set.

```
from herschel.ia.numeric.toolbox.interp import  Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic  import *
images=RESHAPE(Double1d.range(16), [4,4])
x=[-0.5,2]
y=[0.5,1.5]
bi_imag1=Bilinear(Bilinear.GRID, x,y, -1.0)(images)
print bi_imag1
[
    [-1.0,-1.0],
    [8.5,9.5]
]
```

Example 6: Some data points are outside boundary with missing=-2 and without GRID is set

```
from herschel.ia.numeric.toolbox.interp import  Bilinear
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.basic import *
images=RESHAPE(Double1d.range(16), [4,4])
x=Double1d.range(3).add(1.5)
y=Double1d.range(3).add(1.5)
img1=Bilinear(Bilinear.GRID, x,y, -2.0)(images)
print img1
[7.5,12.5,-2.0]
```

Example 7: Java example

```
import herschel.ia.numeric.Double1d;
import herschel.ia.numeric.Double2d;
import herschel.ia.numeric.toolbox.basic.Reshape;
import herschel.ia.numeric.toolbox.interp.Bilinear;
public class Test {
   public static void main(String[] args) {
     Double2d images=(Double2d) Double1d.range(16).apply(new Reshape(4,4));
     Double1d x=Double1d.range(3).add(0.5);
     Double1d y=Double1d.range(3).add(0.5);
     Double2d newImages1=(Double2d) images.apply(new Bilinear(Bilinear.GRID,x,
 y));
     Double1d newImages2=(Double1d) images.apply(new Bilinear(x, y));
     Double2d newImages3=(Double2d) new Bilinear(Bilinear.GRID,x,
 y ).of(images);
     Double1d newImages4=(Double1d) new Bilinear(x, y).of(images);
    System.out.println("newImages1="+newImages1);
    System.out.println("newImages2="+newImages2);
    System.out.println("newImages3="+newImages3);
    System.out.println("newImages4="+newImages4);
   }
}
  newImages1=[
[2.5,3.5,4.5],
[6.5,7.5,8.5],
[10.5,11.5,12.5]
]
newImages2=[2.5,7.5,12.5]
newImages3=[
[2.5,3.5,4.5],
[6.5,7.5,8.5],
[10.5,11.5,12.5]
]
newImages4=[2.5,7.5,12.5]
```

API Summary

Constructors
Bilinear(Double1d x, Double1d y)
Bilinear(double[] x, double[] y)
Bilinear(Boolean grid, Double1d x, Double1d y)
Bilinear(Boolean grid, double[] x, double[] y)
Bilinear(Double1d x, Double1d y, double missing)
Bilinear(double[] x, double[] y, double missing)
Bilinear(Boolean grid, Double1d x, Double1d y, double missing)
Bilinear(Boolean grid, double[] x, double[] y, double missing)

Method
ArrayData **of**(Double2d images)

API details

Constructors

Bilinear(Double1d x, Double1d y)

Arguments

Double1d **x** [INPUT, MANDATORY, default=no default value]

Bilinear(Double1d x, Double1d y)

The array containing the x "virtual subscripts" of the images for which to interpolate values.

`Double1d` **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

Errors

`Array`

size error The length of array x and y has to be equal. The exception will be thrown if the lengths of the array x and y are not equal.

Bilinear(double[] x, double[] y)

Arguments

`double[]` **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

`double[]` **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

Errors

`Array`

size error The length of array x and y has to be equal. The exception will be thrown if the lengths of the array x and y are not equal.

Bilinear(Boolean grid, Double1d x, Double1d y)

Arguments

`Boolean` **grid** [INPUT, MANDATORY, default=no default value]

If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.

`Double1d` **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

`Double1d` **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

Bilinear(Boolean grid, double[] x, double[] y)

Arguments

`Boolean` **grid** [INPUT, MANDATORY, default=no default value]

If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.

`double[]` **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

`double[]` **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

Bilinear(Double1d x, Double1d y, double missing)

Arguments

Bilinear(Double1d x, Double1d y, double missing)

Double1d **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

Double1d **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

double **missing** [INPUT, MANDATORY, default=no default value]

When missing is given, the value outside the boundary is set to missing.

Bilinear(double[] x, double[] y, double missing)

Arguments

double[] **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

double[] **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

double **missing** [INPUT, MANDATORY, default=no default value]

When missing is given, the value outside the boundary is set to missing.

Bilinear(Boolean grid, Double1d x, Double1d y, double missing)

Arguments

Boolean **grid** [INPUT, MANDATORY, default=no default value]

If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.

Double1d **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

Double1d **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

double **missing** [INPUT, MANDATORY, default=no default value]

When missing is given, the value outside the boundary is set to missing.

Bilinear(Boolean grid, double[] x, double[] y, double missing)

Arguments

Boolean **grid** [INPUT, MANDATORY, default=no default value]

If grid is true and if the x contains m elements and y contains n elements, the result has dimensions [m, n], where the element [i,j] of the result contains the value of Images interpolated at position (xi, yj). If the grid is false, the result is the same as calling Bilinear without the grid argument.

double[] **x** [INPUT, MANDATORY, default=no default value]

The array containing the x "virtual subscripts" of the images for which to interpolate values.

double[] **y** [INPUT, MANDATORY, default=no default value]

The array containing the y "virtual subscripts" of the images for which to interpolate values

double **missing** [INPUT, MANDATORY, default=no default value]

When missing is given, the value outside the boundary is set to missing.

Method

```
ArrayData of(Double2d images)
```

Argument

`Double2d` **`images`** `[INPUT, MANDATORY, default=no default value]`

A two dimension data array.

Return

ArrayData

the interpolated image values at the (xi, yj).

See also

- CubicSplineInterpolator
- NearestNeighborInterpolator
- http://en.wikipedia.org/wiki/Bilinear

History

- `2007-03-05 - first`: version
- `2007-03-24 - revised`: version
- `2008-03-27 - implemented`: SPR4608
- `2009-02-25 - add`: URM SPR-6124
- March 5, 2007 - first version
- March 24, 2008 - revised version
- Include missing argument to set the value where the point is outside the boundary
- images: A two dimensional array, Double2d data
- Missing:
- The value to return for elements outside the bounds of iamges. If this keyword is not
- specified, interpolated positions that fall outside the bounds of the array images -
- that is, elements of the x or y arguments that are either less than zero or
- greater than the largest subscript in the corresponding dimension of images - are set
- equal to the value of the nearest element ofimages.
- Feb. 24, 2009
- Updated documentation

3.30. BinCentres

Full Name:	herschel.ia.numeric.toolbox.basic.BinCentres
Alias:	BinCentres
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import BinCentres

Description

Return the bin centers for a Numeric data histogram base on a

use-supplied bin size.

Examples

Example 1: Plot Histogram over BinCentres for a Double1d

```
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.random import RandomGauss
from herschel.ia.numeric.toolbox.basic import Histogram
from herschel.ia.numeric.toolbox.basic import BinCentres
d = Double1d(200000,10.0)
d[90000:115000] = 20.0
d[99000:110000] = 22.0
d[99900:100100] = 23.0
d[99990:100010] = 24.0
d[100000] = 24,5
rnd = RandomGauss()
for ix in range(200000):
  d[ix] += rnd.calc(0.1)
p = PlotXY (d)
binsize = STDDEV (d) * 2.35
print binsize
hist=Histogram(binsize)
bins=BinCentres(binsize)
p2=PlotXY(bins(d),hist(d))
style=p2.getStyle()
style.setChartType(Style.HISTOGRAM)
p.close ()
p2.close ()
```

Example 2: Plot Histogram over BinCentres for a Byte1d

```
vals = [0, 27, 25, 60, 62, 70, 71, 73, 74, 100, 120, 92, 85, 116]
d = Byte1d (vals)
p = PlotXY (d)
binsize = 5
print binsize
hist=Histogram(binsize)
bins=BinCentres(binsize)
p2=PlotXY(bins(d),hist(d))
style=p2.getStyle()
style.setChartType(Style.HISTOGRAM)
p.close ()
p2.close ()
```

Example 3: Plot Histogram over BinCentres for a Double5d

```
d = Double5d (5,4,3,2,1, 2.0)
d.set (4,3,2,1,0, 10.0)
d.set (3,3,2,1,0, 9.0)
d.set (2,3,2,1,0, 8.0)
```

Example 3: Plot Histogram over BinCentres for a Double5d

```
d.set (1,3,2,1,0, 7.0)
binsize = 1.0
hist = Histogram (binsize)
bins = BinCentres (binsize)
p=PlotXY(bins(d),hist(d))
style=p.getStyle()
style.setChartType(Style.HISTOGRAM)
p.close ()
```

API Summary

Jython Syntax
bin=BinCentres(1) <c>=bin(<x>)

Property
MANDATORY. **x INPUT** [any scalar array, MANDATORY, default=no default value]

API details

Property

MANDATORY. x INPUT [any scalar array, MANDATORY, default=no default value]

See also

- Histogram

3.31. Bool1d

Full Name:	herschel.ia.numeric.Bool1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Bool1d

Description

A rectangular numeric boolean array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.32. Bool2d

Full Name:	herschel.ia.numeric.Bool2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Bool2d

Description

A rectangular numeric boolean array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.33. Bool3d

Full Name:	herschel.ia.numeric.Bool3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Bool3d

Description

A rectangular numeric boolean array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.34. Bool4d

Full Name:	herschel.ia.numeric.Bool4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Bool4d

Description

A rectangular numeric boolean array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.35. Bool5d

Full Name:	herschel.ia.numeric.Bool5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Bool5d

Description

A rectangular numeric boolean array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.36. BoxCarFilter

Full Name:	herschel.ia.numeric.toolbox.filter.BoxCarFilter
Alias:	BoxCarFilter
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.filter import BoxCarFilter

Description

Creates a box car filter, that can be applied to numeric arrays of rank 1.

Example

Example 1: Apply BoxCarFilter on a Int1d

```
x=Int1d([1,3,2,4,6,5])
f=BoxCarFilter(2)
print f(x) # [0.5,2.0,2.5,3.0,5.0,5.5]
```

API Summary

Jython Syntax

```
<f>=BoxCarFilter(<width> [,<center>=true|false]
[,<edge>=Convolution.ZEROES|CIRCULAR|REPEAT])
<x>=<f>(<x>)
```

Properties

```
integer width [INPUT, MANDATORY, default=no default value]
boolean center [INPUT, NOT_MANDATORY, default=false]
Convolution.ZEROES|CIRCULAR|REPEAT edge [INPUT, NOT_MANDATORY,
default=Convolution.ZEROES]
```

API details

Properties

integer width [INPUT, MANDATORY, default=no default value]

It must be an integer value

boolean center [INPUT, NOT_MANDATORY, default=false]

Set center to true or false

Convolution.ZEROES|CIRCULAR|REPEAT edge [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]

Set edge to the following:

- edge=Convolution.ZEROES: Set result to zero at edges.
- edge=Convolution.CIRCULAR: Wrap around at edges (circular convolution).

Convolution.ZEROES|CIRCULAR|REPEAT edge [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]
• edge=Convolution.REPEAT: Repeat the edge values of input array.

See also

- Convolution
- GaussianFilter

3.37. BoxCarSmoothingTask

Full Name:	herschel.ia.toolbox.image.BoxCarSmoothingTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import BoxCarSmoothingTask

Description

A Task to smooth an image using a convolution with a box car.

A Task to smooth an image by convolving it with a box car function.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double width [INPUT, MANDATORY, default=Default value : Integer(3)]`
`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image before smoothing.

`Double width [INPUT, MANDATORY, default=Default value : Integer(3)]`

The width of the box car function.

`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

The image after smoothing.

3.38. Byte1d

Full Name:	herschel.ia.numeric.Byte1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Byte1d

Description

A rectangular numeric byte array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.39. Byte2d

Full Name:	herschel.ia.numeric.Byte2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Byte2d

Description

A rectangular numeric byte array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.40. Byte3d

Full Name:	herschel.ia.numeric.Byte3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Byte3d

Description

A rectangular numeric byte array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.41. Byte4d

Full Name:	herschel.ia.numeric.Byte4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Byte4d

Description

A rectangular numeric byte array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.42. Byte5d

Full Name:	herschel.ia.numeric.Byte5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Byte5d

Description

A rectangular numeric byte array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.43. CachePool

Full Name:	herschel.ia.pal.pool.cache.CachePool
Type:	Java Class -
Import:	from herschel.ia.pal.pool.cache import CachePool

Description

A ProductPool implementation that caches a remote pool. This pool is

useful in situations where performance is important, or that network bandwidth needs to be optimized. The cached data is stored in your local file system under the directory defined by: `${hcss.ia.pal.pool.cache.dir}/pal_cache`. By default: `${HOME}/.hcss/pal_cache` (UNIX), or `C:\Documents and Settings\<user>\.hcss\pal_cache` (Windows).

The identifier for the CachedPool (which you may need eg if you want to access that pool remotely) is the same ID as the pool which is being cached.

Example

Example 1: Create a CachedPool around a DbPool storage=ProductStorage()

```
storage.register(CachedPool(DbPool.getInstance()))
```

3.44. CEIL

Full Name:	herschel.ia.numeric.toolbox.basic.Ceiling
Alias:	CEIL
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Ceiling

Description

Gives the smallest integer greater than or equal to x.

Returns a float array for float input array and double array for any other numeric array type.

Example

Example 1: Apply CEIL on a Float1d

```
x=Float1d([0.1,0.5,0.9])
print CEIL(x) # [1.0,1.0,1.0] *
```

API Summary

Jython Syntax
`<y>=CEIL(<x>)`

Properties
`any array of rank 1 x [INPUT, MANDATORY, default=no default value]`
`float or double array y [OUTPUT, MANDATORY, default=no default value]`

API details

Properties

`any array of rank 1 x [INPUT, MANDATORY, default=no default value]`

The input array can be an array of rank 1

`float or double array y [OUTPUT, MANDATORY, default=no default value]`

Returns a float array for float input array and double array for any other numeric array type.

See also

- FLOOR
- ROUND

3.45. ChannelMapPlotting

Full Name:	herschel.ia.gui.cube.ChannelMapPlotting
Type:	Java Class -
Import:	from herschel.ia.gui.cube import ChannelMapPlotting

Description

ChannelMapPlotting

A class to deal with ChannelMapPlotting.

API Summary

Constructor
ChannelMapPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox) Constructor for ChannelMapPlotting. When the new ChannelMapPlotting is

Methods
int **getUsedContourValues**() Returns the used type of contour values for this ChannelMapPlotting
int **getUsedContourLevelRange**() Returns the used type of contour level range for this
String **getErrorMessage**() Returns a convenient error message for the given input. If no
ArrayList **getContourValues**() Returns a list with all contour values for this ChannelMapPlotting.
ArrayList **getContourColors**() Returns the colors that should be used for the contours of the
int **getNumberOfContourLevels**() Returns the number of contour levels for this ChannelMapPlotting if
double[] **getContourLevels**() Returns the used lower and upper contour value for this
double **getMinimumContourLength**() Returns the minimum length for contours to be drawn on the image.
String **getUsedDistribution**() Returns the type of distribution of the contour levels, that is
ArrayList **getImageFigures**() Returns all ImageFigures used for this ChannelMapPlotting.

API details

Constructor

ChannelMapPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)

component-based, a window for contour plotting is opened; otherwise nothinng will be shown.

Arguments

`boolean` **useAsComponent** `[INPUT, MANDATORY]`

`CubeSpectrumAnalysisToolbox` **toolbox** `[INPUT, MANDATORY]`

Methods

int getUsedContourValues()

(manual or automatic).

Return

int

Returns the used type of contour values for this ChannelMapPlotting (manual or automatic).

int getUsedContourLevelRange()

ChannelMapPlotting (manual or image cut levels).

Return

int

Returns the used type of contour level range for this ChannelMapPlotting (manual or image cut levels).

String getErrorMessage()

errors occur in the input, null is returned.

Return

String

Returns a convenient error message for the given input. If no errors occur in the input, null is returned.

ArrayList getContourValues()

Return

ArrayList

Returs a list with all contour values for this ChannelMapPlotting.

ArrayList getContourColors()

different contour values.

Return

ArrayList

Returns the colors that should be used for the contours of the different contour values.

`int getNumberOfContourLevels()`

the contour values are to be calculated automatically; otherwise -1 is returned.

Return

int

Returns the number of contour levels for this ChannelMapPlotting if the contour values are to be calculated automatically; otherwise -1 is returned.

`double[] getContourLevels()`

ChannelMapPlotting (manual or image cut levels).

Return

double[]

The used lower and upper contour value for this ChannelMapPlotting (manual or image cut levels).

`double getMinimumContourLength()`

Return

double

Returns the minimum length for contours to be drawn on the image.

`String getUsedDistribution()`

used for this ChannelMapPlotting (linear, log or ln).

Return

String

Returns the type of distribution of the contour levels, that is used for this ChannelMapPlotting (linear, log or ln).

`ArrayList getImageFigures()`

Return

ArrayList

Returns all ImageFigures used for this ChannelMapPlotting.

3.46. CircleHistogramExplorer

Full Name:	herschel.ia.gui.image.CircleHistogramExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import CircleHistogramExplorer

Description

An explorer for CircleHistogramProducts.

API Summary

Constructors
`CircleHistogramExplorer()` The constructor of a new CircleHistogramExplorer.
`CircleHistogramExplorer(Object object)` The constructor of a new CircleHistogramExplorer associated

Methods
`String` **`getName()`** Returns the name for this CircleHistogramExplorer.
`String` **`getDescription()`** Returns the description for this CircleHistogramExplorer.
`boolean` **`canHandle(Class className)`** Checks whether this CircleHistogramExplorer can handle objects of the
`setObject(Object object)` Sets the object for this CircleHistogramExplorer to the given object.
`CircleHistogramProduct` **`getObject()`** Returns the object for this CircleHistogramExplorer.
`Class` **`getVariableType()`** Returns the expected variable type for this CircleHistogramExplorer.
`JTable` **`getParameterTable()`** Returns the parameter table for this

API details

Constructors

`CircleHistogramExplorer()`

`CircleHistogramExplorer(Object object)`

with the given object.

Argument

`Object` **`object`** `[INPUT, MANDATORY]`

Methods

String getName()

Return

String

Returns the name for this CircleHistogramExplorer.

String getDescription()

Return

String

Returns the description for this CircleHistogramExplorer.

boolean canHandle(Class className)

given class.

Argument

Class **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this CircleHistogramExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

Object **object** [INPUT, MANDATORY]

CircleHistogramProduct getObject()

Return

CircleHistogramProduct

Returns the object for this CircleHistogramExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this CircleHistogramExplorer.

JTable getParameterTable()

CircleHistogramExplorer.

Return

JTable

Returns the parameter table for this CircleHistogramExplorer

3.47. CircleHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.CircleHistogramPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import CircleHistogramPanel

Description

A panel for the CircleHistogramPanel.

API Summary

Constructor
`CircleHistogramPanel()` The construction of a new CircleHistogramPanel.

Methods
`drawFigure()` Draws the circle on the image associated with this
`updateFigure()` Updates the circle associated with this
`trigger()` Triggers the execution of the task associated
`updateHistogram()` Updates the histogram associated with this

API details

Constructor

`CircleHistogramPanel()`

Methods

`drawFigure()`
CircleHistogramPanel.

`updateFigure()`
CircleHistogramPanel.

`trigger()`
with this CircleHistogramPanel.

`updateHistogram()`
CircleHistogramPanel.

3.48. CircleHistogramProduct

Full Name:	herschel.ia.dataset.image.CircleHistogramProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import CircleHistogramProduct

Description

A class to deal with the results of a circle histogram.

API Summary

Constructor
`CircleHistogramProduct()` The constructor of a new CircleHistogramProduct.

Methods
`setCenter``(double centerX, double centerY)` Sets the center for this CircleHistogramProduct
`setCenter``(double centerX, double centerY, String centerRA, String centerDec)` Sets the center for this CircleHistogramProduct
`setRadius``(double pixels)` Sets the radius for this CircleHistogramProduct
`setRadius``(double pixels, double arcsec)` Sets the radius for this ImageHistogramProduct
`Double1d` **`getCenterPixelCoordinates`**`()` Returns the center for this CircleHistogramProduct in
`String1d` **`getCenterSkyCoordinates`**`()` Returns the center for this CircleHistogramProduct in
`double` **`getRadiusPixels`**`()` Returns the radius for this CircleHistogramProduct in pixels.
`double` **`getRadiusArcsec`**`()` Returns the radius for this CircleHistogramProduct in arcsec.

API details

Constructor

`CircleHistogramProduct()`

Methods

`setCenter(double centerX, double centerY)`
to the given pixel coordinates. **Arguments**

setCenter(double centerX, double centerY)

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

setCenter(double centerX, double centerY, String centerRA, String centerDec)

to the given pixel and sky coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

String **centerRA** [INPUT, MANDATORY]

String **centerDec** [INPUT, MANDATORY]

setRadius(double pixels)

to the given number of pixels.

Argument

double **pixels** [INPUT, MANDATORY]

setRadius(double pixels, double arcsec)

to the given number of pixels and arcsec.

Arguments

double **pixels** [INPUT, MANDATORY]

double **arcsec** [INPUT, MANDATORY]

Double1d getCenterPixelCoordinates()

pixel coordinates.

Return

Double1d

Returns the center for this CircleHistogramProduct in pixel coordinates.

String1d getCenterSkyCoordinates()

sky coordinates.

Return

String1d

Returns the center for this CircleHistogramProduct in sky coordinates.

double getRadiusPixels()

Return

double

Returns the radius for this CircleHistogramProduct in pixels.

double getRadiusArcsec()

Return

`double getRadiusArcsec()`

double

Returns the radius for this CircleHistogramProduct in arcsec.

3.49. CircleHistogramTask

Full Name:	herschel.ia.toolbox.image.CircleHistogramTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import CircleHistogramTask

Description

A Task to make a histogram within a circle.

A Task to make a histogram of a region of interest, which is bounded by a circle.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`
`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`
`Double centerY [INPUT, OPTIONAL, default=Default value : NaN]`
`String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]`
`String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]`
`Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]`
`Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`

The lower cut level.

`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`

The upper cut level.

`Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`

The number of bins.

`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the center of the circle.

Double centerY [INPUT, OPTIONAL, default=Default value : NaN]

The y-pixel-coordinate of the center of the circle.

String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]

The right ascension of the center of the circle.

String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the center of the circle.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The radius of the circle in pixels.

Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The radius of the circle in arcsec.

3.50. CircularIntegrationTask

Full Name:	herschel.ia.toolbox.image.CircularIntegrationTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import CircularIntegrationTask

Description

An abstract Task that deals with the integration of a circular region.

To do the integration of a circular region, this region is divided into : * the central pixel (in which the center of the circular region lies) * the rows and columns between two quadrants

3.51. clamp

Full Name:	herschel.ia.toolbox.image.ClampTask
Alias:	clamp
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ClampTask
Category:	task/image

Description

Clamping images and cubes.

The Clamp task for Images and Cubes. ClampTask is a task which restricts the range of pixel values for a source image by constraining the range of pixels to defined "low" and "high" values. The clamp algorithm sets all the pixels whose value is below "low" to "low" and sets all the pixels whose value is above "high" to "high". The "low" value must be less than or equal to the "high" value.

Example

Example 1: Clamping an image to lower value 11 and higher value 240

```
clamp(image = image, low = 11, high = 240)
```

API Summary

Jython Syntax

```
clamp(image, 11, 240)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
double low [INPUT, OPTIONAL, default=No default value]
double high [INPUT, OPTIONAL, default=No default value]
Image clampedImage [OUTPUT, MANDATORY, default=No default value]
```

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The Image to be clamped.

double low [INPUT, OPTIONAL, default=No default value]

The low clamp value.

double high [INPUT, OPTIONAL, default=No default value]

The high clamp value.

Image clampedImage [OUTPUT, MANDATORY, default=No default value]

The clamped image.

3.52. clear

Full Name:	herschel.ia.task.example.InterruptableTask
Alias:	clear
Type:	Java Task -
Import:	from herschel.ia.task.example import InterruptableTask
Category:	developer

Description

Demonstrate the use of the Ctr-s on top a task

Capture Ctrl-s events and provides a recovery action

Examples

Example 1: run the task

```
interruptable()
```

Example 2: run the task with 1000 iteration

```
interruptable(1000)
```

API Summary

Jython Syntax

```
clear("variableName") <br>
clear("variableA,variableB") <br>
clear(all=True)
```

Property

```
Integer iterations [INPUT, NO, default=10000]
```

Limitations

The current implementation allows deleting of packages and class, no pre-filter is performed

Miscellaneous

No miscellaneous

API details

Property

`Integer iterations [INPUT, NO, default=10000]`

The number of iterations to be performed

See also

- references

History

- `2004-07-13 - NdC`: first release.

3.53. clear

Full Name:	herschel.ia.toolbox.util.ClearTask
Alias:	clear
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import ClearTask
Category:	task

Description

Clear variables from the jython session.

The clear task is used to clear a variable from the jython session and reclaim the allocated memory. The task allows also for clearing every variable with the execption of reserved names. When used as in the following "Clear(all=True)" deletes all parameters made by the user - including those from user set up files

Examples

Example 1: clear a single variable

```
clear("variableName")
```

Example 2: clear a list of variables

```
clear("variableA, variableB, ...")
```

Example 3: clear all variables created by the user - including those from user set up files

```
clear(all=True)
```

API Summary

Jython Syntax

```
clear("variableName")
clear("variableA,variableB")
clear(all=True)
```

Properties

```
String variable [INPUT, No, default=NO default value]
Boolean all [INPUT, NO, default=False]
```

Limitations

The current implementation allows deleting of packages and class, no pre-filter is performed

Miscellaneous

No miscellaneous

API details

Properties

String variable [INPUT, No, default=NO default value]
The name of the variable to erase or a list (comma separated) of variables

Boolean all [INPUT, NO, default=False]
The all option for erasing every variable

History

- `2004-07-13 - NdC`: first release

3.54. Complex1d

Full Name:	herschel.ia.numeric.Complex1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Complex1d

Description

A rectangular numeric Complex array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.55. Complex2d

Full Name:	herschel.ia.numeric.Complex2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Complex2d

Description

A rectangular numeric Complex array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.56. Complex3d

Full Name:	herschel.ia.numeric.Complex3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Complex3d

Description

A rectangular numeric Complex array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.57. Complex4d

Full Name:	herschel.ia.numeric.Complex4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Complex4d

Description

A rectangular numeric Complex array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.58. Complex5d

Full Name:	herschel.ia.numeric.Complex5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Complex5d

Description

A rectangular numeric Complex array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.59. CONCATENATE

Full Name:	herschel.ia.numeric.toolbox.basic.Concatenate
Alias:	CONCATENATE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Concatenate

Description

Concatenates an array of rank > 1 into an array rank 1 of the same type.

The elements are concatenated starting from the highest dimension.

Example

Example 1: Apply ABS on a Int1d

```
x=Double2d([ [1,2,3,4],[5,6,7,8] ])
print CONCATENATE(x) # [1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0]
x=Int3d([ [ [1,2],[3,4] ], [ [5,6],[7,8] ], [ [9,10],[11,12] ] ])
print CONCATENATE(x) # [1,2,3,4,5,6,7,8,9,10,11,12]
```

API Summary

Jython Syntax
<y>=CONCATENATE(<x>)

Properties
any array type **x** [INPUT, MANDATORY, default=no default value]
an array of any type **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array type x [INPUT, MANDATORY, default=no default value]

The input must be an array of rank > 1

an array of any type y [OUTPUT, MANDATORY, default=no default value]

The result is an array of rank 1 .

See also

- RESHAPE

3.60. Condense

Full Name:	herschel.ia.numeric.toolbox.basic.Condense
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Condense

Description

Condense array x along dimension d applying function f using a truncation size.

This function obtains an array where each item is the result of applying a function (ie: SUM, COS, etc.) to the original array to the specified dimension. The 'chunkSize' argument specifies how many items are used for each item of the returned array.

Condense modes (see 'mode' parameter):

-Default mode : Condense array x along dimension d applying function f using a truncation size

-Boxcar mode : Run a boxcar window of a certain size over the data array

Condense works like any ArrayToNumber or ArrayReductor over a dimension (ie: SUM(array,along_dim_d)) but you can split the original array into chunks. You will obtain an array with the number of chunks that you have specified (the number of items of the returned array is truncated if it is required, see example). Nevertheless, if you use 'boxcar' mode, chunks are created based on the current position for each item of the returned array (see 'mode' parameter and the example).

Example

Example 1: Untitled

```
IA>>x=RESHAPE(Int1d.range(4*2),[4,2])
IA>>print x
  [
   [0,1],
   [2,3],
   [4,5],
   [6,7]
  ]
IA>>print Condense(0,1,SUM)(x) #mode = Default
  java.lang.IllegalArgumentException: Chunk need to be bigger then 1 !
IA>>print Condense(0,2,SUM)(x) #mode = Default
  [
   [2,4],
   [10,12]
  ]
  #Get chunks of two items along dimension 0:
  #First chunk: SUM(Int2d([[0,1],[2,3]]),0) = [2,4]
  #Second chunk: SUM(Int2d([[4,5],[6,7]]),0) = [10,12]
IA>>print Condense(0,3,SUM)(x) #mode = Default
  [
   [6,9]
  ]
  #Get chunks of three items along dimension 0:
  #First chunk: SUM(Int2d([[0,1],[2,3],[4,5]]),0) = [6,9]
  #Second chunk: [6,7] skipped
IA>>print Condense(0,4,SUM)(x) #mode = Default
  [
   [12,16]
  ]
  #Get chunks of four items along dimension 0:
  #First chunk: SUM(Int2d([[0,1],[2,3],[4,5],[6,7]]),0) = [12,16]
```

Example 1: Untitled

```
IA>>print Condense(0,2,SUM,"boxcar")(x)
  [
   [2,4],
   [6,8],
   [10,12],
   [6,7]
  ]
  #Get chunks of two items along dimension 0 => one item after current
 position:
  #First   chunk: SUM(Int2d([[0,1],[2,3]]),0) = [2,4]
  #Second  chunk: SUM(Int2d([[2,3],[4,5]]),0) = [6,8]
  #Third   chunk: SUM(Int2d([[4,5],[6,7]]),0) = [10,12]
  #Fourth  chunk: SUM(Int2d([[6,7]]),0) = [10,12]
IA>>print Condense(0,3,SUM,'boxcar')(Int2d([[0,1],[2,3],[4,5],[6,7],[8,9],
[10,11],[12,13]]))
  [
   [2,4],
   [6,9],
   [12,15],
   [18,21],
   [24,27],
   [30,33],
   [22,24]
  ]
  #Get chunks of three items along dimension 0 => one item before current
 position, one item
  #after current position:
  #First   chunk: SUM (Int2d([[0,1],[2,3]]),0)            = [2,4]
  #Second  chunk: SUM (Int2d([[0,1],[2,3],[4,5]]),0)      = [6,9]
  #Third   chunk: SUM (Int2d([[2,3],[4,5],[6,7]]),0)      = [12,15]
  #Fourth  chunk: SUM (Int2d([[4,5],[6,7],[8,9]]),0)      = [18,21]
  #Fifth   chunk: SUM (Int2d([[6,7],[8,9],[10,11]]),0)    = [24,27]
  #Sixth   chunk: SUM (Int2d([[8,9],[10,11],[12,13]]),0)  = [30,33]
  #Seventh chunk: SUM (Int2d([[10,11],[12,13]]),0)        = [22,24]
IA>>print Condense(0,4,SUM,'boxcar')(Int2d([[0,1],[2,3],[4,5],[6,7],[8,9],
[10,11],[12,13]]))
  [
   [6,9],
   [12,16],
   [20,24],
   [28,32],
   [36,40],
   [18,20],
   [22,24]
  ]
  #Get chunks of four items along dimension 0 => one item before current
 position, two items
  #after current position:
  #First   chunk: SUM (Int2d([[0,1],[2,3],[4,5]]),0)           = [6,9]
  #Second  chunk: SUM (Int2d([[0,1],[2,3],[4,5],[6,7]]),0)     = [12,16]
  #Third   chunk: SUM (Int2d([[2,3],[4,5],[6,7],[8,9]]),0)     = [20,24]
  #Fourth  chunk: SUM (Int2d([[4,5],[6,7],[8,9],[10,11]]),0)   = [28,32]
  #Fifth   chunk: SUM (Int2d([[6,7],[8,9],[10,11],[12,13]]),0) = [36,40]
  #Sixth   chunk: SUM (Int2d([[8,9],[10,11]]),0)               = [18,20]
  #Seventh chunk: SUM (Int2d([[10,11],[12,13]]),0)             = [22,24]
</pre>
```

API Summary

Jython Syntax

```
outArray = Condense ( alongDimension, chunkSize, function [,mode])
(inArray)
```

Properties

float or double array **inArray** [INPUT, MANDATORY, default=p]

Properties
integer **alongDimension** [INPUT, MANDATORY, default=p]
int **chunkSize** [INPUT, MANDATORY, default=p]
ArrayReductor|ArrayToNumber **function** [INPUT, MANDATORY, default=p]
OPTIONAL **mode** [INPUT, default = false, default=no default value]
float or double array **outArray** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

float or double array inArray [INPUT, MANDATORY, default=p]

Input array of n dimensions

integer alongDimension [INPUT, MANDATORY, default=p]

Dimension in which the function will be applied (starting on zero)

int chunkSize [INPUT, MANDATORY, default=p]

Size of chunks to process. See boxcar parameter.

ArrayReductor|ArrayToNumber function [INPUT, MANDATORY, default=p]

Function to apply

OPTIONAL mode [INPUT, default = false, default=no default value]

Selection of special mode. Currently "default" or "boxcar"

If this argument is specified, the mode is set to "boxcar" (it does not matter the string you write). If this argument is not specified, the mode is set to "default".

- "default": the array is splitted into arrays of the size specified by 'chunkSize' (along the specified dimension). The remaining items (that cannot fit in a chunk) are ignored. 'chunkSize' 1 is not allowed.
- "boxcar" : chunks are created using a "box" centered in the current item. The process goes item by item along the specified dimension:
  - 'chunkSize' = 1: not allowed.
  - 'chunkSize' < 1:
    - Odd 'chunkSize': chunk is compound of the item in the current position, (chunkSize/2) item(s) before the current position and (chunkSize/2) item(s) after the current position. (Division is integer division)
    - Even 'chunkSize': chunk is compound of the item in the current position, ((chunkSize/2)-1) item(s) before the current position and (chunkSize/2) item(s) after the current position. (Division is integer division)

  Example: 'chunkSize' = 3: (along the specified dimension) one item before current position, the current item and another item before current position. 'chunkSize' = 4: one item before current position, the current item and two items after current position.

OPTIONAL mode [INPUT, default = false, default=no default value]

(See examples.)

float or double array outArray [OUTPUT, MANDATORY, default=no default value]

Return an array where the function 'function' was applied on data chunks 'chunkSize' in dimension 'alongDimension'

The center of the box is always the actual index. in case of an "even" box the value before the theorhetical center (see boxcar parameter)

3.61. ConnectorBox

Full Name:	herschel.ia.dataflow.ConnectorBox
Type:	Java Class -
Import:	from herschel.ia.dataflow import ConnectorBox

Description

Non-process components as processes.

This class allows non-process components to be managed by dataflow as they were processes. The non-process component will have a ConnectorBox as member variable and the developer can call create* methods. The non-process component must implement Connectable interface and in its getProcess() method will return the connector box.

Example

Example 1: how to create a ConnectorBox

```
<pre>
public class MyComponent extends JFrame implement Connectable,
 ProcessInputListener {
  private ConnectorBox _conn_box;
  // implementation of Connectable getProcess method
  public IaProcess getProcess() { return _conn_box; }
  public MyComponent( String name_as_component, String name_as_process ) {
      super(name_as_component);
      _conn_box = new ConnectorBox( name_as_process );
      ProcessInput input = _conn_box.createInput( "input", Product.class );
      input.addProcessInputListener( this );
  }
  // implementation of ProcessInputListener handle method
  public void handle( ProcessInputEvent ev ) {
    // do stuff with the product that comes inside ev.
  }
}
</pre>
```

Limitations

no limitation

See also

- reference

History

- 26-5-2005 jcg: change javadoc format for help support.

3.62. Contour

Full Name:	herschel.ia.dataset.image.Contour
Type:	Java Class -
Import:	from herschel.ia.dataset.image import Contour

Description

A class to deal with Contours.

API Summary

Constructor
Contour(Double2d contourPoints) This constructor for Contour creates a new Contour and sets the

Methods
int **getNbOfContourPoints**() Returns the number of contour points for this Contour.
convert(Wcs wcsOld, Wcs wcsNew) Converts the pixel coordinates of the contour points of this

API details

Constructor

Contour(Double2d contourPoints)

data (contour points) for the new Contour to the given data (contour points).

Argument

Double2d **contourPoints** [INPUT, MANDATORY]

Methods

int getNbOfContourPoints()

Return

int

Returns the number of contour points for this Contour.

convert(Wcs wcsOld, Wcs wcsNew)

Contour from one Wcs to pixel coordinates to pixel coordinates in another Wcs.

Arguments

Wcs **wcsOld** [INPUT, MANDATORY]

Wcs **wcsNew** [INPUT, MANDATORY]

3.63. ContourLevel

Full Name:	herschel.ia.dataset.image.ContourLevel
Type:	Java Class -
Import:	from herschel.ia.dataset.image import ContourLevel

Description

A class to deal with ContourLevels.

API Summary

Constructor
`ContourLevel(double contourValue)`
This constructor creates a new ContourLevel for the given contour

Method
`addContour(Contour contour)`
Adds the given Contour to this ContourLevel.

API details

Constructor

`ContourLevel(double contourValue)`

value.

Argument

`double` **`contourValue`** `[INPUT, MANDATORY]`

Method

`addContour(Contour contour)`

Argument

`Contour` **`contour`** `[INPUT, MANDATORY]`

3.64. ContourPlotting

Full Name:	herschel.ia.toolbox.image.gui.ContourPlotting
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import ContourPlotting

Description

An implementation of contour plotting.

API Summary

Constructors
ContourPlotting(ImageAnalysisToolbox toolbox) The standard constructor for ContourPlotting. This constructor creates a window
ContourPlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox) Constructor for ContourPlotting. When the new ContourPlotting is

Methods
int **getUsedContourValues**() Returns the used type of contour values for this ContourPlotting
int **getUsedContourLevelRange**() Returns the used type of contour level range for this
String **getErrorMessage**() Returns an appropriate error message for the given input. If no
ArrayList **getContourValues**() Returns a list with all contour values for this ContourPlotting.
ArrayList **getContourColors**() Returns the colors that should be used for the contours of the
int **getNumberOfContourLevels**() Returns the number of contour levels for this ContourPlotting if
double[] **getContourLevels**() Returns the used lower and upper contour value for this
double **getMinimumContourLength**() Returns the minimum length for contours to be drawn on the image.
String **getUsedDistribution**() Returns the type of distribution of the contour levels, that is
ArrayList **getImageFigures**() Returns all ImageFigures used for this ContourPlotting.

API details

Constructors

`ContourPlotting(ImageAnalysisToolbox toolbox)`

in which you must give the input parameters, needed to perform contour plotting on the image, analyzed with the given ImageAnalysisToolbox.

Argument

`ImageAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

`ContourPlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox)`

component-based, a window for contour plotting is opened; otherwise nothing will be shown.

Arguments

`boolean` **`useAsComponent`** `[INPUT, MANDATORY]`

`ImageAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

Methods

`int getUsedContourValues()`

(manual or automatic).

Return

int

Returns the used type of contour values for this ContourPlotting (manual or automatic).

`int getUsedContourLevelRange()`

ContourPlotting (manual or image cut levels).

Return

int

Returns the used type of contour level range for this ContourPlotting (manual or image cut levels).

`String getErrorMessage()`

errors occur in the input, null is returned.

Return

String

Returns an appropriate error message for the given input. If no errors occur in the input, null is returned.

`ArrayList getContourValues()`

Return

ArrayList

Returs a list with all contour values for this ContourPlotting.

ArrayList getContourColors()

different contour values.

Return

ArrayList

Returns the colors that should be used for the contours of the different contour values.

int getNumberOfContourLevels()

the contour values are to be calculated automatically; otherwise -1 is returned.

Return

int

Returns the number of contour levels for this ContourPlotting if the contour values are to be calculated automatically; otherwise -1 is returned.

double[] getContourLevels()

ContourPlotting (manual or image cut levels).

Return

double[]

The used lower and upper contour value for this ContourPlotting (manual or image cut levels).

double getMinimumContourLength()

Return

double

Returns the minimum length for contours to be drawn on the image.

String getUsedDistribution()

used for this ContourPlotting (linear, log or ln).

Return

String

Returns the type of distribution of the contour levels, that is used for this ContourPlotting (linear, log or ln).

ArrayList getImageFigures()

Return

ArrayList

Returns all ImageFigures used for this ContourPlotting.

3.65. ContourTask

Full Name:	herschel.ia.toolbox.image.ContourTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ContourTask

Description

A Task for making contours for a given contour value.

A Task for making contours for a given contour value and that is called withing AutomaticContourTask and ManualContourTask.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double value [INPUT, MANDATORY, default=No default value]`
`ImageContour contours [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double value [INPUT, MANDATORY, default=No default value]`

The contour value.

`ImageContour contours [OUTPUT, MANDATORY, default=No default value]`

The image contour.

3.66. Convolution

Full Name:	herschel.ia.numeric.toolbox.filter.Convolution
Alias:	Convolution
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.filter import Convolution

Description

Creates a Convolution filter, that can be applied to numeric arrays of rank 1.

Example

Example 1: Apply Convolution on a Int1d

```
x=Int1d([1,3,2,4,6,5])
f=Convolution( Double1d([1,3,2]) )
print f(x) # [1.0,6.0,13.0,16.0,22.0,31.0]
```

API Summary

Jython Syntax

```
<f>=Convolution(<kernel> [,<center>=true|false]
[,<edge>=Convolution.ZEROES|CIRCULAR|REPEAT])
<x>=<f>(<x>)
```

Properties

```
Double1d kernel [INPUT, MANDATORY, default=no default value]
boolean center [INPUT, NOT_MANDATORY, default=false]
Convolution.ZEROES|CIRCULAR|REPEAT center [INPUT, NOT_MANDATORY,
default=Convolution.ZEROES]
```

API details

Properties

Double1d kernel [INPUT, MANDATORY, default=no default value]

It must be a Double1d array, when =true, the kernel should have an odd number of elements.

boolean center [INPUT, NOT_MANDATORY, default=false]

Set center to true or false.

Convolution.ZEROES|CIRCULAR|REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]

Set edge to true or false

- edge=Convolution.ZEROES: Set result to zero at edges.
- edge=Convolution.CIRCULAR: Wrap around at edges (circular convolution).

Convolution.ZEROES|CIRCULAR|REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]
• edge=Convolution.REPEAT: Repeat the edge values of input array.

See also

- BoxCarFilter
- GaussianFilter

History

- 23 Apr 2008 AS Modify default for center parameter; default is center=True.

3.67. Correlate

Full Name:	herschel.ia.numeric.toolbox.basic.Correlate
Alias:	Correlate
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Correlate

Description

Yields the linear Pearson correlation coefficient of the input arrays.

If one vector is longer than the other, only the values up to the length of the shortest vector will be taken into account.

Example

Example 1: Correlation of two vectors of long

```
x = Long1d([65, 63, 67, 64, 68, 62, 70, 66, 68, 67, 69, 71])
y = Long1d([68, 66, 68, 65, 69, 66, 68, 65, 71, 67, 68, 70])
print Correlate(x)(y) # 0.7026516450800809
```

API Summary

Jython Syntax

```
<r>=Correlate(<x>)(<y>)
```

Properties

```
any ordered 1d array x [INPUT, MANDATORY, default=no default value]
any ordered 1d array y [INPUT, MANDATORY, default=no default value]
double r [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any ordered 1d array x [INPUT, MANDATORY, default=no default value]

May be an integral or floating-point array; the former implicitly transformed to a double array.

any ordered 1d array y [INPUT, MANDATORY, default=no default value]

May be an integral or floating-point array; the former implicitly transformed to a double array.

double r [OUTPUT, MANDATORY, default=no default value]

Returns the linear Pearson correlation coefficient of the input arrays.

See also

- CorrelateMatrix

3.68. CorrelateMatrix

Full Name:	herschel.ia.numeric.toolbox.basic.CorrelateMatrix
Alias:	CorrelateMatrix
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import CorrelateMatrix

Description

Yields the linear Pearson correlation coefficients of the input M x N matrix.

The result is a N x N matrix with each i, j value equal to the correlation coefficient of the i and j columns of the original matrix.

Example

Example 1: Apply Correlation of a matrix of integers

```
m = Int2d([ [65, 67], [64, 68], [67, 71] ])
print CorrelateMatrix()(m)
# [ [1.0, 0.8386278693775344], [0.8386278693775344, 1.0] ]
```

API Summary

Jython Syntax

```
<r>=CorrelateMatrix()(<m>)
```

Properties

```
any ordered 2d array (M rows x N columns) m [INPUT, MANDATORY, default=no default value]
Double2d r [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

`any ordered 2d array (M rows x N columns) m [INPUT, MANDATORY, default=no default value]`

May be an integral or floating-point array; the former implicitly transformed to a double array.

`Double2d r [OUTPUT, MANDATORY, default=no default value]`

Returns the N x N matrix with the linear Pearson correlation coefficients of the input matrix.

See also

- Correlate

3.69. COS

Full Name:	herschel.ia.numeric.toolbox.basic.Cos
Alias:	COS
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Cos

Description

Computes the trigonometric cosine of an number or array

Gives the cosine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

Example

Example 1: Apply COS on a Float1d

```
x=Float1d([0,0.5])
print  COS(x) # [1.0,0.87758255]
```

API Summary

Jython Syntax
`<y>=COS(<x>)`

Properties
`any type x [INPUT, MANDATORY, default=no default value]`
`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

API details

Properties

`any type x [INPUT, MANDATORY, default=no default value]`

The input is in radians, and may be of any type.

`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- ARCCOS
- COSH
- SIN
- TAN

3.70. COSH

Full Name:	herschel.ia.numeric.toolbox.basic.CosH
Alias:	COSH
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import CosH

Description

Computes the trigonometric hyperbolic cosine of an number or array

Gives the hyperbolic cosine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

API Summary

Jython Syntax
`<y>=COSH(<x>)`

Properties
`any type x [INPUT, MANDATORY, default=no default value]`
`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

Miscellaneous

Does not work for complex values.

API details

Properties

`any type x [INPUT, MANDATORY, default=no default value]`

The input is in radians, and may be of any type.

`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- COS
- ARCCOS

3.71. crop

Full Name:	herschel.ia.toolbox.image.CropTask
Alias:	crop
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import CropTask
Category:	task/image

Description

A Task to crop images and cubes.

A Task to crop images and cubes to a specified rectangular area. The upper left pixel and the lower right pixel should be given.

Example

Example 1: Cropping an image between x-coordinate 11 and 240, and between y-coordinates 240 and 300

```
crop(image = image, x1 = 11, x2 = 55, y1 = 240, y2 = 300)
```

API Summary

Jython Syntax

```
crop(image, 11, 240, 55, 300)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
int row1 [INPUT, MANDATORY, default=No default value]
int column1 [INPUT, MANDATORY, default=No default value]
int row2 [INPUT, OPTIONAL, default=No default value]
int column2 [INPUT, MANDATORY, default=No default value]
```

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The Image to be cropped

int row1 [INPUT, MANDATORY, default=No default value]

The x coordinate of the left upper pixel of the rectangle to crop

int column1 [INPUT, MANDATORY, default=No default value]

The y coordinate of the left upper pixel of the rectangle to crop

int row2 [INPUT, OPTIONAL, default=No default value]

The x coordinate of the lower right pixel of the rectangle to crop

int column2 [INPUT, MANDATORY, default=No default value]
The y coordinate of the lower right pixel of the rectangle to crop

3.72. CubeSpectrumAnalysis

Full Name:	herschel.ia.gui.cube.CubeSpectrumAnalysis
Type:	Java Class -
Import:	from herschel.ia.gui.cube import CubeSpectrumAnalysis

Description

CubeSpectrumAnalysis

A class for dealing with CubeSpectrumAnalysis (in the current version : SpectrumPlotting, Averaged Spectrum Plotting, Velocitiy map Channel Map).

API Summary

Constructor
CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title) Standard constructor for CubeSpectrumAnalysis. A new window with the

Methods
JFrame **getFrame**() Method that returns the JFrame of this CubeSpectrumAnalysis.
CubeSpectrumAnalysisToolbox **getToolbox**() Returns the CubeSpectrumAnalysisToolbox, associated with this
Container **getComponent**() Returns the content pane (Container) of the JFrame of this
ArrayList **getImageFigures**() Returns all ImageFigures used for this CubeSpectrumAnalysis.

API details

Constructor

CubeSpectrumAnalysis(CubeSpectrumAnalysisToolbox toolbox, String title)

given title and associated with the given CubeSpectrumAnalysisToolbox is crested.

Arguments

CubeSpectrumAnalysisToolbox **toolbox** [INPUT, MANDATORY]

String **title** [INPUT, MANDATORY]

Methods

JFrame getFrame()

Return

JFrame

JFrame getFrame()

The JFrame of this CubeSpectrumAnalysis.

CubeSpectrumAnalysisToolbox getToolbox()

CubeSpectrumAnalysis.

Return

CubeSpectrumAnalysisToolbox

Returns the CubeSpectrumAnalysisToolbox, associated with this CubeSpectrumAnalysis.

Container getComponent()

CubeSpectrumAnalysis.

Return

Container

Returns the content pane (Container) of the JFrame of this CubeSpectrumAnalysis.

ArrayList getImageFigures()

Return

ArrayList

Returns all ImageFigures used for this CubeSpectrumAnalysis.

3.73. CubeSpectrumAnalysisToolbox

Full Name:	herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox
Type:	Java Class -
Import:	from herschel.ia.gui.cube import CubeSpectrumAnalysisToolbox

Description

An implementation of a toolbox for spectrum analysis.

In the current version do the following :

1) You can extract a raw single pixel spectrum from a position selected when clicking on the mouse in the image.

2) You can extract an averaged spectrum on a region defined the image by drawing a shape with the mouse. - The circle selection use a linear approximation to compute the weight of the border pixels - The circle selection is "free floating"

5) You can construct a velocity position map, by drawing a line on the image. the velocities are computed from the different layers along the Z axis

6) From 15-09-08 this cube Spectrum Analysis toolbox is registered as a Viewer for the SimpleCube. this mean that in HIPE a simpleCube Variable can be automatically visualized with it . This toolbox is registefred in HIPE on the SimpleCube and therefore can be launch via the right button menu In all cases the spectrum analysis is performed on the cube with the shown layer considered as the reference frame. The result is shown in a multitab section on the righthandside. All the features are using Task which can be launched from e JIDE environment.

A basic example on how to open a toolbox for spectrum analysis on an SimpleCube Cds or a couple of arrays (Double3d cube,Double1d wavelength)

or

A basic example on how to open a toolbox for spectrum analysis on an SimpleCube Cds or a couple of arrays (Double3d cube,Double1d wavelength)

cat = CubeSpectrumAnalysisToolbox(Cds)

or

cat = CubeSpectrumAnalysisToolbox(Cube,Wavearray)

cat = CubeSpectrumAnalysisToolbox()

cat.setCube(simpleCube)

or

cat = CubeSpectrumAnalysisToolbox()

cat.setCube(SpectralSimpleCube)

or

cat = CubeSpectrumAnalysisToolbox()

```
cat.setCube(Cube,VaweArray)

or

cat = CubeSpectrumAnalysisToolbox()

cat.setCube(SimpleCube)

to create a simplecube and launch the CubeAnalysistoolbox do this:

from herschel.ia.dataset import *

from herschel.ia.numeric import *

from herschel.ia.dataset.image import *

from herschel.ia.dataset.image.wcs import *

import herschel.ia.gui.plot.PlotXY

from herschel.share.unit.Length import *

Waves =Double1d.range(900)/900.*4.E-6

print Waves

a= Double1d(900)

b= RandomUniform()

cube = Double3d(900,5,6)

for raw in range(5):

for column in range(6):

shift1=int( 100.*b.calc(1.))

for wave in range(900):

if raw==2:

cube[wave,raw,column]=3 + EXP((-1.)*(wave -(350.+shift1))*(wave - (350.+shift1))/
((50.*50.)))*100.+b.calc(1.)

elif raw==3:

cube[wave,raw,column]=3 + EXP((-1.)*(wave - (350.+shift1))*(wave - (350.+shift1))/
((50.*50.)))*100.+b.calc(1.)

else :

cube[wave,raw,column]=0

myWcs = Wcs()

simplecube2 = SimpleCube()

simplecube2.setCube(cube)
```

myWcs.setNAxis(3)

myWcs.setCrval1(0.0)

myWcs.setCrval2(0.0)

myWcs.setCrpix1(0)

myWcs.setCrpix2(0)

myWcs.setCdelt1(0.1)

myWcs.setCdelt2(0.1)

myWcs.setCtype1("RA---TAN")

myWcs.setCtype2("DEC--TAN")

myWcs.setCtype3("Wavelength")

myWcs.setCunit1("[4.84813681109536E-6 rad]")

myWcs.setCunit2("[4.84813681109536E-6 rad]")

myWcs.setCunit3("[MICROMETERS]")

myWcs.setEpoch(2000)

myWcs.setImageIndex(Waves,MICROMETERS)

simplecube2.setWcs(myWcs)

import herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox

initialization of the CubeSpectrumAnalysisToolbox

cat = CubeSpectrumAnalysisToolbox()

cat.setCube(simplecube2)

Example

Example 1: A basic example on how to open a toolbox for spectrum analysis on an

```
SimpleCube Cds or a couple of arrays (Double3d cube,Double1d wavelength)
cat = CubeSpectrumAnalysisToolbox(Double3d)<br>
or<br>
cat = CubeSpectrumAnalysisToolbox(SimpleCube)<br>
or<br>
cat = CubeSpectrumAnalysisToolbox(Cube,Wavearray)<br>
cat = CubeSpectrumAnalysisToolbox()<br>
cat.setCube(Cds)<br>
or <br>
cat = CubeSpectrumAnalysisToolbox() <br>
cat.setCube(Cube,VaweArray)<br>
or <br>
<br>
cat = CubeSpectrumAnalysisToolbox() <br>
cat.setCube(SimpleCube)<br>
to create a simplecube and launch the CubeAnalysistoolbox do this:
```

Example 1: A basic example on how to open a toolbox for spectrum analysis on an

```
from herschel.ia.dataset import *
from herschel.ia.numeric import *
from herschel.ia.dataset.image import *
from herschel.ia.dataset.image.wcs import *
import herschel.ia.gui.plot.PlotXY
from herschel.share.unit.Length import *
Waves =Double1d.range(900)/900.*4.E-6
print Waves
a= Double1d(900)
b= RandomUniform()
cube = Double3d(900,5,6)
for raw in range(5):
for column in range(6):
shift1=int( 100.*b.calc(1.))
for wave in range(900):
if raw==2:
  cube[wave,raw,column]=3 + EXP((-1.)*(wave -(350.+shift1))*(wave - (350.
+shift1))/((50.*50.)))*100.+b.calc(1.)
elif raw==3:
  cube[wave,raw,column]=3 + EXP((-1.)*(wave - (350.+shift1))*(wave - (350.
+shift1))/((50.*50.)))*100.+b.calc(1.)
else :
  cube[wave,raw,column]=0
myWcs = Wcs()
simplecube2 = SimpleCube()
simplecube2.setCube(cube)
myWcs.setNAxis(3)
myWcs.setCrval1(0.0)
myWcs.setCrval2(0.0)
myWcs.setCrpix1(0)
myWcs.setCrpix2(0)
myWcs.setCdelt1(0.1)
myWcs.setCdelt2(0.1)
myWcs.setCtype1("RA---TAN")
myWcs.setCtype2("DEC--TAN")
myWcs.setCtype3("Wavelength")
myWcs.setCunit1("[4.84813681109536E-6 rad]")
myWcs.setCunit2("[4.84813681109536E-6 rad]")
myWcs.setCunit3("[MICROMETERS]")
myWcs.setEpoch(2000)
myWcs.setImageIndex(Waves,MICROMETERS)
simplecube2.setWcs(myWcs)
import herschel.ia.gui.cube.CubeSpectrumAnalysisToolbox
# initialization of the CubeSpectrumAnalysisToolbox
cat =  CubeSpectrumAnalysisToolbox()
cat.setCube(simplecube2)
```

API Summary

Constructors
CubeSpectrumAnalysisToolbox()
The standard constructor for CubeSpectrumAnalysisToolbox. This
CubeSpectrumAnalysisToolbox(Array3dData array3d)
Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d)
Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d, String specDim, String specUnit)
Constructor for CubeSpectrumAnalysisToolbox. This constructor shows an
CubeSpectrumAnalysisToolbox(type simpleCube)

Constructors
Constructor for CubeSpectrumAnalysisToolbox. This constructor shows a

Methods
SaveCurrentImage() Makes a menubar (Image, Help) for the Jpanel version this CubeSpectrumAnalysisToolbox.
setCube(Array3dData array3d) Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,
setCube(SimpleCube simplecube) Sets the Cube you wish to analyse with this CubeSpectrumAnalysisToolbox,
setCube(SpectralSimpleCube spectralsimplecube) Sets the Cube you wish to analyse with this CubeSpectrumAnalysisToolbox,
setCube(Array3dData array3d, Double1d wavearray) Sets the image you wish to analyse with this CubeSpectrumAnalysisToolbox,
addCube(Array3dData array3d) Adds the given image (Array3dData) to the images, analysed with
exit() Exits this CubeSpectrumAnalysisToolbox (closes (the window of) this
closeActiveTab() Closes the active tab, if there is one.
closeAllTabs() Closes all tabs, if there are any.
setEnabled(boolean enabled) Disables/enables all tabs on the JTabbedPane ans the menus for
boolean **isEnabled**() Returns whether the JTabbedPane and menus for closing tabs and
Double1d **getWaves**() Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
boolean **isInHipe**() Method that returns the way the CubeSpectrumAnalysisToolbox was openned.
SimpleCube **getSimpleCube**() Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
SpectralSimpleCube **getSpectralSimpleCube**() Method that returns the Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
String **getSpecDim**() Method that return The string indicating the physical dimension of the spectrum
String **getSpecUnit**() Method that return The string indicating the unit of the spectrum
JFrame **getFrame**() Method that returns the JFrame of this CubeSpectrumAnalysisToolbox.
JPanel **getPanel**()

Methods
Method that returns the JPanel of this CubeSpectrumAnalysisToolbox.
`JTabbedPane` **`getTabbedPane()`** Method that returns the JTabbedPane of this CubeSpectrumAnalysisToolbox.
`SimpleImage` **`getSimpleImage()`** Returns the displayed image (SimpleImage).
`PlotXY` **`getRtSpectrum()`** Returns the displayed image (ImageDataset).
`Display` **`getDisplay()`** Returns the Display of the ImageDataset you are analysing with
`int` **`getSelectedLayer()`** Returns the index of the shown layer, or the index of the selected
`Double` **`getMaxValue()`** Returns the maximal value of the "image" of the cube
`Double` **`getMinValue()`** Returns the maximal value of the "image" of the cube
`int` **`getSelectedTab()`** Returns the index of the tab, selected for the shown layer.
`JTabbedPane` **`getTabOnPane()`** Returns the tab (JTabbedPane) on the JTabbedPane, that is
`VelocityPosMapPlotting` **`getVelocityPosMap()`** Returns the tab (JTabbedPane) on the JTabbedPane, that is
`SpectrumPlotting` **`getSpecPlot()`** Returns the SpectrumPlotting to acces to all the values of
`SpectrumAvgPlotting` **`getAvgSpecPlot()`** Returns the SpectrumAvgPlotting jframe to acces to all the values of
`ChannelMapPlotting` **`getChannelMapPlot()`** Returns the ChannelMapPlotting jframe to acces to all the values of
`IntegratedMapDisplay` **`getIntegratedmpaPlot()`** Returns the IntegratedMapDisplay jframe to acces to all the values of
`SimpleCube` **`getExtractedCube()`** Returns the range extraction gui jframe to acces to all the values of
`Spectrum1d` **`getSinglePixelSpectrum()`** Returns the single pixel spectrum as Spectrum1d
`Spectrum1d` **`getAvgspectrum()`** that returns the Averaged spectrum from the region spectrum extarction task
`SimpleCube` **`getRangeExtractedCube()`** returns the sub-range cube from the range extraction range feature
`SimpleImage` **`getVelocityAxisImage()`** returns the velocity position map for the "map mode" of the velocityposition map feature
`SimpleCube` **`getVelocityMapCube()`** returns the velocity position map for the "map mode" of the velocityposition map feature

Methods
`ArrayList` **`getIntegratedMapImages`**`()` Returns the single pixel spectrum as Spectrum1d
`setWaves``(Array3dData cube3d)` Method that initialize the Double1d array of the wavelength of
`setWaves``(Double1d waves1d)` Method that initialize the Double1d array of the wavelength of this CubeSpectrumAnalysisToolbox.
`boolean` **`isInImage`**`(MouseEvent e)` Checks whether the given MouseEvent has occured within the image
`boolean` **`isInImage`**`(double pixX, double pixY)` Checks whether the point with given PixelCoordinates (pixX, pixY)
`boolean` **`hasWCS`**`()` Returns true if SkyCoordinates are available for the image,
`showSpectrum``()` Draws a staight line on the image, starting at the point in the
`showSpectrumAvg``()` Drawn a region on the image, depending of the option selected
`showVelocityPosMap``()` Draws a staight line on the image, starting at the point in the
`showChannelMap``()` Draws a staight line on the image, starting at the point in the
`showIntegratedMap``()` Draws a staight line on the image, starting at the point in the
`guiRangeExtract``()` Draws a staight line on the image, starting at the point in the
`ArrayList` **`getAllFigures`**`()` Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.
`addFigures``(ArrayList figures)` Adds the ImageFigures belonging to the selected tab up front to
`removeLastFigure``(type figures The ImageFigures to add to the list of ImageFigures)` Adds the ImageFigures belonging to the selected tab up front to
`updateImage``()` Updates the image, when another tab is made active.
`removeAllAnnotations``()` Makes all annotations invisible.
`createSpaceOnTabbedPane``()` Creates a free space at the end of the Arraylist of ArrayLists of
`createSpaceInTab``()` Creates a free space at index 0 in the ArrayList of CanvasFigures,
`RangeExtractionGui` **`getRangeextraction`**`()`

API details

Constructors

CubeSpectrumAnalysisToolbox()

constructor shows an empty CubeSpectrumAnalysisToolbox window in 2 parts, with only a title, buttons for maximizing, minimizing and closing the window, a menubar (File, Image, Help) and a colorbar and a statusbar at the bottom.

CubeSpectrumAnalysisToolbox(Array3dData array3d)

empty CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximising, minimising and closing the window, and a menubar (file, Spectrum, Help). Also the image you wish to analyse, is displayed.

Argument

`Array3dData` **array3d** [INPUT, MANDATORY]

CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d)

empty CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, and a menubar (file, Spectrum, Help). Also the image you wish to analyse, is displayed. THE DEFAULT UNIT is MICROMETER

Arguments

`Array3dData` **array3d** [INPUT, MANDATORY]

`Double1d` **wave1d** [INPUT, MANDATORY]

CubeSpectrumAnalysisToolbox(Array3dData array3d, Double1d wave1d, String specDim, String specUnit)

empty CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximising, minimising and closing the window, and a menubar (file, Spectrum, Help). Also the image you wish to analyse, is displayed.

Arguments

`Array3dData` **array3d** [INPUT, MANDATORY]

`Double1d` **wave1d** [INPUT, MANDATORY]

`String` **specDim** [INPUT, MANDATORY]

`String` **specUnit** [INPUT, MANDATORY]

CubeSpectrumAnalysisToolbox(type simpleCube)

CubeSpectrumAnalysisToolbox window in 2 parts, with a title, buttons for maximising, minimising and closing the window, and a menubar (file, Spectrum, Help). Also the "cube" of the Simplecube you wish to analyse, is displayed.

Argument

type **simpleCube** [INPUT, MANDATORY, default=no default value]

The SimpleCube containing all the information of an observation you wish to analyse and display with this CubeSpectrumAnalysisToolbox.

Methods

SaveCurrentImage()

setCube(Array3dData array3d)

to the given ImageDataset.

Argument

Array3dData **array3d** [INPUT, MANDATORY]

setCube(SimpleCube simplecube)

to the given SimpleCube.

Argument

SimpleCube **simplecube** [INPUT, MANDATORY]

setCube(SpectralSimpleCube spectralsimplecube)

to the given spectralSimpleCube.

Argument

SpectralSimpleCube **spectralsimplecube** [INPUT, MANDATORY]

setCube(Array3dData array3d, Double1d wavearray)

to the given ImageDataset.

Arguments

Array3dData **array3d** [INPUT, MANDATORY]

Double1d **wavearray** [INPUT, MANDATORY]

addCube(Array3dData array3d)

this CubeSpectrumAnalysisToolbox.

Argument

Array3dData **array3d** [INPUT, MANDATORY]

exit()

CubeSpectrumAnalysisToolbox).

closeActiveTab()

closeAllTabs()

setEnabled(boolean enabled)

closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox.

Argument

boolean **enabled** [INPUT, MANDATORY]

boolean isEnabled()

performing image analysis, of this CubeSpectrumAnalysisToolbox are enabled/disabled.

Return

boolean

Returns true is the JTabbedPane and menus for closing tabs and performing image analysis, of this CubeSpectrumAnalysisToolbox are enabled; false otherwise.

Double1d getWaves()

Return

Double1d

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

boolean isInHipe()

true if it's open with the right menu in the variable list of hipe false when openned with the classical command line call

Return

boolean

the way the CubeSpectrumAnalysisToolbox was openned. true if it's open with the right menu in the variable list of hipe false when openned with the classical command line call

SimpleCube getSimpleCube()

Return

SimpleCube

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

SpectralSimpleCube getSpectralSimpleCube()

Return

SpectralSimpleCube

The Float1d array of the wavelength of this CubeSpectrumAnalysisToolbox.

String getSpecDim()

Return

String

The string indicating the physical dimension of the spectrum

String getSpecUnit()

Return

String

The string indicating the unit of the spectrum

JFrame getFrame()

Return

JFrame

The JFrame of this CubeSpectrumAnalysisToolbox.

JPanel getPanel()

Return

JPanel getPanel()

JPanel

The JPanel of this CubeSpectrumAnalysisToolbox.

JTabbedPane getTabbedPane()

Return

JTabbedPane

The JTabbedPane of this CubeSpectrumAnalysisToolbox.

SimpleImage getSimpleImage()

Return

SimpleImage

Returns the displayed image (SimpleImage).

PlotXY getRtSpectrum()

Return

PlotXY

Returns the displayed image (ImageDataset).

Display getDisplay()

this CubeSpectrumAnalysisToolbox.

Return

Display

Returns the display of the ImageDataset you are analysing with this CubeSpectrumAnalysisToolbox.

int getSelectedLayer()

tab on the JTabbedPane.

Return

int

Returns the index of the shown layer,or the index of the selected tab on the JTabbedPane.

Double getMaxValue()

Return

Double

Returns the maximal value of the "image" of the cube

Double getMinValue()

Return

Double

Double getMinValue()

Returns the maximal value of the "image" of the cube

int getSelectedTab()

Return

int

The index of the tab, selected for the shown layer.

JTabbedPane getTabOnPane()

selected.

Return

JTabbedPane

Returns the tab (JTabbedPane) on the JTabbedPane, that is selected.

VelocityPosMapPlotting getVelocityPosMap()

selected.

Return

VelocityPosMapPlotting

Returns the tab (JTabbedPane) on the JTabbedPane, that is selected.

SpectrumPlotting getSpecPlot()

single pixel spectrum extraction.

Return

SpectrumPlotting

Returns the SpectrumPlotting to acces to all the values of single pixel spectrum extraction.

SpectrumAvgPlotting getAvgSpecPlot()

single pixel spectrum extraction.

Return

SpectrumAvgPlotting

Returns the SpectrumAvgPlotting to acces to all the values of single pixel spectrum extraction.

ChannelMapPlotting getChannelMapPlot()

channel map exraction

Return

ChannelMapPlotting

Returns the ChannelMapPlotting jframe to acces to all the values of channel map exraction

IntegratedMapDisplay getIntegratedmpaPlot()

channel map exraction

Return

IntegratedMapDisplay getIntegratedmpaPlot()

IntegratedMapDisplay

Returns the IntegratedMapDisplay jframe to acces to all the values of channel map exraction

SimpleCube getExtractedCube()

channel map exraction

Return

SimpleCube

Returns the range extraction gui jframe to acces to all the values of channel map exraction

Spectrum1d getSinglePixelSpectrum()

Return

Spectrum1d

Returns the single pixel spectrum as Spectrum1D

Spectrum1d getAvgspectrum()

Return

Spectrum1d

that returns the Averaged spectrum from the region spectrum extarction task

SimpleCube getRangeExtractedCube()

Return

SimpleCube

returns the sub-range cube from the range extraction range feature

SimpleImage getVelocityAxisImage()

Return

SimpleImage

returns the velocity position map for the "map mode" of the velocityposition map feature

SimpleCube getVelocityMapCube()

Return

SimpleCube

returns the velocity position map for the "map mode" of the velocityposition map feature

ArrayList getIntegratedMapImages()

Return

ArrayList

Returns the single pixel spectrum as Spectrum1D

setWaves(Array3dData cube3d)

this CubeSpectrumAnalysisToolbox.

Argument

`Array3dData` **cube3d** [INPUT, MANDATORY]

setWaves(Double1d waves1d)

Argument

`Double1d` **waves1d** [INPUT, MANDATORY]

boolean isInImage(MouseEvent e)

we are analysing with this CubeSpectrumAnalysisToolbox.

Argument

`MouseEvent` **e** [INPUT, MANDATORY]

Return

boolean

Returns true if the given MouseEvent has occured within the image that is being analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

boolean isInImage(double pixX, double pixY)

lies in the image we are analysing with this CubeSpectrumAnalysisToolbox.

Arguments

`double` **pixX** [INPUT, MANDATORY]

`double` **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the point with given PixelCoordinates (pixX, pixY) lies in the image we are analysing with this CubeSpectrumAnalysisToolbox; false otherwise.

boolean hasWCS()

analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

Return

boolean

Returns true if SkyCoordinates are available for the image, analysed with this CubeSpectrumAnalysisToolbox; false otherwise.

showSpectrum()

you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.

showSpectrumAvg()

on the right side of the window and of the point clicked in the image with the mouse. and plots the averaged spectrum along that "cylinder" .When you click or move outside of the image, no shape is defined.

showVelocityPosMap()

you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.

showChannelMap()

you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.

showIntegratedMap()

you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.

guiRangeExtract()

you clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight lin. When you click or move outside of the image, no straight line is drawn and no plot is shown.

ArrayList getAllFigures()

Return

ArrayList

Returns all ImageFigures used for this CubeSpectrumAnalysisToolbox.

addFigures(ArrayList figures)

the list of ImageFigures, used for this CubeSpectrumAnalysisToolbox.

Argument

`ArrayList` **figures** `[INPUT, MANDATORY]`

removeLastFigure(type figures The ImageFigures to add to the list of ImageFigures)

the list of ImageFigures, used for this CubeSpectrumAnalysisToolbox.

Argument

`type` **figures The ImageFigures to add to the list of ImageFigures** `[INPUT, MANDATORY, default=no default value]`

used for this CubeSpectrumAnalysisToolbox.

updateImage()

removeAllAnnotations()

createSpaceOnTabbedPane()

ArrayLists of ImageFigures, that belong to the image analysis on the shown layer.

createSpaceInTab()

belonging to a new image analysis on the shown layer.

RangeExtractionGui getRangeextraction()
Return **RangeExtractionGui** the RangeExtractionGui

See also

- Display, PlotXY

3.74. CubicSplineInterpolator

Full Name:	herschel.ia.numeric.toolbox.interp.CubicSplineInterpolator
Alias:	CubicSplineInterpolator
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.interp import CubicSplineInterpolator

Description

Creates a cubic spline interpolation function from a set of knots (x,y),

that can be applied to numeric arrays of rank 1.

The second derivative is assumed to be zero at the end points (i.e. a natural spline).

Example

Example 1: Create and apply a CubicSplineInterpolator on a Double1d

```
# create some knots
x=Double1d.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
f=CubicSplineInterpolator(x,SQUARE(x))
u=Float1d([1,1.1,2,2.6,3,3.9])
print f(u) # [1.0,1.1970944,4.0,6.7656145,9.0,15.209593]
```

API Summary

Jython Syntax

```
<f>=CubicSplineInterpolator(<x>,<y>[,allowExtrapolation])
```

Properties

```
Double1d x [INPUT, MANDATORY, default=no default value]
Double1d y [INPUT, NOT_MANDATORY, default=false]
boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]
```

API details

Properties

`Double1d x [INPUT, MANDATORY, default=no default value]`

The knots are Double1d

`Double1d y [INPUT, NOT_MANDATORY, default=false]`

The knots are Double1d

`boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]`

Extrapolation is not allowed by default. Boolean.TRUE allows extrapolation.

See also

- LinearInterpolator

- NearestNeighborInterpolator

3.75. cutLevels

Full Name:	herschel.ia.toolbox.image.CutLevelsTask
Alias:	cutLevels
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import CutLevelsTask
Category:	task/image

Description

A Task to calculate the cut levels of an image.

CutLevelsTask is a task which returns the cut levels of an Image. The cut levels can be calculated using 2 methods :

- PERCENT : A certain percentage of the pixels is calculated.
- MEDIAN_FILTER : A window of 7 pixels is taken. From this window, the median is taken. This median is then compared with the minimum and maximum value of the image. 21 pixels are skipped in the column dimension and 3 rows are skipped.

Examples

Example 1: Calculate the cut levels where 95% of the values are included.

```
cutLevelsTask(image = im, method=CutLevels.PERCENT, percent = 95.0)
```

Example 2: Calculate the cut levels using the median filter

```
cutLevelsTask(image = im, method=CutLevels.MEDIAN_FILTER)
```

API Summary

Jython Syntax

```
cutLevels(image, CutLevels.PERCENT, 95.0)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
int method [INPUT, MANDATORY, default=CutLevels.MEDIAN_FILTER]
double percent [INPUT, OPTIONAL, default=99.5]
double[] cutLevels [OUTPUT, MANDATORY, default=No default value]
```

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The image to use for calculating the cut levels.

int method [INPUT, MANDATORY, default=CutLevels.MEDIAN_FILTER]

The method to use for calculating the cut levels (CutLevels.PERCENT or CutLevels.MEDIAN_FILTER).

double percent [INPUT, OPTIONAL, default=99.5]

The percentage of pixels to use in the calculation of the cut levels.

double[] cutLevels [OUTPUT, MANDATORY, default=No default value]

The minimum and maximum cut level.

3.76. DataFlow

Full Name:	herschel.ia.dataflow.DataFlow
Type:	Java Class -
Import:	from herschel.ia.dataflow import DataFlow

Description

Groups several processes as one big process.

It is a process that may manage processes. Processes are created within the dataflow, and may be connected among them. DataFlow allows created processes outside to be added to it. A developer may inherit from this class in order to create a specific dataflow.

Example

Example 1: how to use a dataflow with given processes.

```
<pre>
from herschel.ia.dataflow import *
#creating dataflow and processes.
df = DataFlow("MyDataFlow")
df.createProcess("generator", "myprocesses.ProcessGenerator")
df.createProcess("fft", "myprocesses.ProcessFFT")
# adding a created process
p = myprocess.ProcessFFT("viewer")
df.addProcess(p)
#connecting processes
df.connect("generator.output", "fft.input")
df.connect("fft.output", "viewer.input")
# putting the viewer in a JFrame.
from javax.swing import *
frame = JFrame("MyFrame")
viewer = df.getProcess("viewer")
frame.getContentPane().add(viewer.getJComponent())
frame.setSize(600,600)
frame.setVisible(1)
# starting the dataflow.
df.start()
</pre>
```

Limitations

no limitation

See also

- reference

History

- 26-5-2005 jcg: change javadoc format for help support.

3.77. DataFlowManager

Full Name:	herschel.ia.dataflow.DataFlowManager
Type:	Java Class -
Import:	from herschel.ia.dataflow import DataFlowManager

Description

Shows dataflows.

Allows to the user to see the dataflow in a graphical environment.

Example

Example 1: how to show a dataflow

```
<pre>
from herschel.ia.dataflow import *
df = DataFlow("mydf")
df.createProcess(...)
df.createProcess(...)
df.createProcess(...)
DataFlowManager(df)
</pre>
```

Limitations

no limitation

See also

- reference

History

- 26-5-2005 jcg: change javadoc format for help support.

3.78. DbFactory

Full Name:	herschel.ia.pal.pool.db.DbFactory
Type:	Java Class -
Import:	from herschel.ia.pal.pool.db import DbFactory
Category:	PAL

Description

An implementation of a ProductPool which saves data to a versant database.

You need to define a property which will store the name of the database you wish the DbPool to use. Please check that the property of the same name points to a valid physical database.

Depending on your network performance and proximity to the database pool you may want to wrap a CachedPool around the DbPool. See the last example below.

Examples

Example 1: Create a DbPool that points to the database property "hcss.ia.pal.pool.db.database"

```
storage=ProductStorage();<br>
storage.register(DbFactory.getStore())
```

Example 2: Create a DbPool that points to the database <dbname>@<hostname>

```
storage=ProductStorage() <br>
storage.register(DbFactory.getStore(&lt;dbname&gt;@&lt;hostname&gt;, poolName))
```

Example 3: Saving and Loading to/from a DbPool

```
ref=storage.save(myproduct) <br>
ref=storage.load(ref)
```

Example 4: Wrapping a CachedPool around a DbPool

```
storage.register(CachedPool(DbFactory.getStore()))
```

3.79. DbPool

Full Name:	herschel.ia.pal.pool.db.DbPool
Type:	Java Class -
Import:	from herschel.ia.pal.pool.db import DbPool
Category:	PAL

Description

An implementation of a ProductPool which saves data to a versant database (to be exact, any

object database that uses the HCSS ObjectStore interface).

New: The DbPool continues to support the direct creation of its pools, however it is now recommended that DbPool(s) be created using the DbFactory methods.

You need to define a property which will store the name of the database you wish the DbPool to use. By default this is "hcss.test.database". Please check that the property of the same name points to a valid physical database.

Depending on your network performance and proximity to the database pool you may want to wrap the a CachedPool around the DbPool. See the last example below.

Examples

Example 1: Create a DbPool that points to the default database property "hcss.test.database"

```
storage=ProductStorage() <br>
storage.register(DbPool.getInstance())
```

Example 2: Create a DbPool that points to the database property "my.database"

```
storage=ProductStorage() <br>
storage.register(DbPool.getInstance("my.database", poolName))
```

Example 3: Saving and Loading to/from a DbPool

```
ref=storage.save(myproduct)<br>
ref=storage.load(ref)
```

Example 4: Wrapping a CachedPool around a DbPool

```
storage.register(CachedPool(DbPool.getInstance()))
```

API Summary

Methods

`DbPool` **`getInstance`**`(String databaseProperty, String poolName)`

DbPool

`DbPool` **`getInstance`**`(String databaseProperty)`

DbPool

Methods
`DbPool` **`getInstance()`** Creates an instance of a DbPool connected to the logical database of default property named
`DbPool` **`createInstance`**`(String databaseProperty)` DbPool

API details

Methods

`DbPool getInstance(String databaseProperty, String poolName)`

Creates an instance of a DbPool connected to a logical database

Arguments

`String` **`databaseProperty`** `[INPUT, MANDATORY, default=no default value]`

The logical name of the database, for example "hcss.test.database"

`String` **`poolName`** `[INPUT, MANDATORY, default=no default value]`

The actual name of the database pool

Return

DbPool

An instance of the DbPool.

`DbPool getInstance(String databaseProperty)`

Creates an instance of a DbPool connected to a logical database

Argument

`String` **`databaseProperty`** `[INPUT, MANDATORY, default=no default value]`

The logical name of the database, for example "hcss.test.database"

Return

DbPool

An instance of the DbPool.

`DbPool getInstance()`

"hcss.test.database"

Return

DbPool

An instance of the DbPool.

`DbPool createInstance(String databaseProperty)`

Creates an instance of a DbPool connected to a logical database

Argument

`String` **`databaseProperty`** `[INPUT, MANDATORY]`

Return

```
DbPool createInstance(String databaseProperty)
```

DbPool

An instance of the DbPool.

3.80. DETERMINANT

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixDeterminant
Alias:	DETERMINANT
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixDeterminant

Description

Yields the determinant of a square matrix.

Example

Example 1: Apply a DETERMINANT to a Double2d matrix

```
A=Double2d([ [1,2],[3,4] ])
print DETERMINANT(A) # -2
```

API Summary

Jython Syntax

```
<y>=DETERMINANT(<x>)
```

Properties

```
any square matrix x [INPUT, MANDATORY, default=no default value]
double y [INPUT, NOT_MANDATORY, default=false]
```

Miscellaneous

Does not work for complex matrices.

API details

Properties

any square matrix x [INPUT, MANDATORY, default=no default value]

Any square matrix

double y [INPUT, NOT_MANDATORY, default=false]

Returns a double

See also

- CubicSplineInterpolator
- LinearInterpolator

3.81. DFT2dTask

Full Name:	herschel.ia.toolbox.image.DFT2dTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import DFT2dTask

Description

A Task for two dimensional Discrete Fourier Transforms.

A Task that calculates the 2D Discrete Fourier Transform and the power spectrum.

API Summary

Properties
`Numeric2dData image [INPUT, MANDATORY, default=No default value]`
`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`
`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Numeric2dData image [INPUT, MANDATORY, default=No default value]`

The input image.

`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`

The transform.

`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

The spectrum.

3.82. Display

Full Name:	herschel.ia.gui.image.Display
Alias:	Display
Type:	Java Class -
Import:	from herschel.ia.gui.image import Display
Category:	Image

Description

An Image display for DP. A class to display images.

This class can display a SimpleImage, SimpleCube or any numeric2d or numeric3d object. A status bar, color bar, a zoomed section of the image and an overview of the image are also available.

Examples

Example 1: Display a SimpleImage

```
d = Display(mySimpleImage)
```

Example 2: A basic example on how to Display a Double2d

```
im = Double2d(600, 400)
for i in range(600):
 for j in range(400):
  im.set(i, j, i + j)
d = Display(im)
```

API Summary

Constructors
Display()
The standard constructor
Display(boolean useAsComponent, boolean imageVisible)
A Display constructor
Display(boolean useAsComponent)
A constructor to use Display as a component.
Display(Image imds)
A constructor which displays an Image.
Display(Cube cube)
A constructor which displays a Cube.
Display(Stack stack)
A constructor which displays a Stack.
Display(Image imds, boolean imageVisible)
A Display constructor
Display(Cube imds, boolean imageVisible)
A Display constructor
Display(Stack stack, boolean imageVisible)
A Display constructor

Constructors
Display(Array2dData data) A constructor to display a Numeric2d.
Display(Array2dData data, boolean imageVisible) A Display constructor
Display(Array3dData data) A constructor to display a numeric3d
Display(Array3dData data, boolean imageVisible) A display constructor

Methods
int **getLayerShown**() Returns the number of the shown layer
setVisible(boolean imageVisible) Toggles the display visible or invisible.
JPanel **getComposedComponent**() Returns a panel with all components of this Display on it.
ImageColorbar **getColorBarComponent**() Returns the color bar component.
DivaMainImageDisplay **getComponent**() Returns the component where the image is displayed.
ImagePanner **getPannerComponent**() Returns the panner component with the overview of the image.
StatusPanel **getStatusPanelComponent**() Returns the component with the status panel.
setStatusPanelComponent(StatusPanel imageDisplayStatusPanel) Set a new status panel for the display.
ImageZoom **getZoomComponent**() Returns the component with the zoomed view.
setImage(Array2dData imageData) Set a new numeric2d image.
setImage(Array3dData imageData) Sets a new numeric3d image.
setImage(Image imds) Sets a new Image to Display.
setImage(Cube cube) Sets a new cube to display.
setImage(Stack stack) Sets a new stack to display.
addLayer(Layer layer) Adds a new layer to the display.
addLayer(Array2dData imageData) Adds a new layer with numeric2d data

Methods
addLayer(Array3dData imageData) Adds a new layer with numeric3d data
addLayer(Image imds) Adds a new layer with an Image to the display.
addLayer(Cube cube) Adds a new layer with a Cube to the display.
addLayer(Stack stack) Adds a new layer with a Stack to the display.
Layer **getLayer**() Returns the shown layer.
Layer **getLayer**(int i) Returns the Layer object
showLayer(int i) Shows a chosen layer.
removeLayer() Removes the current layer.
removeLayer(int i) Removes a layer.
updateImage(Array2dData imageData) Updates the layer with numeric2d data.
updateImage(Image imds) Updates the layer with an Image.
updateImage(Array3dData imageData) Updates the layer with numeric3d data.
updateImage(Cube imds) Updates the layer with a cube.
updateImage(Stack stack) Updates the layer with a stack.
setBackground(Color bgColor) Sets the background color.
annotationToolbox() Fires up the annotation toolbox.
CanvasFigure **addAnnotation**(String text, double row, double column) Adds a text annotation.
CanvasFigure **addAnnotationWorldCoordinates**(String text, double ra, double decl) Adds a text annotation
CanvasFigure **addGreekAnnotation**(String text, double row, double column) Adds a Greek annotation

Methods
`CanvasFigure addGreekAnnotationWorldCoordinates(String text, double ra, double decl)` Adds a Greek annotation
`Font getAnnotationFont()` Returns the default font for annotations.
`Font getAnnotationFont(double row, double column)` Returns the font for the specified annotation.
`Font getAnnotationFontWorldCoordinates(double ra, double dec)` Returns the font for the specified annotation.
`Color getAnnotationFontColor()` Returns the default color for annotations.
`Color getAnnotationFontColor(double row, double column)` Returns the color of the specified annotation.
`Color getAnnotationFontColorWorldCoordinates(double ra, double dec)` Returns the color of the specified annotation.
`removeAnnotation(double row, double column)` Removes the specified annotation
`removeAnnotationWorldCoordinates(double ra, double dec)` Remove the specified annotation.
`removeAnnotation(CanvasFigure fig)` Removes the specified annotation.
`removeAnnotations()` Removes all annotations.
`setAnnotationFont(Font f)` Changes the font of all Annotations.
`setAnnotationFont(double row, double column, Font f)` Changes the font of the specified annotations.
`setAnnotationFontWorldCoordinates(double ra, double dec, Font f)` Changes the font of the specified annotations.
`setAnnotationFont(int size)` Changes the font size of all annotations.
`setAnnotationFont(double row, double column, int size)` Changes the font size of the specified annotations.
`setAnnotationFontWorldCoordinates(double ra, double dec, int size)` Changes the font size of the specified annotations.
`setAnnotationFontColor(Color color)` Sets the default color for annotations.
`setAnnotationFontColor(double row, double column, Color color)` Sets the font color for the specified annotation.

Methods
setAnnotationFontColorWorldCoordinates(double ra, double dec, Color color) Sets the font color for the specified annotation.
CanvasFigure **addEllipse**(double row, double column, double w, double h, float lineWidth, Color color) Adds an ellipse.
addFigure(CanvasFigure fig) Adds a CanvasFigure
CanvasFigure **addEllipse**(double row, double column, double w, double h, float lineWidth, Color color, double positionAngle) Adds an ellipse
CanvasFigure **addCircle**(double row, double column, double radius, float lineWidth, Color color) Adds a circle
CanvasFigure **addLine**(double row1, double column1, double row2, double column2, float lineWidth, Color color) Adds a line.
CanvasFigure **addPolygon**(double[] coords, float lineWidth, Color color) Adds a polygon
CanvasFigure **addPolyline**(double[] coords, float lineWidth, Color color) Adds a polyline.
CanvasFigure **addRectangle**(double minRow, double minColumn, double w, double h, float lineWidth, Color color) Adds a rectangle.
ArrayList **addImageContour**(ImageContour imageContour, ArrayList colors, int minLength) Adds an ImageContour.
ArrayList **addImageContour**(ImageContour imageContour, ArrayList colors) Adds an ImageContour.
ArrayList **addWcsImageContour**(ImageContour imageContour, ArrayList colors, int minLength) Add an ImageContour.
ArrayList **addWcsImageContour**(ImageContour imageContour, ArrayList contourColors) Adds an ImageContour.
ArrayList **addContourLevel**(ContourLevel contourLevel, Color contourColor, int minimumContourLength) Adds a ContourLevel.
ArrayList **addContourLevel**(ContourLevel level, Color color) Adds a ContourLevel.
ArrayList **addContourLevel**(ContourLevel level, Wcs wcs, Color color, int minLength)

Methods
Adds a ContourLevel.
`ArrayList` **`addContourLevel`**`(ContourLevel level, Wcs wcs, Color color)` addContourLevel(ContourLevel level, Wcs wcs, Color color)
`ImageFigure` **`addContour`**`(Contour contour, Color color, int minLength)` Adds a Contour.
`ImageFigure` **`addContour`**`(Contour contour, Color color)` Adds a Contour
`ImageFigure` **`addContour`**`(Contour contour, Wcs wcs, Color color, int minLength)` Adds a Contour.
`ImageFigure` **`addContour`**`(Contour contour, Wcs wcs, Color color)` Adds a Contour.
`ArrayList` **`addPositionList`**`(PositionList positionList, Color color)` Overlays the given source list on this Display,
`ArrayList` **`addPositionList`**`(PositionList positionList)` Overlays the given source list on this Display,
`ArrayList` **`showAxes`**`(boolean showAxis)` Enables or disables the axes for the displayed image.
`ImageAxis` **`getLeftaxis`**`()` Returns the left axis.
`ImageAxis` **`getRightaxis`**`()` Returns the right axis.
`ImageAxis` **`getBottomaxis`**`()` Returns the bottom axis.
`ImageAxis` **`getTopaxis`**`()` Returns the top axis.
`ArrayList` **`getAxes`**`()` Returns an array with the axes.
`addAxis``(String label, Position orientation)` Adds new axis.
`deleteAxis``(ImageAxis axis)` Deletes the given axis.
`close``()` Closes the display.
`editColors``()` Edit the colors using a popup.
`editCutLevels``()` Edit the cut levels using a popup.
`String` **`getColortable`**`()` Returns the name of the color table.

Methods
`double[] getCutLevels()` Returns the cut levels.
`Image getImage()` Returns the displayed image.
`double[] getPixelCoordinates(double c1, double c2)` Returns the image coordinates
`Number getIntensity(int row, int column)` Returns the intensity of the pixel
`Number getIntensityFromWorldCoordinates(double c1, double c2)` Returns the intensity of the pixel.
`Unit getUnit()` Returns the unit.
`float getZoomFactor()` Returns the used zoom factor.
`printDialog()` Opens a printer dialog.
`getPrintControl()` Returns the printControl.
`createPrintJob()` Creates a print job.
`PrinterJob getPrintJob()` Returns the printer job.
`setPrintJob(PrinterJob job)` Sets a printer job.
`print(String path)` Print the displayed image to a ps file.
`print(String path, String orientation)` Prints the displayed image to a ps file.
`save()` Pops up a dialog to save your image.
`saveAsJPG(String filename)` Saves the display as a jpg file.
`saveScreenshot(String filename)` Save the file as a screenshot.
`saveCurrentView(String filename)` Saves the current view of the image.
`setColortable(String colortableName)` Sets the color table.
`setColortable(String colortableName, String intensityName)` Sets the color table.
`setColortable(String colortableName, String intensityName, String scaleName)`

Methods
Sets the color table.
setCutLevels(double percent) Sets the cut levels.
setCutLevels(double min, double max) Sets the cut levels.
setCutLevels() Sets the cut levels.
setCutLevels(double[] minmax) Sets the cut levels
setUnit(Unit u) Sets the unit.
setZoomFactor(float zoomFactor) Sets the zoom factor.
zoom(double row, double column, float zoomFactor) Zooms the image.
zoomWorldCoordinates(double ra, double decl, float zoomFactor) Zooms the image.
zoomIn() Zooms the image in.
zoomOut() Zoom the image out.
zoomFit() Zoom the image.
flipYAxis() Flips the y axis of the displayed image.
setFlipYAxis(boolean flipAxis) Sets whether the current image should be flipped.
getFlipYAxis() Returns whether the Y axis is flipped.
isFlipped() Returns whether the current layer is flipped.
setDepthAxis(int depthAxis) Sets the depth axis for cubes.
getDepthAxis() Returns the depth axis for cubes.

API details

Constructors

Display()
The standard constructor for Display. This constructor shows an empty display window.

Display(boolean useAsComponent, boolean imageVisible)

A constructor where you have the possibility to display the image in the background (which means the image will not be shown) or to use the display as a component to include in a gui in jide.

Arguments

`boolean` **`useAsComponent`** `[INPUT, MANDATORY]`

`boolean` **`imageVisible`** `[INPUT, MANDATORY]`

Display(boolean useAsComponent)

A constructor where you have the possibility to use the display as a component to include in a gui in jide.

Argument

`boolean` **`useAsComponent`** `[INPUT, MANDATORY]`

Example

Example how to use Display as a component

```
from javax.swing import * from java.awt import BorderLayout
dc = Display(True)
frame = JFrame()
frame.setTitle("Image display using Components")
frame.setSize(800, 600) # The main component, with the image
component = dc.getComponent()
# The zoom, panner, status and colorbar components
zoomComponent = dc.getZoomComponent()
pannerComponent = dc.getPannerComponent()
statusComponent = dc.getStatusPanelComponent()
colorbarComponent = dc.getColorBarComponent()
# Adding the components to the frame
frame.getContentPane().add(component, BorderLayout.CENTER)
frame.getContentPane().add(statusComponent, BorderLayout.NORTH)
frame.getContentPane().add(colorbarComponent, BorderLayout.SOUTH)
frame.show()
dc.setImage(image)
```

Display(Image imds)

A constructor which immediately displays the given Image.

Argument

`Image` **`imds`** `[INPUT, MANDATORY]`

Display(Cube cube)

A constructor which immediately displays the given Cube.

Argument

`Cube` **`cube`** `[INPUT, MANDATORY]`

Display(Stack stack)

A constructor which immediately displays the given Stack.

Argument

`Stack` **`stack`** `[INPUT, MANDATORY]`

Display(Image imds, boolean imageVisible)

A constructor which immediately displays the given Image. It is possible to not yet display the Image.

Arguments

`Image` **`imds`** `[INPUT, MANDATORY]`

Display(Image imds, boolean imageVisible)

boolean **imageVisible** [INPUT, MANDATORY]

Display(Cube imds, boolean imageVisible)

A constructor which immediately displays the given Cube. It is possible to not yet display the Cube.

Arguments

Cube **imds** [INPUT, MANDATORY]

boolean **imageVisible** [INPUT, MANDATORY]

Display(Stack stack, boolean imageVisible)

A constructor which immediately displays the given Stack. It is possible to not yet display the Stack.

Arguments

Stack **stack** [INPUT, MANDATORY]

boolean **imageVisible** [INPUT, MANDATORY]

Display(Array2dData data)

A constructor which immediately displays the given numeric2d

Argument

Array2dData **data** [INPUT, MANDATORY]

Display(Array2dData data, boolean imageVisible)

A constructor which immediately displays the given numeric2d. It is possible to not yet display the image.

Arguments

Array2dData **data** [INPUT, MANDATORY]

boolean **imageVisible** [INPUT, MANDATORY]

Display(Array3dData data)

A constructor which immediately displays the given numeric3d

Argument

Array3dData **data** [INPUT, MANDATORY]

Display(Array3dData data, boolean imageVisible)

A constructor which immediately displays the given numeric3d

Arguments

Array3dData **data** [INPUT, MANDATORY]

boolean **imageVisible** [INPUT, MANDATORY]

Methods

int getLayerShown()

Returns the number of the layer which is shown.

Return

int

The number of the layer which is shown.

setVisible(boolean imageVisible)

Toggles the imageDisplay visible or invisible. True to make the image visible, False to hide the image.

Argument

`boolean` **`imageVisible`** `[INPUT, MANDATORY]`

JPanel getComposedComponent()

Returns a panel with all components of this Display on it.

Return

JPanel

Returns a panel with all component of this Display on it.

ImageColorbar getColorBarComponent()

Returns a JComponent that contains the color bar. The color bar can be used to create your own component based Display.

Return

ImageColorbar

The ImageColorbar to use in you own components.

DivaMainImageDisplay getComponent()

Returns the component where the image is shown. This can be used to make your own GUI's with an image.

Return

DivaMainImageDisplay

The component to use in your own GUI's

Example

Example on how to integrate Display in your own GUI.

```
from javax.swing import * from java.awt import BorderLayout
dc = Display(True)
frame = JFrame() frame.setTitle("Image display using Components")
frame.setSize(800, 600)
# The main component, with the image
component = dc.getComponent()
# The zoom, panner, status and colorbar components
zoomComponent = dc.getZoomComponent()
pannerComponent = dc.getPannerComponent()
statusComponent = dc.getStatusPanelComponent()
colorbarComponent = dc.getColorBarComponent()
# Adding the components to the frame
frame.getContentPane().add(component, BorderLayout.CENTER)
frame.getContentPane().add(statusComponent, BorderLayout.NORTH)
frame.getContentPane().add(colorbarComponent, BorderLayout.SOUTH)
frame.show()
dc.setImage(image)
```

ImagePanner getPannerComponent()

Returns the component which contains the overview of the image (100x100). This can be used to make your own GUI's with an image.

Return

ImagePanner getPannerComponent()

ImagePanner

The component with the overview to use in your own GUI's

StatusPanel getStatusPanelComponent()

Returns the component which contains the status bar of the image. The status exists of zoom buttons, magnification, pixel coordinates, pixel intensity and sky coordinates. This can be used to make your own GUI's with an image.

Return

StatusPanel

The component with the status panel to use in your own GUI's

setStatusPanelComponent(StatusPanel imageDisplayStatusPanel)

Attaches a new status panel to the display.

Argument

StatusPanel **imageDisplayStatusPanel** [INPUT, MANDATORY]

ImageZoom getZoomComponent()

Returns the component which contains the a zoomed view of the image. The zoomed view has a zoom factor of 4. This component can be used to make your own GUI's with an image.

Return

ImageZoom

The component with the zoomed view to use in your own GUI's

setImage(Array2dData imageData)

Sets a new image on the display. A Bool2d, Int2d, Double2d, ... can be used.

Argument

Array2dData **imageData** [INPUT, MANDATORY]

setImage(Array3dData imageData)

Shows a new image on the display. A Bool3d, Int3d, Double3d, ... can be used. The first image is show, the other images are used as extra layers.

Argument

Array3dData **imageData** [INPUT, MANDATORY]

setImage(Image imds)

Shows a new image on the display. The world-coordinate system of the Image is used.

Argument

Image **imds** [INPUT, MANDATORY]

setImage(Cube cube)

A Cube is shown on the display. The world-coordinate system of the Cube is used.

Argument

Cube **cube** [INPUT, MANDATORY]

setImage(Stack stack)

A Stack is shown on the display. The world-coordinate system of the Cube is used.

Argument

Stack **stack** [INPUT, MANDATORY]

addLayer(Layer layer)

Adds a new layer to the display. A Layer must be constructed before.

Argument

Layer **layer** [INPUT, MANDATORY]

addLayer(Array2dData imageData)

Adds a new layer to the display. A Bool2d, Int2d, Double2d, ... can be used.

Argument

Array2dData **imageData** [INPUT, MANDATORY]

addLayer(Array3dData imageData)

Adds new layers to the display. A Bool3d, Int3d, Double3d, ... can be used.

Argument

Array3dData **imageData** [INPUT, MANDATORY]

addLayer(Image imds)

Adds a new layer to the display. An Image is added as extra layer of the display.

Argument

Image **imds** [INPUT, MANDATORY]

addLayer(Cube cube)

Adds a new layer to the display. A Cube is added as extra layer of the display.

Argument

Cube **cube** [INPUT, MANDATORY]

addLayer(Stack stack)

Adds a new layer to the display. A Stack is added as extra layer of the display.

Argument

Stack **stack** [INPUT, MANDATORY]

Layer getLayer()

Returns the Layer that is shown.

Return

Layer

The layer that is shown at the moment.

Layer getLayer(int i)

The Layer with the given index is returned.

Argument

Layer getLayer(int i)

int **i** [INPUT, MANDATORY]

Return

Layer

The layer with the given index.

showLayer(int i)

Shows the Layer with the given index. The Layer with the given index is shown.

Argument

int **i** [INPUT, MANDATORY]

removeLayer()

Removes the Layer that is shown.

removeLayer(int i)

Removes the given Layer.

Argument

int **i** [INPUT, MANDATORY]

updateImage(Array2dData imageData)

Updates the layer of the image that is displayed. A numeric2d will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.

Argument

Array2dData **imageData** [INPUT, MANDATORY]

updateImage(Image imds)

Updates the layer of the image that is displayed. An Image will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.

Argument

Image **imds** [INPUT, MANDATORY]

updateImage(Array3dData imageData)

Updates the layer of the image that is displayed. A numeric3d will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.

Argument

Array3dData **imageData** [INPUT, MANDATORY]

updateImage(Cube imds)

Updates the layer of the image that is displayed. A cube will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.

Argument

Cube **imds** [INPUT, MANDATORY]

updateImage(Stack stack)

Updates the layer of the image that is displayed. A Stack will be used as input for the new layer. The annotations, cut levels and zoom factor are kept.

updateImage(Stack stack)

Argument

Stack **stack** [INPUT, MANDATORY]

setBackground(Color bgColor)

Changes the background of the image. The background can be black or white.

Argument

Color **bgColor** [INPUT, MANDATORY]

annotationToolbox()

Returns the AnnotationToolbox for the display. The AnnotationToolbox lets the user put annotation on the image, like ellipses, rectangles, text, ...

CanvasFigure addAnnotation(String text, double row, double column)

Adds a text annotation A text annotation will be placed, starting at the given pixel coordinates.

Arguments

String **text** [INPUT, MANDATORY]

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Return

CanvasFigure

The annotation as a CanvasFigure

CanvasFigure addAnnotationWorldCoordinates(String text, double ra, double decl)

A text annotation will be placed, starting at the given sky coordinates.

Arguments

String **text** [INPUT, MANDATORY]

double **ra** [INPUT, MANDATORY]

double **decl** [INPUT, MANDATORY]

Return

CanvasFigure

The annotation as a CanvasFigure

CanvasFigure addGreekAnnotation(String text, double row, double column)

A greek text annotation will be placed, starting at the given pixel coordinates. All ascii text will be converted to greek characters : a becomes alpha, b becomes beta, ...

Arguments

String **text** [INPUT, MANDATORY]

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Return

CanvasFigure

CanvasFigure addGreekAnnotation(String text, double row, double column)

The annotation as a CanvasFigure

CanvasFigure addGreekAnnotationWorldCoordinates(String text, double ra, double decl)

A greek text annotation will be placed, starting at the given world coordinates. All ascii text will be converted to greek characters : a becomes alpha, b becomes beta, ...

Arguments

String **text** [INPUT, MANDATORY]

double **ra** [INPUT, MANDATORY]

double **decl** [INPUT, MANDATORY]

Return

CanvasFigure

The annotation as a CanvasFigure

Font getAnnotationFont()

Returns the font used for the annotations.

Return

Font

The font used for the annotations.

Font getAnnotationFont(double row, double column)

Returns the font used for the annotation that begins at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Return

Font

The font used for the specified annotation.

Font getAnnotationFontWorldCoordinates(double ra, double dec)

Returns the font used for the annotation that begins at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

Return

Font

The font used for the specified annotation.

Color getAnnotationFontColor()

Returns the default color used for the text annotations.

Color getAnnotationFontColor()

Return

Color

The default color used for the text annotations.

Color getAnnotationFontColor(double row, double column)

Returns the font color used for the annotation that begins at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Return

Color

The color used for the specified text annotation.

Color getAnnotationFontColorWorldCoordinates(double ra, double dec)

Returns the color used for the annotation that begins at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

Return

Color

The color used for the specified text annotation.

removeAnnotation(double row, double column)

Removes the annotation at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

removeAnnotationWorldCoordinates(double ra, double dec)

Removes the annotation at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

removeAnnotation(CanvasFigure fig)

Removes the given annotation.

Argument

CanvasFigure **fig** [INPUT, MANDATORY]

removeAnnotations()

Removes all the annotations from the display.

setAnnotationFont(Font f)

Changes the font of all annotations that are visible on the Display.

Argument

Font **f** [INPUT, MANDATORY]

setAnnotationFont(double row, double column, Font f)

Changes the font of the annotations which are located at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Font **f** [INPUT, MANDATORY]

setAnnotationFontWorldCoordinates(double ra, double dec, Font f)

Changes the font of the annotations which are located at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

Font **f** [INPUT, MANDATORY]

setAnnotationFont(int size)

Changes the font size of all annotations.

Argument

int **size** [INPUT, MANDATORY]

setAnnotationFont(double row, double column, int size)

Changes the font size of the annotations which are located at the given pixel coordinates.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

int **size** [INPUT, MANDATORY]

setAnnotationFontWorldCoordinates(double ra, double dec, int size)

Changes the font size of the annotations which are located at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

int **size** [INPUT, MANDATORY]

setAnnotationFontColor(Color color)

Changes the font color of all annotations.

Argument

Color **color** [INPUT, MANDATORY]

setAnnotationFontColor(double row, double column, Color color)

Changes the font color of the annotations which are located at the given pixel coordinates.

Arguments

setAnnotationFontColor(double row, double column, Color color)

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

setAnnotationFontColorWorldCoordinates(double ra, double dec, Color color)

Changes the font color of the annotations which are located at the given world coordinates.

Arguments

double **ra** [INPUT, MANDATORY]

double **dec** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, Color color)

Adds an ellipse to the image at a given place, with a given dimension, line width and color.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

double **w** [INPUT, MANDATORY]

double **h** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

CanvasFigure

The annotation as CanvasFigure

addFigure(CanvasFigure fig)

Adds the given CanvasFigure to the image.

Argument

CanvasFigure **fig** [INPUT, MANDATORY]

CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, Color color, double positionAngle)

Adds an ellipse to the image at a given place, with a given dimension, line width, color and position angle.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

double **w** [INPUT, MANDATORY]

double **h** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

double **positionAngle** [INPUT, MANDATORY]

Return

CanvasFigure addEllipse(double row, double column, double w, double h, float lineWidth, Color color, double positionAngle)

CanvasFigure

The annotation as CanvasFigure

CanvasFigure addCircle(double row, double column, double radius, float lineWidth, Color color)

Adds a circle to the image at a given place, with a given dimension, line width and color.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

double **radius** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

CanvasFigure

The annotation as CanvasFigure

CanvasFigure addLine(double row1, double column1, double row2, double column2, float lineWidth, Color color)

Adds an line to the image at a given place, with a given line width and color.

Arguments

double **row1** [INPUT, MANDATORY]

double **column1** [INPUT, MANDATORY]

double **row2** [INPUT, MANDATORY]

double **column2** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

CanvasFigure

The annotation as CanvasFigure

CanvasFigure addPolygon(double[] coords, float lineWidth, Color color)

Adds a polygon to the image at a given place, with a given line width and color. The coordinates should be given as [row1, column1, row2, column2, ...]

Arguments

double[] **coords** [INPUT, MANDATORY]

float **lineWidth** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

CanvasFigure

The annotation as CanvasFigure

CanvasFigure addPolyline(double[] coords, float lineWidth, Color color)

Adds a polyline to the image at a given place, with a given line width and color. The coordinates should be given as [row1, column1, row2, column2, ...]

Arguments

`double[] coords [INPUT, MANDATORY]`

`float lineWidth [INPUT, MANDATORY]`

`Color color [INPUT, MANDATORY]`

Return

CanvasFigure

The annotation as CanvasFigure

CanvasFigure addRectangle(double minRow, double minColumn, double w, double h, float lineWidth, Color color)

Adds an rectangle to the image at a given place, with a given dimension, line width and color.

Arguments

`double minRow [INPUT, MANDATORY]`

`double minColumn [INPUT, MANDATORY]`

`double w [INPUT, MANDATORY]`

`double h [INPUT, MANDATORY]`

`float lineWidth [INPUT, MANDATORY]`

`Color color [INPUT, MANDATORY]`

Return

CanvasFigure

The annotation as CanvasFigure

ArrayList addImageContour(ImageContour imageContour, ArrayList colors, int minLength)

Draws the Contours of the given ImageContour in the given Color on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

`ImageContour imageContour [INPUT, MANDATORY]`

`ArrayList colors [INPUT, MANDATORY]`

`int minLength [INPUT, MANDATORY]`

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

ArrayList addImageContour(ImageContour imageContour, ArrayList colors)

Draws the Contours of the given ImageContour in the given Colors on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ArrayList addImageContour(ImageContour imageContour, ArrayList colors)

ImageContour **imageContour** [INPUT, MANDATORY]

ArrayList **colors** [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

ArrayList addWcsImageContour(ImageContour imageContour, ArrayList colors, int minLength)

Draws the Contours of the given ImageContour in the given Color on the image based on the Wcs (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ImageContour **imageContour** [INPUT, MANDATORY]

ArrayList **colors** [INPUT, MANDATORY]

int **minLength** [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

ArrayList addWcsImageContour(ImageContour imageContour, ArrayList contourColors)

Draws the Contours of the given ImageContour in the given Color on the image based on the Wcs and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ImageContour **imageContour** [INPUT, MANDATORY]

ArrayList **contourColors** [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(ContourLevel contourLevel, Color contourColor, int minimumContourLength)

Draws the Contours of the given ContourLevel in the given Color on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel **contourLevel** [INPUT, MANDATORY]

Color **contourColor** [INPUT, MANDATORY]

int **minimumContourLength** [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(ContourLevel level, Color color)

Draws the Contours of the given ContourLevel in the given Color on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel **level** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

ArrayList

The drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(ContourLevel level, Wcs wcs, Color color, int minLength)

Draws the Contours of the given ContourLevel in the given Color on the image (if their length is at least the given minimum length) and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel **level** [INPUT, MANDATORY]

Wcs **wcs** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

int **minLength** [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

ArrayList addContourLevel(ContourLevel level, Wcs wcs, Color color)

Draws the Contours of the given ContourLevel in the given Color on the image and returns the drawn Contours as an ArrayList of ImageFigures.

Arguments

ContourLevel **level** [INPUT, MANDATORY]

Wcs **wcs** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

ArrayList

Returns the drawn Contours as an ArrayList of ImageFigures.

ImageFigure addContour(Contour contour, Color color, int minLength)

Draws the given Contour on the image (if its length is at least the given minimum length) and returns the drawn Contour as an ImageFigure.

Arguments

Contour **contour** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

int **minLength** [INPUT, MANDATORY]

Return

ImageFigure addContour(Contour contour, Color color, int minLength)

ImageFigure

Returns the drawn Contour as an ImageFigure.

ImageFigure addContour(Contour contour, Color color)

Draws the given Contour in the given Color on the image and returns the drawn Contour as an ImageFigure.

Arguments

Contour **contour** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

ImageFigure

Returns the drawn Contour as an ImageFigure.

ImageFigure addContour(Contour contour, Wcs wcs, Color color, int minLength)

Draws the given Contour in the given Color on the image (if its length is at least the given minimum length) based on the Wcs and returns the drawn Contour as an ImageFigure.

Arguments

Contour **contour** [INPUT, MANDATORY]

Wcs **wcs** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

int **minLength** [INPUT, MANDATORY]

Return

ImageFigure

Returns the drawn Contour as an ImageFigure.

ImageFigure addContour(Contour contour, Wcs wcs, Color color)

Draws the given Contour in the given Color on the image based on the Wcs and returns the drawn Contour as an ImageFigure.

Arguments

Contour **contour** [INPUT, MANDATORY]

Wcs **wcs** [INPUT, MANDATORY]

Color **color** [INPUT, MANDATORY]

Return

ImageFigure

Returns the drawn Contour as an ImageFigure.

ArrayList addPositionList(PositionList positionList, Color color)

in the given color.

Arguments

PositionList **positionList** [INPUT, MANDATORY]

ArrayList addPositionList(PositionList positionList, Color color)

Color **color** [INPUT, MANDATORY]

Return

ArrayList

Returns a list of sources.

ArrayList addPositionList(PositionList positionList)

in the given color.

Argument

PositionList **positionList** [INPUT, MANDATORY]

Return

ArrayList

Returns a list of sources.

ArrayList showAxes(boolean showAxis)

Enables or disables the axes for the displayed image. If the parameter is true, two axes are drawn, one on the left side and one at the bottom. An array with the axes is returned.

Argument

boolean **showAxis** [INPUT, MANDATORY]

Return

ArrayList

An ArrayList with the axes.

ImageAxis getLeftaxis()

Returns the left axis of the Display. If there is no left axis, the standard left axis is made, shown and returned.

Return

ImageAxis

The left axis

ImageAxis getRightaxis()

Returns the right axis of the Display. If there is no right axis, the standard right axis is made, shown and returned.

Return

ImageAxis

The right axis

ImageAxis getBottomaxis()

Returns the bottom axis of the Display. If there is no bottom axis, the standard bottom axis is made, shown and returned.

Return

ImageAxis

ImageAxis getBottomaxis()

The bottom axis

ImageAxis getTopaxis()

Returns the top axis of the Display. If there is no top axis, the standard top axis is made, shown and returned.

Return

ImageAxis

The top axis

ArrayList getAxes()

Returns an array with the axes. Using methods on the Axes, the appearance can be changed.

Return

ArrayList

An ArrayList with the Axes.

addAxis(String label, Position orientation)

Adds a new axis.

Arguments

`String` **label** `[INPUT, MANDATORY]`

`Position` **orientation** `[INPUT, MANDATORY]`

deleteAxis(ImageAxis axis)

Deletes the given axis.

Argument

`ImageAxis` **axis** `[INPUT, MANDATORY]`

close()

Closes the display and frees the memory.

editColors()

A windows opens where you can change the color table, intensity and scale.

editCutLevels()

A windows opens where you can set the cut levels of the image.

String getColortable()

Returns the name of the color table

Return

String

The name of the color table

double[] getCutLevels()

Returns an array with the cut levels of the displayed image.

Return

double[] getCutLevels()

double[]

The cut levels of the displayed image.

Image getImage()

Returns the displayed Image.

Return

Image

The displayed Image

double[] getPixelCoordinates(double c1, double c2)

Returns the image coordinates of the given world coordinates

Arguments

`double` **`c1`** `[INPUT, MANDATORY]`

`double` **`c2`** `[INPUT, MANDATORY]`

Return

double[]

The corresponding image coordinates as a 2 dimensional array of doubles, the first one describing the row and the second the column.

Number getIntensity(int row, int column)

Returns the intensity of the pixel with given pixel coordinates

Arguments

`int` **`row`** `[INPUT, MANDATORY]`

`int` **`column`** `[INPUT, MANDATORY]`

Return

Number

The intensity of the given pixel

Number getIntensityFromWorldCoordinates(double c1, double c2)

Returns the intensity of the pixel with given world coordinates.

Arguments

`double` **`c1`** `[INPUT, MANDATORY]`

`double` **`c2`** `[INPUT, MANDATORY]`

Return

Number

double The intensity of the given pixel.

Unit getUnit()

Returns the unit of the image

Return

Unit

Unit getUnit()

The unit of the image.

float getZoomFactor()

Returns the used zoom factor. The returned zoomfactor is bigger than 1 if the image is zoomed in, otherwise, the zoomfactor is between 0 and 1

Return

float

The zoomfactor of the displayed image.

printDialog()

A dialog is opened where you can choice the printer, number of copies, page setup and appearance.

getPrintControl()

Returns the printControl to make it possible to print using the command line.

Example

How to print from the command line

```
job = d.getPrintJob()
job.setCopies(2) # Set the number of copies
pservices = job.lookupPrintServices()
job.setPrintService(pservices[3]) #Use the printer of choice
d.setPrintJob(job)
job.print()
```

createPrintJob()

Creates a print job.

PrinterJob getPrintJob()

Returns the printer job.

Return

PrinterJob

The printerJob.

setPrintJob(PrinterJob job)

Set a given printer job.

Argument

PrinterJob **job** [INPUT, MANDATORY]

print(String path)

Prints the displayed image to a ps file. The settings can be changed with the method setPrintJob(job) or with the GUI obtained from the method createPrintJob()

Argument

String **path** [INPUT, MANDATORY]

print(String path, String orientation)

Prints the displayed image to a ps file. The settings can be changed with the method setPrintJob(job) or with the GUI obtained from the method createPrintJob() The orientation of

print(String path, String orientation)

the page can be "landscape" (or "l"), "portrait" (or "p"), "reverse_landscape" (or "rl"), "reverse_portrait" (or "rp")

Arguments

`String` **path** [INPUT, MANDATORY]

`String` **orientation** [INPUT, MANDATORY]

save()

Pops up a file chooser and saves the image as fits, jpeg, tiff, png or bmp file.

saveAsJPG(String filename)

Saves the display as a jpg file.

Argument

`String` **filename** [INPUT, MANDATORY]

saveScreenshot(String filename)

Saves the file as screenshot. Saves the display as jpg, png, tiff, bmp or FITS.

Argument

`String` **filename** [INPUT, MANDATORY]

saveCurrentView(String filename)

Saves the view as screenshot. The annotations are also saved!

Argument

`String` **filename** [INPUT, MANDATORY]

setColortable(String colortableName)

Sets the color table of the displayed image.

Argument

`String` **colortableName** [INPUT, MANDATORY]

setColortable(String colortableName, String intensityName)

Sets the colortable of the displayed image.

Arguments

`String` **colortableName** [INPUT, MANDATORY]

`String` **intensityName** [INPUT, MANDATORY]

setColortable(String colortableName, String intensityName, String scaleName)

Sets the colortable Sets the colortable of the displayed image

Arguments

`String` **colortableName** [INPUT, MANDATORY]

`String` **intensityName** [INPUT, MANDATORY]

`String` **scaleName** [INPUT, MANDATORY]

setCutLevels(double percent)

Sets the cut level of the displayed image.

setCutLevels(double percent)

Argument

double **percent** [INPUT, MANDATORY]

setCutLevels(double min, double max)

Sets the cut level of the displayed image.

Arguments

double **min** [INPUT, MANDATORY]

double **max** [INPUT, MANDATORY]

setCutLevels()

Sets the cut level of the displayed image.

setCutLevels(double[] minmax)

Sets the cut level of the displayed image.

Argument

double[] **minmax** [INPUT, MANDATORY]

setUnit(Unit u)

Sets the unit of the displayed image.

Argument

Unit **u** [INPUT, MANDATORY]

setZoomFactor(float zoomFactor)

Sets the zoomfactor of the displayed image.

Argument

float **zoomFactor** [INPUT, MANDATORY]

zoom(double row, double column, float zoomFactor)

Sets the zoomfactor of the displayed image and the coordinates which should appear in the center of the image.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

float **zoomFactor** [INPUT, MANDATORY]

zoomWorldCoordinates(double ra, double decl, float zoomFactor)

Sets the zoomfactor of the displayed image and the coordinates which should appear in the center of the image.

Arguments

double **ra** [INPUT, MANDATORY]

double **decl** [INPUT, MANDATORY]

float **zoomFactor** [INPUT, MANDATORY]

zoomIn()

Zooms the displayed image. The zoom factor changes the following way : ..., 1/4, 1/3, 1/2, 1, 2, 3, 4, ...

zoomOut()

Zooms the displayed image out. The zoom factor changes the following way : ..., 4, 3, 2, 1, 1/2, 1/3, 1/4, ...

zoomFit()

Zooms the image to a zoomfactor so that the image is shown in total on the screen. Axes are taken into account

flipYAxis()

If this method is called, the current axis is flipped. If the displayed image has the WCS keyword flipy, this method will have no effect.

setFlipYAxis(boolean flipAxis)

The current image will be flipped.

Argument

`boolean` **`flipAxis`** `[INPUT, MANDATORY]`

getFlipYAxis()

Returns true if the current image is flipped.

isFlipped()

Returns true if the current layer is flipped.

setDepthAxis(int depthAxis)

Set the depth axis for the current and newly added cubes. The first axis (0) is the standard depth axis.

Argument

`int` **`depthAxis`** `[INPUT, MANDATORY]`

getDepthAxis()

Returns which axis is the depth axis.

See also

- SimpleImage
- SimpleCube

3.83. displaylog

Full Name:	herschel.ia.toolbox.util.DisplayLogTask
Alias:	displaylog
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import DisplayLogTask
Category:	task

Description

Open a window displying log messages

The display task is used to open a log window handler and assign it to a log

Examples

Example 1: attach the window to a logger

```
# create a logger
log=java.util.logging.Logger.getLogger("")
# display the logger in the window
displaylog(log)
# sending logs
log.info("ahh, an info message")
log.warning("ehm, a warning message")
log.severe("oops! a severe message")
```

Example 2: remove the window attached to a logger

```
# create a logger
log=java.util.logging.Logger.getLogger("")
# display the logger in the window
displaylog(log)
# sending logs
log.info("ahh, an info message")
# stop displaing messages from log by removing it from the window
displaylog(log, option="remove")
# no displaied any longer
log.info("ahh, an new info message")
```

Example 3: change the size (number of lines to be displaied) in the window

```
displaylog(option="change", size=8000)
```

API Summary

Jython Syntax

```
displaylog(logger)
```

Properties

```
Logger logger [INPUT, No,
default=java.util.logging.Logger.getLogger("")]
```

```
String option [INPUT, No, default="add"]
```

```
Integer size [INPUT, No, default=4096]
```

Limitations

No time buffering, logs could clutter the java window system

Miscellaneous

No miscellaneous

API details

Properties

```
Logger logger [INPUT, No,
default=java.util.logging.Logger.getLogger("")]
```

The log to display the messages into the window

```
String option [INPUT, No, default="add"]
```

Specify the action to be taken, default value is "add" which enables the display for the specified logger, the "remove" option stops displaying messages for an attached logger, the "change" option works in pair with parameter size.

```
Integer size [INPUT, No, default=4096]
```

Specify the number of lines to be displaied in the window, the parameter is effectively set up *only* if the "change" value for the parameter "option" is specified

History

- `2007-01-26 - NdC`: first release

3.84. displayQCLog

Full Name:	herschel.ia.qcp.DisplayQCLogTask
Alias:	displayQCLog
Type:	Java Task -
Import:	from herschel.ia.qcp import DisplayQCLogTask
Category:	task

Description

Open a window displaying QClLogs messages organized in a product

The show task is used to open a log window handler and assign it to a log

Example

Example 1: display the QCLog

```
displayQCLog()
```

Limitations

No time buffering, logs could clutter the java window system

Miscellaneous

No miscellaneous

History

- `2007-03-22 - NdC`: first release

3.85. DivideSpectrum

Full Name:	herschel.ia.toolbox.spectrum.DivideSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import DivideSpectrum

Description

Task for dividing the flux data included in a spectrum container by a scalar or for dividing two spectrum containers on a spectrum-by-spectrum basis by each other.

For the scalar mode, 'ds' and 'param' need to specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1,3,4,2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1,3,4]` and for the pair operation `segments1=[1,3,4], segments1=[3,6,5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the orginal datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import DivideSpectrum
divide = DivideSpectrum()
dividedByFactor = divide(ds=spectra, param=2.1)
dividedByFactor = divide(ds=spectra, selection={"bbtype":[6031]}, param=2.1)
dividedByFactor = divide(ds=spectra, selection=[0,1,2,3], param=2.1)
dividedByFactor = divide(ds=spectra, selection={"Chopper":([-4.4,5.9],0.2),
 "bbtype":[6031]}, param=2.1)
dividedByFactor = divide(ds=spectra, selection={"LoFrequency":(4000.0,5000.0)},
 param=2.1)
dividedPairWise = divide(ds1=spectra1, ds2=spectra2)
dividedPairWise = divide(ds1=spectra1, ds2=spectra2, selection={"bbtype":
[6031]})
dividedPairWise = divide(ds1=spectra1, ds2=spectra2, selection=[0,1,2,3])
dividedPairWise = divide(ds1=spectra1, ds2=spectra2, selection={"Chopper":
([-4.4,5.9],0.2), "bbtype":[6031]})
```

Example 1: from Jide:

```
dividedPairWise = divide(ds1=spectra1, ds2=spectra2, selection={"LoFrequency":
(4000.0,5000.0)})
```

API Summary

Properties
SpectrumContainer **ds** [INPUT, OPTIONAL, default=no default value.]
Double **param** [INPUT, OPTIONAL, default=no default value.]
Object **selection** [INPUT, OPTIONAL, default=None.]
PyDictionary|Map> **selection_lookup** [INPUT, OPTIONAL, default=no default value.]
PyList **selection_index** [INPUT, OPTIONAL, default=No default value.]
Boolean **overwrite** [INPUT, OPTIONAL, default=False.]
SpectrumContainer **ds1** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **ds2** [INPUT, OPTIONAL, default=no default value.]
Object **segments** [INPUT, OPTIONAL, default=no default value.]
Object **segments1** [INPUT, OPTIONAL, default=no default value.]
Object **segments2** [INPUT, OPTIONAL, default=no default value.]
String **variant** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]

Input container to be processed by the task in case the task is used as a scalar operation.

Double param [INPUT, OPTIONAL, default=no default value.]

The scalar parameter to be considered when the task is used as scalar operation.

Object selection [INPUT, OPTIONAL, default=None.]

Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra for which the add should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also

- SpectrumTask

History

- `2007-08-17 - meli`: initial.

- `2008-03-16 - meli`: major refactoring of the toolbox.
- `2008-10-03 - meli`: segments, javadoc

3.86. Double1d

Full Name:	herschel.ia.numeric.Double1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Double1d

Description

A rectangular numeric double array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.87. Double2d

Full Name:	herschel.ia.numeric.Double2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Double2d

Description

A rectangular numeric double array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.88. Double3d

Full Name:	herschel.ia.numeric.Double3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Double3d

Description

A rectangular numeric double array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.89. Double4d

Full Name:	herschel.ia.numeric.Double4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Double4d

Description

A rectangular numeric double array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.90. Double5d

Full Name:	herschel.ia.numeric.Double5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Double5d

Description

A rectangular numeric double array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.91. EigenvalueDecomposition

Full Name:	herschel.ia.numeric.toolbox.matrix.EigenvalueDecomposition
Alias:	EigenvalueDecomposition
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.matrix import EigenvalueDecomposition

Example

Example 1: A=Double2d([[1.,1.,1.],[1.,2.,3.],[1.,3.,6.]])

```
evd=A.apply(EigenvalueDecomposition())
print evd.getD()
print evd.getV()
```

API Summary

Jython Syntax
`A=Double2d()`
`evd=A.apply(EigenvalueDecomposition())`

Property
`Double2d A [INPUT, MANDATORY, default=no default value]`

API details

Property

`Double2d A [INPUT, MANDATORY, default=no default value]`

Input must be a Double2d or Float2d array.

3.92. EllipseHistogramExplorer

Full Name:	herschel.ia.gui.image.EllipseHistogramExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import EllipseHistogramExplorer

Description

An explorer for EllipseHistogramProducts.

API Summary

Constructors
`EllipseHistogramExplorer()`
The constructor of a new EllipseHistogramExplorer.
`EllipseHistogramExplorer(Object object)`
The constructor of a new EllipseHistogramExplorer associated

Methods
`String` **`getName()`**
Returns the name for this EllipseHistogramExplorer.
`String` **`getDescription()`**
Returns the description for this EllipseHistogramExplorer.
`boolean` **`canHandle(Class className)`**
Checks whether this EllipseHistogramExplorer can handle objects of the
`setObject(Object object)`
Sets the object for this EllipseHistogramExplorer to the given object.
`EllipseHistogramProduct` **`getObject()`**
Returns the object for this EllipseHistogramExplorer.
`Class` **`getVariableType()`**
Returns the expected variable type for this EllipseHistogramExplorer.
`JTable` **`getParameterTable()`**
Returns the parameter table for this

API details

Constructors

`EllipseHistogramExplorer()`

`EllipseHistogramExplorer(Object object)`

with the given object.

Argument

`Object` **`object`** `[INPUT, MANDATORY]`

Methods

String getName()

Return

String

Returns the name for this EllipseHistogramExplorer.

String getDescription()

Return

String

Returns the description for this EllipseHistogramExplorer.

boolean canHandle(Class className)

given class.

Argument

Class **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this EllipseHistogramExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

Object **object** [INPUT, MANDATORY]

EllipseHistogramProduct getObject()

Return

EllipseHistogramProduct

Returns the object for this EllipseHistogramExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this EllipseHistogramExplorer.

JTable getParameterTable()

EllipseHistogramExplorer.

Return

JTable

Returns the parameter table for this EllipseHistogramExplorer.

3.93. EllipseHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.EllipseHistogramPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import EllipseHistogramPanel

Description

A panel for the EllipseHistogramPanel.

API Summary

Constructor
EllipseHistogramPanel() The construction of a new EllipseHistogramPanel.

Methods
drawFigure() Draws the ellipse on the image associated with this
updateFigure() Updates the ellipse associated with this
trigger() Triggers the execution of the task associated
updateHistogram() Updates the histogram associated with this

API details

Constructor

EllipseHistogramPanel()

Methods

drawFigure()
EllipseHistogramPanel.

updateFigure()
EllipseHistogramPanel.

trigger()
with this EllipseHistogramPanel.

updateHistogram()
EllipseHistogramPanel.

3.94. EllipseHistogramProduct

Full Name:	herschel.ia.dataset.image.EllipseHistogramProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import EllipseHistogramProduct

Description

A class to deal with the results of an ellipse histogram.

API Summary

Constructor
`EllipseHistogramProduct()` The constructor of a new EllipseHistogramProduct.

Methods
`setCenter``(double centerX, double centerY)` Sets the center for this EllipseHistogramProduct
`setCenter``(double centerX, double centerY, String centerRA, String centerDec)` Sets the center for this EllipseHistogramProduct
`setDimensions``(double widthPixels, double heightPixels)` Sets the dimensions for this EllipseHistogramProduct to the
`setDimensions``(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)` Sets the dimensions for this EllipseHistogramProduct to the
`Double1d` **`getCenterPixelCoordinates`**`()` Returns the center for this EllipseHistogramProduct in
`String1d` **`getCenterSkyCoordinates`**`()` Returns the center for this EllipseHistogramProduct in
`double` **`getWidthPixels`**`()` Returns the width for this EllipseHistogramProduct in pixels.
`double` **`getWidthArcsec`**`()` Returns the width for this EllipseHistogramProduct in arcsec.
`double` **`getHeightPixels`**`()` Returns the height for this EllipseHistogramProduct in pixels.
`double` **`getHeightArcsec`**`()` Returns the height for this EllipseHistogramProduct in arcsec.

API details

Constructor

`EllipseHistogramProduct()`

Methods

setCenter(double centerX, double centerY)

to the given pixel coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

setCenter(double centerX, double centerY, String centerRA, String centerDec)

to the given pixel and sky coordinates.

Arguments

double **centerX** [INPUT, MANDATORY]

double **centerY** [INPUT, MANDATORY]

String **centerRA** [INPUT, MANDATORY]

String **centerDec** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels)

given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)

given dimensions in pixels and arcsec.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

double **widthArcsec** [INPUT, MANDATORY]

double **heightArcsec** [INPUT, MANDATORY]

Double1d getCenterPixelCoordinates()

pixel coordinates.

Return

Double1d

Returns the center for this EllipseHistogramProduct in pixel coordinates.

String1d getCenterSkyCoordinates()

sky coordinates.

Return

String1d

Returns the center for this EllipseHistogramProduct in sky coordinates.

`double getWidthPixels()`

Return

double

Returns the width for this EllipseHistogramProduct in pixels.

`double getWidthArcsec()`

Return

double

Returns the width for this EllipseHistogramProduct in arcsec.

`double getHeightPixels()`

Return

double

Returns the height for this EllipseHistogramProduct in pixels.

`double getHeightArcsec()`

Return

double

Returns the height for this EllipseHistogramProduct in arcsec.

3.95. EllipseHistogramTask

Full Name:	herschel.ia.toolbox.image.EllipseHistogramTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import EllipseHistogramTask

Description

A Task to make a histogram within an ellipse.

A Task to make a histogram of a region of interest, which is bounded by an ellipse.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`
`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`
`Double centerY [INPUT, OPTIONAL, default=Default value : NaN]`
`String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]`
`String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]`
`Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]`
`Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]`
`Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]`
`Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`

The lower cut level.

`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`

The upper cut level.

`Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`

The number of bins.

Double centerX [INPUT, OPTIONAL, default=Default value : NaN]

The x-pixel-coordinate of the center of the ellipse.

Double centerY [INPUT, OPTIONAL, default=Default value : NaN]

The y-pixel-coordinate of the center of the ellipse.

String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]

The right ascension of the center of the ellipse.

String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the center of the ellipse.

Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]

The width of the ellipse in pixels.

Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]

The height of the ellipse in pixels.

Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The width of the ellipse in arcsec.

Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The height of the ellipse in arcsec.

3.96. Erfc

Full Name:	herschel.ia.numeric.toolbox.basic.Erfc
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Erfc

Description

Computes the Complementary Error function.

Example

Example 1: Apply ERFC on a Double1d

```
x = Double1d([-5.0,-1.0,-0.5,0.0,0.5,1.0,5.0])
erc = ERFC(x)
print erc  # [1.9999999999984626,1.8427007900291827,1.520499876068384,

 1.0,0.4795001239316159,0.15729920997081737,1.5374597939680894E-12]
```

API Summary

Jython Syntax

```
<y>=ERFC(<x>)
```

Properties

```
an doulble array or number x [INPUT, MANDATORY, default=.]
```

```
returns an array (or a number if the input is a number) y [OUTPUT,
MANDATORY, default=no default value]
```

API details

Properties

an doulble array or number x [INPUT, MANDATORY, default=.]

returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]

where each element is the Complementary Error function of the corresponding element of the input array.

See also

- Erf

3.97. Erf

Full Name:	herschel.ia.numeric.toolbox.basic.Erf
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Erf

Description

Computes the Error function.

Example

Example 1: Apply ERF on a Double1d

```
x = Double1d([-5.0,-1.0,-0.5,0.0,0.5,1.0,5.0])
er = ERF(x)
print er  # [-0.9999999999984626,-0.8427007900291826,-0.5204998760683841,
            0.0,0.5204998760683841,0.8427007900291826,0.9999999999984626]
```

API Summary

Jython Syntax

```
<y>=ERF(<x>)
```

Properties

```
an double array or number x [INPUT, MANDATORY, default=.]
returns an array (or a number if the input is a number) y [OUTPUT,
MANDATORY, default=no default value]
```

API details

Properties

an double array or number x [INPUT, MANDATORY, default=.]

returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]

where each element is the Error function of the corresponding element of the input array.

See also

- Erfc

3.98. EXP10

Full Name:	herschel.ia.numeric.toolbox.basic.Exp10
Alias:	EXP10
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Exp10

Description

Gives 10 raised to the power of array x

Example

Example 1: Apply EXP10 on a Int1d

```
x=Double1d([1,2,3,4])
print EXP10(x) #  [10.0, 100.0, 1000.0, 10000.0]
```

API Summary

Jython Syntax

```
<y>=EXP10(<x>)
```

Properties

any number or array **x** [INPUT, MANDATORY, default=no default value]

a number or an array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any number or array x [INPUT, MANDATORY, default=no default value]

a number or an array y [OUTPUT, MANDATORY, default=no default value]

The result is a number or an array depending on the input

See also

- EXP10

3.99. EXP

Full Name:	herschel.ia.numeric.toolbox.basic.Exp
Alias:	EXP
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Exp

Description

Gives the exponential function applied to a number or array.

Example

Example 1: Apply EXP on a Int1d

```
x=Double1d([0,1])
print LOG(EXP(x)) #  [0.0,1.0]
```

API Summary

Jython Syntax
<y>=EXP(<x>)

Properties
any number or array **x** [INPUT, MANDATORY, default=no default value]
a number or an array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any number or array x [INPUT, MANDATORY, default=no default value]

a number or an array y [OUTPUT, MANDATORY, default=no default value]

The result is a number or an array depending on the input

See also

- LOG

3.100. ExpN

Full Name:	herschel.ia.numeric.toolbox.basic.ExpN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import ExpN

Description

Gives the specified base raised to each item array: y=ExpN base$^{(x)}$.

Example

Example 1: Apply ExpN on a Int1d

```
x=Double1d([1,2,3,4])
print ExpN(2)(x) # [2.0,4.0,8.0,16.0]
print x.apply(ExpN(2)) # [2.0,4.0,8.0,16.0]
```

API Summary

Jython Syntax

```
<y>=ExpN(<base>)(<x>)
```

Properties

any array or number **x** [INPUT, MANDATORY, default=no default value]
a number **base** [INPUT, MANDATORY, default=no default value]
a number or an array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

a number base [INPUT, MANDATORY, default=no default value]

The base n

a number or an array y [OUTPUT, MANDATORY, default=no default value]

The result is a number or an array depending on the input

See also

- ExpN

3.101. exportPalToUfDir

Full Name:	herschel.ia.toolbox.util.ExportPalToUfDirTask
Alias:	exportPalToUfDir
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import ExportPalToUfDirTask
Category:	task

Description

Exports observations related to an URN from a pool to a user friendly directory structure

The exportPalToUfDir task is used to export observations to a user friendly directory structure, so that they can be easily inspected, used and shared outside of HIPE.

Please note that only LocalStores and HsaReadPools can export their observations.

Example

Example 1: Simple example

```
exportPalToUfDir(pool=poolSrc ,
 urn="urn:poolid:herschel.ia.obs.ObservationContext:0", dirout="/exportdir1")
```

API Summary

Jython Syntax

```
exportPalToUfDir(<pool> , <urn>, <dirout> [, <warn>])
```

Properties

```
ProductPool pool [INPUT, MANDATORY, default=No default value]
String urn [INPUT, MANDATORY, default=No default value]
Boolean warn [INPUT, OPTIONAL, default=true]
String dirout [INPUT, MANDATORY, default=No default value]
```

Limitations

The dirout must not exist (cannot repeatedly export to the same dir).

Miscellaneous

No miscellaneous

API details

Properties

`ProductPool pool [INPUT, MANDATORY, default=No default value]`

The pool to export products from.

String urn [INPUT, MANDATORY, default=No default value]
The urn that defines what to export

Boolean warn [INPUT, OPTIONAL, default=true]
If true, asks confirmation if the directory already exists and it is not empty.

String dirout [INPUT, MANDATORY, default=No default value]
The directory where the observations will be exported with a user friendly structure

See also

- ???

History

- 16-01-09 Created

3.102. ExtractFreqRanges

Full Name:	herschel.ia.toolbox.spectrum.ExtractFreqRanges
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import ExtractFreqRanges

Description

Task for extracting a range or ranges from the segments of one or several `SpectrumContainer`.

Different possibilities are available for specifying the ranges (see the `ranges`-parameter below. Specifying frequency intervals means that all the spectra are tested for the suitable ranges. In presence of frequency drift these ranges may be different from spectrum to spectrum. For the spectrum container the enveloping range is taken so that the all the PointSpectra included in the result container have again the same width of the segments. Here, the number of frequency intervals and the number of segments does not need to coincide. For each of the segments with a non-trivial overlap with one of the frequency intervals will lead to a segment in the result container.

Example

Example 1: from Jide

```
slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900),
(6100,6900),(7100,7900)])
slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900),
(6100,6900),(7100,7900)], selection={"bbtype":[6031]})
slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900),
(6100,6900),(7100,7900)], selection={"Chopper":([-4.4,5.9],0.2)})
slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900),
(6100,6900),(7100,7900)], selection=[0,1,2,3])
slices = extract(ds=spectra, ranges=[(4100,4400),(4600,4900),(5100,5900),
(6100,6900),(7100,7900)], selection={"LoFrequency":(4000.0,5000.0)})
slices = extract(ds=spectra, ranges=([4100,4600,5100,6100,7100],
[4400,4900,5900,6900,7900]))
# or in "the old way":
min = Double1d([4100,4600,5100,6100,7100])
max = Double1d([4400,4900,5900,6900,7900])
slices = extract(ds=spectra, min=min, max=max)
```

API Summary

Properties
Object **ds** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=no default value.]
Object **ranges** [INPUT, OPTIONAL, default=no default value.]
Double1d **minFreq** [INPUT, OPTIONAL, default=no default value.]
Double1d **maxFreq** [INPUT, OPTIONAL, default=no default value.]
Object **segments** [INPUT, OPTIONAL, default=no default value.]
Object **selection** [INPUT, OPTIONAL, default=None.]

API details

Properties

Object ds [INPUT, OPTIONAL, default=no default value.]

Input data to be processed by the task. Several types are possible:

- SpectrumContainer
- Array of SpectrumContainer, i.e. SpectrumContainer[]
- List of SpectrumContainer, i.e. List
- Any product with implementations of `SpectrumContainer` inside.

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value.]

The resulting container created by this task.

Object ranges [INPUT, OPTIONAL, default=no default value.]

Specify the range to be extracted - either by specifying ranges of channel numbers or frequency ranges:

- Type Range[]: Range of channel numbers for each sub-segment to be extracted.
- Type Range: Range of channel numbers to be extracted (if only one segment in the data).
- Type PyList or PyTuples: Each PyTuple defines an interval on the frequency scale.
- Type PyTuple of two PyLists: The first list with the minima and the second with the maxima of the intervals on the frequency scale.

Double1d minFreq [INPUT, OPTIONAL, default=no default value.]

For each of the ranges to be extracted the minimum frequency.

Double1d maxFreq [INPUT, OPTIONAL, default=no default value.]

For each of the ranges to be extracted the maximum frequency.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments to restrict on. There are two options available:

- Specify a PyList of segment indices or
- pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.

Object selection [INPUT, OPTIONAL, default=None.]

Specify what point spectra to restricted to. Different ways to specify these selections are possible:

Object selection [INPUT, OPTIONAL, default=None.]
• Specify a list of indices (in jython) of the point spectra for which the processing should be applied. • Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals). • Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above. • Pass any java instance that implements the SelectionModel interface (for the advanced user).

See also

- ManyToOneSpectrumTask

History

- `2008-01-29 - meli`: initial.
- `2008-03-16 - meli`: major refactoring of the toolbox.
- `2009-05-18 - meli`: more ways to specify selections and frequency ranges.

3.103. ExtractMultiplePixelSpectrumPanel

Full Name:	herschel.ia.gui.cube.ExtractMultiplePixelSpectrumPanel
Type:	Java Class -
Import:	from herschel.ia.gui.cube import ExtractMultiplePixelSpectrumPanel

Description

A panel for the SinglePixelSpectrum extraction task

API Summary

Constructor
`ExtractMultiplePixelSpectrumPanel()` The constructor of a new ExtractSinglePixelSpectrumPanel.

Methods
`JPanel` **`getPixelsPanel()`** Returns the panel where to specify the radii for the
`drawFigures()` Draws the circles on the image for this
`moveConfirmedFigures()` Allows the user to move the circles bounding the
`clearSpectrumExtraction()` Clears the annular sky aperture for this
`setClicked``(Double center)` Sets the center of the rectancle, associated with this

API details

Constructor

`ExtractMultiplePixelSpectrumPanel()`

Methods

`JPanel getPixelsPanel()`
apertures for this ExtractSinglePixelSpectrumPanel. **Return** **JPanel** Returns the panel where to specify the radii for the apertures for this AnnularSkyAoerturePhotometryPanel.

`drawFigures()`
ExtractSinglePixelSpectrumPanel.

moveConfirmedFigures()

apertures for this ExtractSinglePixelSpectrumPanel.

clearSpectrumExtraction()

ExtractSinglePixelSpectrumPanel.

setClicked(Double center)

ExtractSinglePixelSpectrumPanel to the given point (in UserCoordinates).

Argument

`Double` **`center`** `[INPUT, MANDATORY]`

3.104. ExtractRegionPixelSpectrumTask

Full Name:	herschel.ia.toolbox.cube.ExtractRegionPixelSpectrumTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import ExtractRegionPixelSpectrumTask

Description

Task which Extract an averaged Spectrum from a set of pixels,

from a simplecube.

In the CubeSpectrumAnalysisToolbox (via the SpectrumAvgPlotting GUI), this task process a set of contiguous pixels and can be used in association with some FILTERRING, the output of this task in this kind of usage is therefore sent directly to the next tasks.

The command line can be used to retrieve manually some spectrum or to include it in scripts. this task use SimpleCube and return

Example

Example 1: How to use the Single pixel spectrum extraction in jide

```
singlePixExtraction = ExtractSinglePixelSpectrumTask();
singlePixExtraction.setValue("simplecube",Cube)
singlePixExtraction.setValue("wholeImg",False)
singlePixExtraction.setValue("posArray",arrayofpixel)
singlePixExtraction.execute()
spectrum =(Double1d) singlePixExtraction.getValue("spectrum")
spectrum 1d spectrum1d =(Double1d) singlePixExtraction.getValue("spectrum")
 {@link #__abs__()} futur
```

API Summary

Properties
SimpleCube **simplecube** [INPUT, MANDATORY, default=no default value]
Boolean **wholeImg** [INPUT, no default value, default=no default value]
Double2d **posArray** [INPUT, MANDATORY, default=no default value]
Integer **nbPixels** [INPUT, MANDATORY, default=no default value]
Integer **arithmetics** [INPUT, MANDATORY, default=no default value]
Double1d **spectrum** [OUTPUT, MANDATORY, default=no default value]
Spectrum1d **finalspectrum** [OUTPUT, MANDATORY, default=no default value]
float **totalWeight** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

SimpleCube simplecube [INPUT, MANDATORY, default=no default value]

The Cube containing the spectra to extract

Boolean wholeImg [INPUT, no default value, default=no default value]

A flag to define the usage whole image or region of this task

Double2d posArray [INPUT, MANDATORY, default=no default value]

The Array of pixels to be read in the cube must contain the X,Y and weight information [X,Y,weight]

Integer nbPixels [INPUT, MANDATORY, default=no default value]

The number of pixel to be read

Integer arithmetics [INPUT, MANDATORY, default=no default value]

a value defining the kind of arithmetic to use for the resulting spectra: choice between average(1) median(2) sum(3) default = sum

Double1d spectrum [OUTPUT, MANDATORY, default=no default value]

The flux of the spectrum extracted at the given position

Spectrum1d finalspectrum [OUTPUT, MANDATORY, default=no default value]

The spectrum extracted at the given position in a Spectrum1d

float totalWeight [OUTPUT, MANDATORY, default=no default value]

the 'weight' of the spectrum , ne exact number of pixel from which the spectrum was read

See also

- ???
- ???
- herschel.ia.dataset.spectrum.Spectrum1d;

History

- 2008-06-17 - AG: first complet description
- 2008-07-03 - AG: Modification of the input Parameters
- 2008-09-08 - AG: add the error computation
- 2009-06-08

3.105. ExtractSinglePixelSpectrumPanel

Full Name:	herschel.ia.gui.cube.ExtractSinglePixelSpectrumPanel
Type:	Java Class -
Import:	from herschel.ia.gui.cube import ExtractSinglePixelSpectrumPanel

Description

A panel for the SinglePixelSpectrum extraction task

API Summary

Constructor
ExtractSinglePixelSpectrumPanel() The constructor of a new ExtractSinglePixelSpectrumPanel.

Methods
JPanel **getPixelsPanel()** Returns the panel where to specify the radii for the
drawFigures() Draws the circles on the image for this
moveConfirmedFigures() Allows the user to move the circles bounding the
clearSpectrumExtraction() Clears the annular sky aperture for this
setClicked(Double center) Sets the center of the rectancle, associated with this

API details

Constructor

ExtractSinglePixelSpectrumPanel()

Methods

JPanel getPixelsPanel()

apertures for this ExtractSinglePixelSpectrumPanel.

Return

JPanel

Returns the panel where to specify the radii for the apertures for this AnnularSkyAoerturePhotometryPanel.

drawFigures()

ExtractSinglePixelSpectrumPanel.

moveConfirmedFigures()

apertures for this ExtractSinglePixelSpectrumPanel.

clearSpectrumExtraction()

ExtractSinglePixelSpectrumPanel.

setClicked(Double center)

ExtractSinglePixelSpectrumPanel to the given point (in UserCoordinates).

Argument

`Double` **`center`** `[INPUT, MANDATORY]`

3.106. ExtractSinglePixelSpectrumTask

Full Name:	herschel.ia.toolbox.cube.ExtractSinglePixelSpectrumTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import ExtractSinglePixelSpectrumTask

Description

Task which extract a Spectrum of a Simplecube from a Single pixel position given as

parameter.

This task is used in the GUI CubeSpectrumAnalysisToolbox for the Real time single spectrum visualisation and in the the Single pixel Spectrum Extraction.

*

This task can be used in common with some filtering and fitting tasks. in this case the input parameters for these tasks are partly the result of this one. In the main GUI or in the JIDE the user will decide and define of the filter to apply.

Example

Example 1: How to use the Single pixel spectrum extraction

```
singlePixExtraction = ExtractSinglePixelSpectrumTask();
singlePixExtraction.setValue()
```

API Summary

Properties
SimpleCube **simplecube** [INPUT, MANDATORY, default=no default value]
Integer **posX** [INPUT, MANDATORY, default=no default value]
Integer **posY** [INPUT, MANDATORY, default=no default value]
Double1d **spectrum** [OUTUT, MANDATORY, default=no default value]
Spectrum1d **spectrum** [OUTUT, MANDATORY, default=no default value]

API details

Properties

SimpleCube simplecube [INPUT, MANDATORY, default=no default value]

The Cube containing the spectra

Integer posX [INPUT, MANDATORY, default=no default value]

The X coordinate of the pixel to bew read.

Integer posY [INPUT, MANDATORY, default=no default value]

The Y coordinate of the pixel to bew read.

Double1d spectrum [OUTUT, MANDATORY, default=no default value]
The spectrum extracted at the given position

Spectrum1d spectrum [OUTUT, MANDATORY, default=no default value]
The spectrum extracted at the given position in a Spectrum1d , will become the only output

History

- 2008-06-17 - AG: first complet description
- 2008-07-03 - AG: Modification of the input Parameters
- 2008-07-24 - AG: added a new output parameters
- 2008-10-3 - AG: temporary removed the unit management
- 2009-01-31 - AG: inverted the output parameters order

3.107. FFT2dTask

Full Name:	herschel.ia.toolbox.image.FFT2dTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import FFT2dTask

Description

A Task for two dimensional Fast Fourier Transforms.

A Task that calculates the 2D Fast Fourier Transform and the power spectrum.

API Summary

Properties
`Numeric2dData image [INPUT, MANDATORY, default=No default value]`
`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`
`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Numeric2dData image [INPUT, MANDATORY, default=No default value]`

The input image.

`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`

The transform.

`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

The spectrum.

3.108. FFT

Full Name:	herschel.ia.numeric.toolbox.xform.FFT
Alias:	FFT
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.xform import FFT

Description

Gives the Fast Fourier Transform <![CDATA[]]>.

Example

Example 1: Apply FFT

```
from java.lang.Math import PI
# Frequency modulated signal: parameters
ts = 1E-6          # Sampling period (sec)
fc = 200000        # Carrier frequency (Hz)
fm = 2000          # Modulation frequency (Hz)
beta = .0003       # Modulation index (Hz)
n = 5000           # Number of samples
# Create signal in complex form
t = Double1d.range(n) * ts
signal = SIN(2 * PI * fc * t * (1 + beta * COS(2 * PI * fm * t)))
z_signal=Complex1d(signal)
# Get spectrum
spectrum = ABS(FFT(z_signal))
# Repeat with apodizing
z_signal=Complex1d(HAMMING(signal))
spectrum2 = ABS(FFT(z_signal))
```

API Summary

Jython Syntax

```
<y>=FFT(<x>)
```

Properties

```
Complex1d x [INPUT, MANDATORY, default=no default value]
Double1d y [INPUT, MANDATORY, default=no default value]
```

API details

Properties

`Complex1d x [INPUT, MANDATORY, default=no default value]`

Complex1d arrays only.

`Double1d y [INPUT, MANDATORY, default=no default value]`

Returns a Double1d

See also

- IFFT

3.109. FitsArchive

Full Name:	herschel.ia.io.fits.FitsArchive
Alias:	FitsArchive
Type:	Java Class -
Import:	from herschel.ia.io.fits import FitsArchive
Category:	class

Description

A class for reading and writing FITS files.

The FitsArchive provides a transparant way to store and retrieve Products as defined within the Herschel Data Processing software.

Examples

Example 1: default usage:

```
from herschel.ia.io.fits import FitsArchive
# read/write a Product in HCSS decorated format.
fits=FitsArchive()
product=fits.load("input.fits")
fits.save("output.fits",product)
```

Example 2: reading a HCSS FITS file which contains a removed class:

```
fits=FitsArchive()
product=fits.load("input.fits",FitsArchive.HANDLE_MISSING_CLASSES)
```

Example 3: reading a externally generated FITS file into a Product:

```
from herschel.ia.io.fits import FitsArchive
# setup a new FitsArchive, but change the reader type
from herschel.ia.io.fits.reader.standard import StandardFitsReader
fits = FitsArchive(reader = StandardFitsReader())
product=fits.load("input.fits")
# Saving the product will be done in HCSS decorated format.
fits.save("output.fits",product)
```

Example 4: saving an empty ArrayDataset is allowed:

```
from herschel.ia.io.fits import FitsArchive
fits=FitsArchive()
ads=ArrayDataset()
product=Product("with empty ArrayDataset")
product['ads'] = ads
fits.save("product.fits", product) # works: image extension without image
```

Example 5: saving an empty column in a TableDataset is NOT allowed:

```
from herschel.ia.io.fits import FitsArchive
fits=FitsArchive()
tds=TableDataset()
tds.addColumn(Column())
product=Product("with empty Column in TableDataset")
product['tds'] = tds
```

Example 5: saving an empty column in a TableDataset is NOT allowed:

```
fits.save("product.fits", product) # fails: binary table extensions require
 data
```

Example 6: saving the product minimizing product metadata keyword translations and making keywords duplicity.:

```
from herschel.ia.io.fits import FitsArchive
fits=FitsArchive()
product=Product("My Product")
fits.save("product.fits", product, 1, 1)
```

API Summary

Fields
`FitsReader HCSS_READER` HCSS FITS Reader
`FitsReader STANDARD_READER` Generic FITS Reader

Methods
`Product load(String name)` Loads a Product from a FITS file.
`Product load(String name, boolean handleMissingClasses)` Loads a Product from a FITS file.
`Product load(InputStream is)` Loads a Product from a FITS InputStream.
`Product load(InputStream is, boolean handleMissingClasses)` Loads a Product from a FITS InputStream.
`save(String name, [Optionally derived] Product. product)` Saves a Product to a FITS file.

API details

Fields

`FitsReader` HCSS_READER

A FITS reader that expects a FITS format that was produced by this archive implementation. This is a shared object among any FitsArchive instance. In a multithread environment, you should use a new instance of a reader.

It can handle nested structures, derived datasets and quantities of the data within FITS.

This is the default reader when you create a FitsArchive.

Examples

creating a new FitsArchive

```
fits=FitsArchive() # default reader=FitsArchive.HCSS_READER
```

FitsReader HCSS_READER

creating a new FitsArchive

```
fits=FitsArchive(reader=FitsArchive.HCSS_READER)
```

FitsReader STANDARD_READER

A generic FITS reader for FITS files that are not created by the HCSS FitsArchive. This is a shared object among any FitsArchive instance. In a multithread environment, you should use a new instance of a reader.

This reader can read FITS files that were generated by other software as long as it complies to the FITS standard and translates the contents into a Product.

Examples

import data from an externally generated FITS file:

```
fits=FitsArchive(reader=FitsArchive.STANDARD_READER)
product=fits.load("external.fits")
```

reusing this FitsArchive but changing the reader:

```
fits.reader=fits.HCSS_READER
product=fits.load("hcss.fits")
```

Methods

Product load(String name)

Loads a Product from a FITS file using the current reader.

Argument

`String` **name** `[INPUT, MANDATORY, default=no default value]`

Name of the FITS file

Return

Product

An object of the herschel.ia.dataset.Product family.

Product load(String name, boolean handleMissingClasses)

Loads a Product from a FITS file using the current reader.

Arguments

`String` **name** `[INPUT, MANDATORY, default=no default value]`

Name of the FITS file

`boolean` **handleMissingClasses** `[INPUT, MANDATORY, default=no default value]`

For creating dummy classes when a Hcss class is not found.

Return

Product

An object of the herschel.ia.dataset.Product family.

Product load(InputStream is)

Loads a Product from a FITS InputStream using the current reader.

`Product load(InputStream is)`

Argument

`InputStream` **`is`** `[INPUT, MANDATORY, default=no default value]`

InputStream to the FITS file

Return

Product

An object of the herschel.ia.dataset.Product family.

`Product load(InputStream is, boolean handleMissingClasses)`

Loads a Product from a FITS InputStream using the current reader.

Arguments

`InputStream` **`is`** `[INPUT, MANDATORY, default=no default value]`

InputStream to the FITS file

`boolean` **`handleMissingClasses`** `[INPUT, MANDATORY, default=no default value]`

For creating dummy classes when a Hcss class is not found.

Return

Product

An object of the herschel.ia.dataset.Product family.

`save(String name, [Optionally derived] Product. product)`

Saves a Product to a FITS file with sufficient format informat to preserve the quantities of the data as well as the original dataset type and contents.

Arguments

`String` **`name`** `[INPUT, MANDATORY, default=no default value]`

Name of the FITS file

`[Optionally derived] Product.` **`product`** `[INPUT, MANDATORY, default=no default value]`

An object of the herschel.ia.dataset.Product family.

3.110. FitterFunction

Full Name:	herschel.ia.toolbox.fit.FitterFunction
Alias:	FitterFunction
Type:	Java Class -
Import:	from herschel.ia.toolbox.fit import FitterFunction

Description

Specialization of RealFunction that fits some given data.

Example

Example 1: Implicit fitting

```
x = Double1d([ 0,0.5,1,2, 3, 4, 5,  6,  10,    12])
y = Double1d([-3,  2,4,5,12,37,92,189,1237,2325])
f = FitterFunction(x, y, PolynomialModel(3))
i = SimpsonIntegrator(1, 10)
print i.integrate(f)  # 2668.499999999972
Explicit fitting
x = Double1d([ 0,0.5,1,2, 3, 4, 5,  6,  10,    12])
y = Double1d([-3,  2,4,5,12,37,92,189,1237,2325])
model = CubicSplinesModel(x)
fitter = AmoebaFitter(x, model)
fitter.setSimplex(params, range)  # customize the fitter as you want
fitter.fit(y)
f = FitterFunction(model)  # or f = FitterFunction(fitter)
# remaining code like before
```

API Summary

Jython Syntax

```
<f>=FitterFunction(<x>,<y>,<m>[,<c>])
<f>=FitterFunction(<m>)
<f>=FitterFunction(<F>)
```

Properties

```
Double1d of abscissas x [INPUT, MANDATORY, default=no default value]
Double1d of values y [INPUT, MANDATORY, default=no default value]
AbstractModel m [INPUT, MANDATORY, default=no default value]
Class of Fitter c [INPUT, OPTIONAL, default=Fitter.class]
Fitter F [INPUT, MANDATORY, default=no default value]
RealFunction f [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

Double1d of abscissas x [INPUT, MANDATORY, default=no default value]

Abscissas; it is only mandatory for the constructors where it appears in the synopsis.

Double1d of values y [INPUT, MANDATORY, default=no default value]

Values corresponding to the abscissas; it is only mandatory for 1st constructors.

AbstractModel m [INPUT, MANDATORY, default=no default value]

Unitialized model for 1st constructors, or already customized model for 2nd constructor.

Class of Fitter c [INPUT, OPTIONAL, default=Fitter.class]

Allows specifying the fitter class to be used in 1st constructor. FitterFunction provides some useful constants: LINEAR (for Fitter.class), AMOEBA (for AmoebaFitter.class) and LEVENBERG (for LevenbergMarquardtFitter.class).

Fitter F [INPUT, MANDATORY, default=no default value]

It is only mandatory for 3rd constructor.

RealFunction f [OUTPUT, MANDATORY, default=no default value]

Function object that fits the provided data with the specified model.

3.111. Fitter

Full Name:	herschel.ia.numeric.toolbox.fit.Fitter
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.fit import Fitter

Description

An introduction to the use of fit package can be found

here. At least once have a look at the background on the organization and structure of the package.

Example

Example 1: The fit/demo directory contains worked

```
<a href="../../../ia/numeric/toolbox/fit/demo/jython.html">examples</a>
on almost all aspects of of the package.
```

Limitations

The calculation of the evidence is an Gaussian approximation which is only exact for linear models with a fixed scale.

3.112. FitterTask

Full Name:	herschel.ia.toolbox.fit.FitterTask
Alias:	FitterTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.fit import FitterTask
Category:	generic task

Description

generic module: FitterTask

Task shell around the package herschel.ia.numeric.toolbox.fit. Both a command line and a GUI interface are provided. With the GUI interface there is more of the Fitter package accessable than on this tasks command line. However not everything possible within the fitter package is accessable via this task, GUI or otherwise. Most text fields in the GUI accept either numerics or JIDE variables or JIDE expressions.

Example

Example 1: FitterTask

```
<pre>
from herschel.hifi.generic.task import FitterTask
# Assume that `tt' and/or `data' are Double1d's
# usage on command line:
ft = FitterTask()( x=tt, y=data, modelname="GaussModel" )
ft.fittername = "LevenbergMarquardtFitter"
ft()            # performs the execute method
print ft.result        # parameters of the fit
print ft.stdev        # standard deviations
print ft.fittername    # name of the fitter
# Etcetera, etc.
# use the GUI.
ft = FitterTask()
ft.gui = 1         # start the GUI
# from the GUI select the data, model, fitter, properties etc and run.
print ft.result        # parameters of the fit
print ft.stdev        # standard deviations
# Etcetera as before.
</pre>
```

API Summary

Jython Syntax

```
see example below
```

Properties

```
NumericData x [INPUT, OPTIONAL, default=null]
DoubleArray y [INPUT, MANDATORY, default=no]
Boolean indexview [INPUT, OPTIONAL, default=true]
DoubleArray weights [INPUT, OPTIONAL, default=null]
AbstractModel model [INPUT, OPTIONAL, default=null]
String modelname [INPUT, OPTIONAL, default=no]
```

Properties
Array1dData **modelarg** [INPUT, OPTIONAL, default=null]
Fitter **fitter** [INPUT, OPTIONAL, default=null]
String **fittername** [INPUT, OPTIONAL, default=no]
Double1d **result** [OUTPUT, OPTIONAL, default=n/a]
Double1d **parameters** [OUTPUT, OPTIONAL, default=n/a]
Double1d **stdev** [OUTPUT, OPTIONAL, default=n/a]
Double **chisq** [OUTPUT, OPTIONAL, default=n/a]
DoubleArray **yfit** [OUTPUT, OPTIONAL, default=n/a]
DoubleArray **yband** [OUTPUT, OPTIONAL, default=n/a]
Double1d **prior** [INPUT, OPTIONAL, default=null]
Double **evidence** [OUTPUT, OPTIONAL, default=n/a]
Double **scale** [OUTPUT, OPTIONAL, default=n/a]
Boolean **auto** [INPUT, OPTIONAL, default=true]
Double **fixed** [INPUT, OPTIONAL, default=1.0]
Double **mixed** [INPUT, OPTIONAL, default=0.0]
Double **tolerance** [INPUT, OPTIONAL, default=0.01]
Integer **iterations** [INPUT, OPTIONAL, default=10000]
Double **temperature** [INPUT, OPTIONAL, default=0.0]
Double **cooling** [INPUT, OPTIONAL, default=0.95]
Integer **tempsteps** [INPUT, OPTIONAL, default=100]
Double1d **initialpars** [INPUT, OPTIONAL, default=from model]
Double1d **highlimits** [INPUT, OPTIONAL, default=null]
Double1d **lowlimits** [INPUT, OPTIONAL, default=null]
Int1d **keepfixed** [INPUT, OPTIONAL, default=null]
Double1d **fixedvalues** [INPUT, OPTIONAL, default=null]
Boolean **gui** [INPUT, OPTIONAL, default=false]

Limitations

Not everything which is possible using the package directly is accessable via this task.

API details

Properties

NumericData x [INPUT, OPTIONAL, default=null]

The independent variable(s) of the fit problem. If not provided, it will use indices ranging from 0 to len(y). If y is more-dimensional the proper 2-d indices are provided.

DoubleArray y [INPUT, MANDATORY, default=no]

The data to be fitted. It can be more-dimensional. E.g. a 2-dimensional map.

Boolean indexview [INPUT, OPTIONAL, default=true]

Applicable parameters appear in the same order as in the y DoubleArray This parameter is only active when y has 2 dimensions or more.

DoubleArray weights [INPUT, OPTIONAL, default=null]

The weights to be used in the fit. Can be more-dimensional. When provided it should be the same size as parameter y.

AbstractModel model [INPUT, OPTIONAL, default=null]

The model to be fitted. One of "model" or "modelname" is MANDATORY

String modelname [INPUT, OPTIONAL, default=no]

The name of the model to be fitted. One of "model" or "modelname" is MANDATORY

Array1dData modelarg [INPUT, OPTIONAL, default=null]

Arguments, if any, needed in the Constructor of the model.

Fitter fitter [INPUT, OPTIONAL, default=null]

The fitter to be used. One of "fitter" or "fittername" is MANDATORY

String fittername [INPUT, OPTIONAL, default=no]

The name of the fitter to be used. One of "fitter" or "fittername" is MANDATORY

Double1d result [OUTPUT, OPTIONAL, default=n/a]

The parameters of the fit. Also the default result of the task.

Double1d parameters [OUTPUT, OPTIONAL, default=n/a]

The parameters of the fit. (same as result)

Double1d stdev [OUTPUT, OPTIONAL, default=n/a]

The standard deviations pertaining to the parameters.

Double chisq [OUTPUT, OPTIONAL, default=n/a]

Chi-squared of the fit.

DoubleArray yfit [OUTPUT, OPTIONAL, default=n/a]

An array of the same size as the data, containing the fitted values.

DoubleArray yband [OUTPUT, OPTIONAL, default=n/a]

An array of the same size as the data, containing a 1-# MonteCarlo error at each point.

Double1d prior [INPUT, OPTIONAL, default=null]

Prior ranges for the parameters, needed when the evidence is requested. When noise scaling is set to auto=true (default), the prior needs 1 extra value as prior for the noise scale.

Double evidence [OUTPUT, OPTIONAL, default=n/a]

Relative evidence the model carries w.r.t. the data It needs values for prior, as many as the number of parameters and one more if auto-scaling is set.

Double scale [OUTPUT, OPTIONAL, default=n/a]

Noise scale of the data (when auto or mixed is selected)

Boolean auto [INPUT, OPTIONAL, default=true]

Select automatic noise scaling. The noise level in the data is not exactly known.

Double fixed [INPUT, OPTIONAL, default=1.0]

Fixed noise scale. The noise level is known.

Double mixed [INPUT, OPTIONAL, default=0.0]

Automatic noise scaling with a minimum. The noise level is not exactly known, but a minimum level is known.

Double tolerance [INPUT, OPTIONAL, default=0.01]

Stopping criterion for iterative fitting.

Integer iterations [INPUT, OPTIONAL, default=10000]

Maximum number of iterations in iterative fitters.

Double temperature [INPUT, OPTIONAL, default=0.0]

Starting temperature in annealing amoeba fitting.

Double cooling [INPUT, OPTIONAL, default=0.95]

Cooling step in annealing amoeba fitting.

Integer tempsteps [INPUT, OPTIONAL, default=100]

Number of exploratory steps at each temperature.

Double1d initialpars [INPUT, OPTIONAL, default=from model]

Initial parameters in iterative fitters.

Double1d highlimits [INPUT, OPTIONAL, default=null]

Upper limits of the parameters (only in AmoebaFitter)

Double1d lowlimits [INPUT, OPTIONAL, default=null]

Lower limits of the parameters (only in AmoebaFitter)

Int1d keepfixed [INPUT, OPTIONAL, default=null]

Keep these parameters fixed. (Parameter numbering starts at 0)

Double1d fixedvalues [INPUT, OPTIONAL, default=null]

Values at which the parameters should be kept.

Boolean gui [INPUT, OPTIONAL, default=false]

Allows to handle this task using a gui.

History

- 15-10-2006 DK

3.113. FixedMask

Full Name:	herschel.ia.numeric.toolbox.mask.FixedMask
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.mask import FixedMask

Description

Class for mask handling, where each mask element is fixed at a specific bit offset position.

The implementation is therefore limited to 64 different definitions of a mask element, corresponding to each bit of a long integer. However, all integer types are supported (byte/short/int/long). Note that the mask data itself is not stored in this class.

Example

Example 1: Define some mask bits and use them for mask manipulations.

```
# Create the mask objects
MASTER = FixedMask (0, "Master")
NOISY  = FixedMask (2, "Noisy")
GLITCH = FixedMask (4, "Glitch")
# Masks can be printed:
print MASTER
# Master @ 0 = 1
print GLITCH
# Glitch @ 4 = 16
#
# They can be set:
x = 0
x = GLITCH.set (x)
x = MASTER.set (x)
print x
# 17
# and unset:
x = GLITCH.unset (x)
print x
# 1
# and tested
print MASTER.isSet(x)
# 1
print GLITCH.isSet(x)
# 0
#
# You can also use them with the numeric library:
a = Int1d.range(10)
print a.where (MASTER)
# [1,3,5,7,9]
print a.where (MASTER | NOISY)
# [1,3,4,5,6,7,9]
print MASTER.isSet (a)
# [false,true,false,true,false,true,false,true,false,true]
print MASTER.set (a)
# [1,1,3,3,5,5,7,7,9,9]
print MASTER.unset (a)
# [0,0,2,2,4,4,6,6,8,8]
```

3.114. FixedSkyAperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.FixedSkyAperturePhotometryExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import FixedSkyAperturePhotometryExplorer

Description

An explorer for FixedSkyAperturePhotometryProducts.

API Summary

Constructors
`FixedSkyAperturePhotometryExplorer()` The constructor of a new FixedSkyAperturePhotometryExplorer.
`FixedSkyAperturePhotometryExplorer(Object object)` The constructor of a new FixedSkyAperturePhotometryExplorer associated with

Methods
`boolean` **`canHandle`**`(Class className)` Checks whether this FixedSkyAperturePhotometryExplorer can
`setObject``(Object object)` Sets the object for this AperturePhotometryExplorer to the given object.
`FixedSkyAperturePhotometryProduct` **`getObject`**`()` Returns the object for this FixedSkyAperturePhotometryExplorer.
`Class` **`getVariableType`**`()` Returns the expected variable type for this FixedSkyAperturePhotometryExplorer.
`JTable` **`getParameterTable`**`()` Constructs and returns the parameter table for this
`JTable` **`getResultsTable`**`()` Constructs and returns the results table for this AperturePhotometryExplorer.
`JPanel` **`getPlotPanel`**`()` Constructs and returns the plot panel for this FixedSkyAperturePhotometryExplorer.

API details

Constructors

`FixedSkyAperturePhotometryExplorer()`

`FixedSkyAperturePhotometryExplorer(Object object)`

the given object.

Argument

`Object` **`object`** `[INPUT, MANDATORY]`

Methods

boolean canHandle(Class className)

handle objects of the given class.

Argument

`Class` **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this FixedSkyAperturePhotometryExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

`Object` **object** [INPUT, MANDATORY]

FixedSkyAperturePhotometryProduct getObject()

Return

FixedSkyAperturePhotometryProduct

Returns the object for this FixedSkyAperturePhotometryExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this FixedSkyAperturePhotometryExplorer.

JTable getParameterTable()

FixedSkyAperturePhotometryExplorer.

Return

JTable

Returns the parameter table for this FixedSkyAperturePhotometryExplorer.

JTable getResultsTable()

Return

JTable

Returns the results table for this AperturePhotometryExplorer.

JPanel getPlotPanel()

Return

JPanel

Returns the plot panel for this FixedSkyAperturePhotometryExplorer.

3.115. FixedSkyAperturePhotometryPanel

Full Name:	herschel.ia.toolbox.image.gui.FixedSkyAperturePhotometryPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import FixedSkyAperturePhotometryPanel

Description

A panel for the FixedSkyAperturePhotometryTask.

API Summary

Constructor
FixedSkyAperturePhotometryPanel() The constructor of a new FixedSkyAperturePhotometryPanel.

Methods
JPanel **getAperturesPanel()** Returns the panel where to specify the radii for the
JPanel **getTargetAperturePanel()** Returns the panel where to specify the target radius
JComponent **getSkyApertureComponent()** Returns the component where to specify the inner and outer
drawFigures() Draws the circle on the image for this
moveConfirmedFigures() Allows to move/resize the figures bounding the apertures
clearSkyAperture() Clears the sky aperture for this FixedAperturePhotometryPanel.

API details

Constructor

FixedSkyAperturePhotometryPanel()

Methods

JPanel getAperturesPanel()
apertures for this AnnularSkyAperturePhotometryPanel. **Return** **JPanel** Returns the panel where to specify the radii for the apertures for this AnnularSkyAoerturePhotometryPanel.

JPanel getTargetAperturePanel()

for this FixedAperturePhotometryPanel.

Return

JPanel

Returns the panel where to specify the target radius for this FixedAperturePhotometryPanel.

JComponent getSkyApertureComponent()

radius of the annular sky aperture for this AnnularSkyAperturePhotometryPanel.

Return

JComponent

Returns the component where to specify the inner and outer radius of the annular sky aperture for this AnnularSkyAperturePhotometryPanel.

drawFigures()

FixedSkyAperturePhotometryPanel.

moveConfirmedFigures()

for this FixedAperturePhotometryPanel.

clearSkyAperture()

3.116. FixedSkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.FixedSkyAperturePhotometryProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import FixedSkyAperturePhotometryProduct

Description

A class to deal with the results of aperture photometry with a circular

target aperture and a fixed sky value.

API Summary

Constructor
`FixedSkyAperturePhotometryProduct()` The constructor of a new FixedSkyAperturePhotometryProduct.

Methods
`setSkyValue(double sky)` Sets the sky value for this FixedSkyAperturePhotometryProduct to the
`setResultsTable(Double2d resultsTable)` Sets the results table for this AperturePhotometryProduct to the
`double` **`getSkyValue()`** Returns the sky value for this FixedSkyAperturePhotometryProduct.
`double` **`getIntensityPerSkyPixel()`** Returns the intensity per pixel for the sky for this FixedSkyAperturePhotometryProduct.
`double` **`getTargetTotal()`** Returns the total flux for the target (sky subtracted) for this
`double` **`getNbOfTargetPixels()`** Returns the number of pixels for the target (sky subtracted) for this
`double` **`getIntensityPerTargetPixel()`** Returns the intensity per pixel for the target (sky subtracted) for this
`double` **`getTargetError()`** Returns the error for the target (sky subtracted) for this

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

`FixedSkyAperturePhotometryProduct()`

Methods

setSkyValue(double sky)

given sky value.

Argument

`double` **`sky`** `[INPUT, MANDATORY]`

setResultsTable(Double2d resultsTable)

given table.

Argument

`Double2d` **`resultsTable`** `[INPUT, MANDATORY]`

double getSkyValue()

Return

double

Returns the sky value for this FixedSkyAperturePhotometryProduct.

double getIntensityPerSkyPixel()

Return

double

Returns the intensity per pixel for the sky for this FixedSkyAperturePhotometryProduct.

double getTargetTotal()

FixedSkyAperturePhotometryProduct.

Return

double

Returns the total flux for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

double getNbOfTargetPixels()

FixedSkyAperturePhotometryProduct.

Return

double

Returns the number of pixels for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

double getIntensityPerTargetPixel()

FixedSkyAperturePhotometryProduct.

Return

double

Returns the intensity per pixel for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

`double getTargetError()`

FixedSkyAperturePhotometryProduct.

Return

double

Returns the error for the target (sky subtracted) for this FixedSkyAperturePhotometryProduct.

3.117. FixedSkyAperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.FixedSkyAperturePhotometryTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import FixedSkyAperturePhotometryTask

Description

A Task for aperture photometry.

A Task for aperture photometry with a circular target aperture and a fixed value for the sky.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`centerX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`centerY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`centerRA`** `[INPUT, OPTIONAL, default=Default value : "centerRA"]`
`String` **`centerDEC`** `[INPUT, OPTIONAL, default=Default value : "centerDec"]`
`Double` **`radiusPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`radiusArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`sky`** `[INPUT, MANDATORY, default=Default value : 0.0]`
`boolean` **`fractional`** `[INPUT, OPTIONAL, default=Default value : true]`
`AperturePhotometryProduct` **`result`** `[OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the target center.

`Double centerY [INPUT, OPTIONAL, default=Default value : NaN]`

The y-pixel-coordinate of the target center.

`String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]`

The right ascension of the target center.

String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the target center.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The target radius (i.e. the radius of the circular target aperture) in pixels.

Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The target radius in arcsec.

Double sky [INPUT, MANDATORY, default=Default value : 0.0]

The value for the sky.

boolean fractional [INPUT, OPTIONAL, default=Default value : true]

The type of pixels.

AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

The result.

3.118. Flag

Full Name:	herschel.ia.dataset.image.Flag
Alias:	Flag
Type:	Java Class -
Import:	from herschel.ia.dataset.image import Flag
Category:	Image

Description

A class to describe Flags for images (SimpleImage, SimpleCube, SimpleStack, SlicedImage, SlicedCube and SlicedStack).

This class represents different kinds of flags in n-dimensions. The Flag is defined as an ArrayDataset. The different flag types are defined in the metadata of the ArrayDataset, a Shortnd describes the flags.

Example

Example 1: A basic example on how to create a Flag for an SimpleImage with dimensions 310 x 200.

```
flag = Flag(200, 310)
# Only the "UNVALID" flag type is available after constructing the Flag.
# Adding an extra "GLITCH" flag type.
flag.addFlagType("GLITCH", "Flag for signals affected by a cosmic ray hit")
# Adding information about the flagged pixels.
# validFlagData is a Bool2d, with the same dimensions of the flag (310 x 200)
flag.setFlag("UNVALID", validFlagData)
flag.setFlag("GLITCH", glitchFlagData)
# Getting information on Flagged pixels
print flag.getFlag("GLITCH")
```

API Summary

Constructors
Flag() The standard constructor for a Flag
Flag(int k) Constructor for a 1 dimensional flag
Flag(int k, int l) Constructor for a 2 dimensional flag
Flag(int k, int l, int m) Constructor for a 3 dimensional flag
Flag(int k, int l, int m, int n) A constructor for a four-dimensional flag of given dimensions.
Flag(int k, int l, int m, int n, int o) Constructor for a 5 dimensional flag
Flag(Flag flag) The copy constructor

Methods
`Flag` **`copy`**`()` The copy method returns a new Flag which is a copy of the original flag
`addFlagType``(String flagType, String description)` Defines a new flag type.
`String1d` **`getFlagTypes`**`()` Give a list with the defined flag types.
`boolean` **`hasFlag`**`(String name)` Checks if the flag exists.
`AbstractArrayData` **`getFlag`**`(String flagType)` Gives the flag for this specific flag type.
`AbstractArrayData` **`getFlag`**`()` Gives the flag.
`AbstractArrayData` **`getFlagAsShortnd`**`()` Gives the flag.
`setFlag``(String flagType, AbstractArrayData flag)` Sets the flag for this specific flag type.
`setFlag``(String flagType, int index, boolean flag)` Sets the flag of a certain pixel.
`setFlag``(String flagType, int row, int column, boolean flag)` Sets the flag of a certain pixel.
`setFlag``(String flagType, int index1, int index2, int index3, boolean flag)` Sets the flag of a certain pixel.
`setFlag``(String flagType, int index1, int index2, int index3, int index4, boolean flag)` Sets the flag of a certain pixel.
`setFlag``(String flagType, int index1, int index2, int index3, int index4, int index5, boolean flag)` Sets the flag of a certain pixel.

API details

Constructors

`Flag()`
A constructor for a flag. After construction, 1 flagtype is available : UNVALID

`Flag(int k)`
A constructor for a one-dimensional flag of given dimension. A Flag of k elements in 1 dimension is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID **Argument** `int` **`k`** `[INPUT, MANDATORY]` **Example** Typical example on how to create a one-dimensional Flag.

Flag(int k)

```
flag = Flag(100)
```

Flag(int k, int l)

A constructor for a two-dimensional flag of given dimensions. A Flag of (k, l) elements in 2 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

`int` **`k`** `[INPUT, MANDATORY]`

`int` **`l`** `[INPUT, MANDATORY]`

Example

Typical example on how to create a 2-dimensional Flag.

```
flag = Flag(100, 50)
```

Flag(int k, int l, int m)

A constructor for a three-dimensional flag of given dimensions. A Flag of (k, l, m) elements in 3 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

`int` **`k`** `[INPUT, MANDATORY]`

`int` **`l`** `[INPUT, MANDATORY]`

`int` **`m`** `[INPUT, MANDATORY]`

Example

Typical example on how to create a 3-dimensional Flag.

```
flag = Flag(100, 50, 75)
```

Flag(int k, int l, int m, int n)

A Flag of (k, l, m, n) elements in 4 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

`int` **`k`** `[INPUT, MANDATORY]`

`int` **`l`** `[INPUT, MANDATORY]`

`int` **`m`** `[INPUT, MANDATORY]`

`int` **`n`** `[INPUT, MANDATORY]`

Example

Typical example on how to create a 4-dimensional Flag.

```
flag = Flag(100, 50, 75, 125)
```

Flag(int k, int l, int m, int n, int o)

A constructor for a five-dimensional flag of given dimensions. A Flag of (k, l, m, n, o) elements in 5 dimensions is made. No pixels are masked out. After construction, 1 flagtype is available : UNVALID

Arguments

`int` **`k`** `[INPUT, MANDATORY]`

Flag(int k, int l, int m, int n, int o)

int **l** [INPUT, MANDATORY]

int **m** [INPUT, MANDATORY]

int **n** [INPUT, MANDATORY]

int **o** [INPUT, MANDATORY]

Example

Typical example on how to create a 5-dimensional Flag.

```
flag = Flag(100, 50, 75, 125, 25)
```

Flag(Flag flag)

The copy constructor makes an exact copy of the flag.

Argument

Flag **flag** [INPUT, MANDATORY]

Methods

Flag copy()

The copy method returns a new Flag which is a copy of the original flag

Return

Flag

Dataset A new Flag which is a copy of the current Flag.

addFlagType(String flagType, String description)

Define a new flag type : a temporary flag type that can be used at the user's discretion in the processing. This method guarantees that the identifier used does not interfere with existing identifiers. There can be no more than 16 different flag types.

Arguments

String **flagType** [INPUT, MANDATORY]

String **description** [INPUT, MANDATORY]

String1d getFlagTypes()

Give a list of the defined flag types. A String1d with all defined flag types is returned.

Return

String1d

The list of the mask type identifiers

boolean hasFlag(String name)

Checks if the flag type exists. Returns true if the flag has the given flag type.

Argument

String **name** [INPUT, MANDATORY]

Return

boolean

The list of the mask type identifiers Returns true if the flag has the given flag type.

AbstractArrayData getFlag(String flagType)

Gives the flag for this specific flag type as an AbstractArrayData. Depending on the dimensions of the flag, a Boolnd is returned with the flag for a given flag type.

Argument

`String flagType [INPUT, MANDATORY]`

Return

AbstractArrayData

A Boolnd (depending on the dimension of the flag), describing the flag of the given flagType.

AbstractArrayData getFlag()

Gives the flag as an AbstractArrayData. Depending on the dimensions of the flag, a Boolnd is returned with the flag.

Return

AbstractArrayData

A Boolnd (depending on the dimension of the flag), describing the flag

AbstractArrayData getFlagAsShortnd()

Gives the flag as an AbstractArrayData. Depending on the dimensions of the flag, a Shortnd is returned with the flag.

Return

AbstractArrayData

A Shortnd (depending on the dimension of the flag), describing the flag

setFlag(String flagType, AbstractArrayData flag)

Sets the flag for this specific flag type as an AbstractArrayData. A boolnd can be given for a flag type. If the boolnd does not have the same dimension as the dimension of the flag, an IllegalArgumentException is thrown.

Arguments

`String flagType [INPUT, MANDATORY]`

`AbstractArrayData flag [INPUT, MANDATORY]`

setFlag(String flagType, int index, boolean flag)

Set a certain flag of a certain flagType to a certain value

Arguments

`String flagType [INPUT, MANDATORY]`

`int index [INPUT, MANDATORY]`

`boolean flag [INPUT, MANDATORY]`

setFlag(String flagType, int row, int column, boolean flag)

Set a certain flag of a certain flagType to a certain value

Arguments

`String flagType [INPUT, MANDATORY]`

`int row [INPUT, MANDATORY]`

`int column [INPUT, MANDATORY]`

setFlag(String flagType, int row, int column, boolean flag)

boolean **flag** [INPUT, MANDATORY]

setFlag(String flagType, int index1, int index2, int index3, boolean flag)

Set a certain flag of a certain flagType to a certain value

Arguments

String **flagType** [INPUT, MANDATORY]

int **index1** [INPUT, MANDATORY]

int **index2** [INPUT, MANDATORY]

int **index3** [INPUT, MANDATORY]

boolean **flag** [INPUT, MANDATORY]

setFlag(String flagType, int index1, int index2, int index3, int index4, boolean flag)

Set a certain flag of a certain flagType to a certain value

Arguments

String **flagType** [INPUT, MANDATORY]

int **index1** [INPUT, MANDATORY]

int **index2** [INPUT, MANDATORY]

int **index3** [INPUT, MANDATORY]

int **index4** [INPUT, MANDATORY]

boolean **flag** [INPUT, MANDATORY]

setFlag(String flagType, int index1, int index2, int index3, int index4, int index5, boolean flag)

Set a certain flag of a certain flagType to a certain value

Arguments

String **flagType** [INPUT, MANDATORY]

int **index1** [INPUT, MANDATORY]

int **index2** [INPUT, MANDATORY]

int **index3** [INPUT, MANDATORY]

int **index4** [INPUT, MANDATORY]

int **index5** [INPUT, MANDATORY]

boolean **flag** [INPUT, MANDATORY]

See also

- SimpleImage
- SimpleCube

3.119. FlagPixels

Full Name:	herschel.ia.toolbox.spectrum.FlagPixels
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import FlagPixels

Description

Task for flagging pixels pf spectra included in a spectrum container.

The flagging task always modifies the input data.

You have different alternatives to specify a 'mask' with the pixel indices to be flagged.

- General mask: For each segment within a specific point spectrum specify an entry in a dictionary, with keys given by tuples (i,j) (where i stands for the point spectrum index and j for the segment index) and values specifying the list of pixel to be flagged.
- Uniform masks: For all the point spectra (possibly restricted by a suitable selection model - see the 'selection' parameter below) the same mask apply. This is given again by a dictionary where the keys specify the segment index (an integer) and the values specify the list of pixel indices to be flagged. Alternatively, you can directly specify a java.util.Map with Integer keys and herschel.ia.dataset.Selection as values. The values specified as Selection's contain the indices of the pixels to be flagged. Alternatively, instead of a Selection you can also specify an Int1d cointaining the indices or a Bool1d with 'true' at the positions the flag should be set.
- Uniform masks, uniform for several segments: Specify a list of pixel indices to be flagged. This mask applies to all the segments and point spectra - possibly restricted by a suitable 'selection' and 'segments' indices.

As input spectra you can specify not only a single spectrum container but also a list of spectrum containers. Alternatively, you can specify a list of tuples with the first argument in the tuple specifying the spectrum container and the second argument the mask with the pixels to be flagged.

Example

Example 1: from Jide:

```
# Set flag 2 in point spectrum 2 and segments 1 at the pixel indices 1,2,667
 and
# in point spectrum 2 and segment 3 at the pixel indices 30,690.
flagPixels(ds=spectra, mask={(2,1):[1,20,667], (2,3):[30,690]}, flag=2)
# Set flag 2 in segments 1 and 2 at the indices 1,2,667 and 30,690
 respectively.
flagPixels(ds=spectra, mask={1:[1,20,667], 2:[30,690]}, flag=2)
# flag power(2,30) is set at the same positions.
flagPixels(ds=spectra, mask={1:[1,20,667], 2:[30,690]})
# Set flag 2 in segments 1 and 2 at the indices 1,2,667 and 30,690
 respectively,
# but only in the point spectra with indices [0,1,2,3]
# For further ways of how to specify selections see e.g. the AverageSpectrum-
task
flagPixels(ds=spectra, selection=[0,1,2,3], mask={1:[1,20,667], 2:[30,690]},
 flag=2)
# Set flag 2 at the pixels 1,20,667 in the segments 1,2,3 in point spectra with
 the bbtype-attribute equal to 6031.
flagPixels(ds=spectra, mask=[1,20,667], segments=[1,2,3], selection={"bbtype":
[6031]}, flag=2)
# For spectra1 and spectra2 two SpectrumContainers:
# Set flag 2 at the pixels 1,20,667 in the segments 1,2,3 for spectra1 and
```

Example 1: from Jide:

```
# for spectra2 in point spectrum 2 and segments 1 at the pixel indices 1,2,667
 and
# in point spectrum 2 and segment 3 at the pixel indices 30,690.
flagPixels(ds=[(spectra1,[1,20,667]),(spectra2,{(2,1):[1,20,667], (2,3):
[30,690]})], segments=[1,2,3], flag=2)
# With a single container you can just specify one tuple:
flagPixels(ds=(spectra,{(2,1):[1,20,667], (2,3):[30,690]}), flag=2)
```

API Summary

Properties
Object **ds** [INOUT, MANDATORY, default=no default value.]
Object **mask** [INPUT, OPTIONAL, default=None.]
Object **selection** [INPUT, OPTIONAL, default=None.]
int **flag** [INPUT, OPTIONAL, default=2 up to the power of 30 = 1073741824.]
Boolean **setFluxToNaN** [INPUT, OPTIONAL, default=False.]

API details

Properties

Object ds [INOUT, MANDATORY, default=no default value.]

The input container(s) in which pixels should be flagged. See examples below.

Object mask [INPUT, OPTIONAL, default=None.]

Specification of the pixels to flag. See the description above and the examples below.

Object selection [INPUT, OPTIONAL, default=None.]

Specification of what point spectra the flagging mask should be applied to. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra for which the mask should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

int flag [INPUT, OPTIONAL, default=2 up to the power of 30 = 1073741824.]

The flag value to set for the specified pixel.

Boolean setFluxToNaN [INPUT, OPTIONAL, default=False.]

Boolean to indicate that the flux value of the pixels to flag should be set to NaN. This will typically remove the pixel from plots.

History

- `2009-04-29 - meli`: initial
- `2009-05-31 - meli`: first extensions

3.120. FlagSaturatedPixelsCubeTask

Full Name:	herschel.ia.toolbox.cube.FlagSaturatedPixelsCubeTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import FlagSaturatedPixelsCubeTask

Description

A Task to flag saturated pixels in a cube.

API Summary

Properties
`Cube cube [INPUT, MANDATORY, default=No default value]`
`Double value [INPUT, MANDATORY, default=No default value]`
`Cube result [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Cube cube [INPUT, MANDATORY, default=No default value]`

The cube.

`Double value [INPUT, MANDATORY, default=No default value]`

The value of saturation.

`Cube result [OUTPUT, MANDATORY, default=No default value]`

The resulting cube.

3.121. FlagSaturatedPixelsTask

Full Name:	herschel.ia.toolbox.image.FlagSaturatedPixelsTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import FlagSaturatedPixelsTask

Description

A Task to flag saturated pixels.

A Task to flag saturated pixels in an image.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double value [INPUT, MANDATORY, default=No default value]`
`Image flaggedImage [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double value [INPUT, MANDATORY, default=No default value]`

The value of saturation.

`Image flaggedImage [OUTPUT, MANDATORY, default=No default value]`

The resulting image.

3.122. Float1d

Full Name:	herschel.ia.numeric.Float1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Float1d

Description

A rectangular numeric float array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.123. Float2d

Full Name:	herschel.ia.numeric.Float2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Float2d

Description

A rectangular numeric float array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.124. Float3d

Full Name:	herschel.ia.numeric.Float3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Float3d

Description

A rectangular numeric float array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.125. Float4d

Full Name:	herschel.ia.numeric.Float4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Float4d

Description

A rectangular numeric float array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.126. Float5d

Full Name:	herschel.ia.numeric.Float5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Float5d

Description

A rectangular numeric float array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.127. FLOOR

Full Name:	herschel.ia.numeric.toolbox.basic.Floor
Alias:	FLOOR
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Floor

Description

Gives the largest integer less than or equal to x.

Returns a float array for float input array and double array for any other numeric array type.

Example

Example 1: Apply FLOOR on a Float1d

```
x=Float1d([0.1,0.5,0.9])
print FLOOR(x) # [0.0,0.0,0.0]
```

API Summary

Jython Syntax
<y>=FLOOR(<x>)

Properties
any array of any rank **x** [INPUT, MANDATORY, default=no default value]
float or double array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array of any rank x [INPUT, MANDATORY, default=no default value]

The input array can be an array of rank 1

float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array and double array for any other numeric array type.

See also

- CEIL
- ROUND

3.128. FoldSpectrum

Full Name:	herschel.ia.toolbox.spectrum.FoldSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import FoldSpectrum

Description

Task for folding frequency-switched spectra.

The folded spectra are constructed by averaging the original spectra with a shifted and inverted copy. This algorithm corresponds to the most simple scheme described in article {\it "Recovering line profiles from frequency-switched spectra}, H.Liszt, Astron. Astrophys. Suppl. Ser. 124, 183-188 (1997)}. By default, the task modifies the input spectra. Note that the operation is performed on a per segment basis. For that reason you may consider stitching the segments before folding.

API Summary

Properties
`SpectrumContainer` **`ds`** `[INPUT, MANDATORY, default=no default value.]`
`SpectrumContainer` **`result`** `[OUTPUT, MANDATORY, default=no default value.]`
`Double` **`freqThrow`** `[INPUT, MANDATORY, default=0.0.]`
`Boolean` **`shift`** `[INPUT, OPTIONAL, default=False.]`
`Boolean` **`overwrite`** `[INPUT, OPTIONAL, default=True.]`

API details

Properties

`SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]`

Input container with the spectra to be folded.

`SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]`

Output container with the folded spectra.

`Double freqThrow [INPUT, MANDATORY, default=0.0.]`

The frequency throw in the spectra to be folded. The same specified value is applied to all the spectra in the container.

`Boolean shift [INPUT, OPTIONAL, default=False.]`

If true, the spectrum is shifted in frequency scale by half the throw distance so that the resulting spectrum is centred between the original and the "switched" spectrum.

`Boolean overwrite [INPUT, OPTIONAL, default=True.]`

Specify whether the input data container can be reused - the values found therein are overwritten. If set to false, a new container is created and the spectral segments are reduced in shape by the frequency throw distance.

3.129. FullQuery

Full Name:	herschel.ia.pal.query.FullQuery
Type:	Java Class -
Import:	from herschel.ia.pal.query import FullQuery

Description

A data mining query formulates a query the full interface of a

Product. As such, one can query any aspect of a Product. Typically this type of query is quite slow and should be used in combination with query refinements.

Example

Example 1: Example of a data mining query

```
q=FullQuery(MyProduct.class,"p",  "ANY(p['array'].data<2)")
```

See also

- Querying

3.130. GAMMALN

Full Name:	herschel.ia.numeric.toolbox.basic.GammaLn
Alias:	GAMMALN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import GammaLn

Description

Computes the natural log of the Gamma function.

Example

Example 1: Apply GAMMALN on a Double1d

```
x = Double1d([1.0,2.0,3.0,4.0,5.0,6.0])
gn = GAMMALN(x)
print gn  # [0.0,-4.440892098500626E-16,0.6931471805599443,
            1.791759469228055,3.1780538303479453,4.787491742782044]
```

API Summary

Jython Syntax

```
<y>=GAMMALN(<x>)
```

Properties

```
an double array or a number x [INPUT, MANDATORY, default=.]
returns an array y [OUTPUT, (or a number if the input is a
number), default=no default value]
```

API details

Properties

an double array or a number x [INPUT, MANDATORY, default=.]

returns an array y [OUTPUT, (or a number if the input is a number), default=no default value]

where each element is the natural logarithm of the Gamma function of the corresponding element of the input array.

See also

- GammaP
- GammaQ

3.131. GammaP

Full Name:	herschel.ia.numeric.toolbox.basic.GammaP
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import GammaP

Description

Computes the incomplete Gamma function P(a,x).

Example

Example 1: Apply GammaP on a Double1d

```
a = 0.5
x = Double1d([0.0316228,0.0707107,5.0,1.04881,2.44949,25.4951])
gp = GammaP(a)(x)
print gp  # [0.19856221732013887,0.2931279825202761,0.9984345977771615,
            0.8524713739638958,0.9731274396553844,0.9999999999990717]
]
```

API Summary

Jython Syntax
<y>=GammaP(a)(<x>)

Properties
a double value **a** [INPUT, MANDATORY, default=.]
a double array or number **x** [INPUT, MANDATORY, default=.]
returns an double or float array (or a number **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

a double value a [INPUT, MANDATORY, default=.]

a double array or number x [INPUT, MANDATORY, default=.]

returns an double or float array (or a number y [OUTPUT, MANDATORY, default=no default value]

if the input is a number) where each element is the incomplete Gamma function of the corresponding element of the input array.

See also

- GammaQ
- GAMMALN

3.132. GammaQ

Full Name:	herschel.ia.numeric.toolbox.basic.GammaQ
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import GammaQ

Description

Computes the complement of incomplete Gamma function Q(a,x).

Example

Example 1: Apply GammaQ on a Double1d

```
a = 0.5
x = Double1d([0.5,0.0316228,0.0707107,5.0,1.04881,2.44949,25.4951])
gq = GammaQ(a)(x)
print gq  # [0.31731050863888255,0.8014377826798611,0.7068720174797238,

 0.001565402222838555,0.14752862603610417,0.026872560344615565,9.282826956361127E-13]
```

API Summary

Jython Syntax
<y>=GammaQ(a)(<x>)

Properties
a double value **a** [INPUT, MANDATORY, default=.]
a double array or number **x** [INPUT, MANDATORY, default=.]
returns an array (or a number if the input is a number) **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

a double value a [INPUT, MANDATORY, default=.]

a double array or number x [INPUT, MANDATORY, default=.]

returns an array (or a number if the input is a number) y [OUTPUT, MANDATORY, default=no default value]

where each element is the complement of the incomplete Gamma function of the corresponding element of the input array.

See also

- GammaP
- GAMMALN

3.133. GaussFitTask

Full Name:	herschel.ia.toolbox.fit.GaussFitTask
Alias:	GaussFitTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.fit import GaussFitTask
Category:	generic task

Description

generic module: GaussFitTask

Task shell around the package herschel.ia.numeric.toolbox.fit. Both a command line and a GUI interface are provided. With the GUI interface there is more of the Fitter package accessable than on this tasks command line. However not everything possible within the fitter package is accessable via this task, GUI or otherwise. Most text fields in the GUI accept either numerics or JIDE variables or JIDE expressions.

Example

Example 1: GaussFitTask

```
<pre>
from herschel.hifi.generic.task import GaussFitTask
# Assume that `tt' and/or `data' are Double1d's
# usage on command line:
ft = GaussFitTask()( x=tt, y=data )
ft.fitter = "LevenbergMarquardtFitter"
ft()            # performs the execute method
print ft.result        # parameters of the fit
print ft.stdev        # standard deviations
print ft.fitter        # name of the fitter
# Etcetera, etc.
# use the GUI.
ft = GaussFitTask()
ft.gui = 1         # start the GUI
# from the GUI select the data, model, fitter, properties etc and run.
print ft.result        # parameters of the fit
print ft.stdev        # standard deviations
print ft.fitter        # name of the fitter
# Etcetera as before.
</pre>
```

API Summary

Jython Syntax

```
see example below
```

Properties

```
NumericData x [INPUT, OPTIONAL, default=null]
DoubleArray y [INPUT, MANDATORY, default=no]
Boolean indexview [INPUT, OPTIONAL, default=true]
DoubleArray weights [INPUT, OPTIONAL, default=null]
AbstractModel model [INPUT, OPTIONAL, default=null]
```

Properties
String **modelname** [INPUT, OPTIONAL, default=no]
Array1dData **modelarg** [INPUT, OPTIONAL, default=null]
Fitter **fitter** [INPUT, OPTIONAL, default=null]
String **fittername** [INPUT, true, default=no]
String **autoinit** [INPUT, OPTIONAL, default=""]
Integer **smooth** [INPUT, OPTIONAL, default=1]
Double1d **result** [OUTPUT, OPTIONAL, default=n/a]
Double1d **parameters** [OUTPUT, OPTIONAL, default=n/a]
Double1d **stdev** [OUTPUT, OPTIONAL, default=n/a]
Double **chisq** [OUTPUT, OPTIONAL, default=n/a]
DoubleArray **yfit** [OUTPUT, OPTIONAL, default=n/a]
DoubleArray **yband** [OUTPUT, OPTIONAL, default=n/a]
Double1d **prior** [INPUT, OPTIONAL, default=null]
Double **evidence** [OUTPUT, OPTIONAL, default=n/a]
Double **scale** [OUTPUT, OPTIONAL, default=n/a]
Boolean **auto** [INPUT, OPTIONAL, default=false]
Double **fixed** [INPUT, OPTIONAL, default=1.0]
Double **mixed** [INPUT, OPTIONAL, default=0.0]
Double **tolerance** [INPUT, OPTIONAL, default=0.01]
Integer **iterations** [INPUT, OPTIONAL, default=10000]
Double **temperature** [INPUT, OPTIONAL, default=0.0]
Double **cooling** [INPUT, OPTIONAL, default=0.95]
Integer **tempsteps** [INPUT, OPTIONAL, default=100]
Double1d **initialpars** [INPUT, OPTIONAL, default=from model]
Double1d **highlimits** [INPUT, OPTIONAL, default=null]
Double1d **lowlimits** [INPUT, OPTIONAL, default=null]
Int1d **keepfixed** [INPUT, OPTIONAL, default=null]
Double1d **fixedvalues** [INPUT, OPTIONAL, default=null]
Boolean **gui** [INPUT, OPTIONAL, default=false]

Limitations

Not everything which is possible using the package directly is accessable via this task.

API details

Properties

NumericData x [INPUT, OPTIONAL, default=null]
The independent variable(s) of the fit problem. If not provided, it will use indices ranging from 0 to len(y). If y is more-dimensional the proper 2-d indices are provided.

DoubleArray y [INPUT, MANDATORY, default=no]
The data to be fitted. It can be more-dimensional. E.g. a 2-dimensional map.

Boolean indexview [INPUT, OPTIONAL, default=true]

Applicable parameters appear in the same order as in the y DoubleArray This parameter is only active when y has 2 dimensions or more.

DoubleArray weights [INPUT, OPTIONAL, default=null]

The weights to be used in the fit. Can be more-dimensional. When provided it should be the same size as parameter y.

AbstractModel model [INPUT, OPTIONAL, default=null]

The model to be fitted. Default: GaussModel (for 1d data) or Gauss2DModel (for 2d data)

String modelname [INPUT, OPTIONAL, default=no]

The model to be fitted. Default: GaussModel (for 1d data) or Gauss2DModel (for 2d data)

Array1dData modelarg [INPUT, OPTIONAL, default=null]

Arguments, if any, needed in the Constructor of the model.

Fitter fitter [INPUT, OPTIONAL, default=null]

Fitter to be used. Default: LevenbergMarquardtFitter

String fittername [INPUT, true, default=no]

Fitter to be used. Default: LevenbergMarquardtFitter

String autoinit [INPUT, OPTIONAL, default=""]

Automated search for initial params. options: "high", "low", "center"

Integer smooth [INPUT, OPTIONAL, default=1]

Smooth data with BoxCarFilter before auto-search. smooth > 1.

Double1d result [OUTPUT, OPTIONAL, default=n/a]

The parameters of the fit.

Double1d parameters [OUTPUT, OPTIONAL, default=n/a]

The parameters of the fit. (same as result)

Double1d stdev [OUTPUT, OPTIONAL, default=n/a]

The standard deviations pertaining to the parameters.

Double chisq [OUTPUT, OPTIONAL, default=n/a]

Chi-squared of the fit.

DoubleArray yfit [OUTPUT, OPTIONAL, default=n/a]

An array of the same size as the data, containing the fitted valued

DoubleArray yband [OUTPUT, OPTIONAL, default=n/a]

An array of the same size as the data, containing a 1-# MonteCarlo error at each point.

Double1d prior [INPUT, OPTIONAL, default=null]

Prior ranges for the parameters, needed when the evidence is requested.

Double evidence [OUTPUT, OPTIONAL, default=n/a]

Relative evidence the model carries w.r.t. the data. It needs values for prior, as many as the number of parameters and one more if auto-scaling is set.

Double scale [OUTPUT, OPTIONAL, default=n/a]

Noise scale of the data (when auto or mixed is selected)

Boolean auto [INPUT, OPTIONAL, default=false]

Select automatic noise scaling.

Double fixed [INPUT, OPTIONAL, default=1.0]

Fixed noise scale.

Double mixed [INPUT, OPTIONAL, default=0.0]

Automatic noise scaling with a minimum.

Double tolerance [INPUT, OPTIONAL, default=0.01]

Stopping criterion for iterative fitting.

Integer iterations [INPUT, OPTIONAL, default=10000]

Maximum number of iterations.

Double temperature [INPUT, OPTIONAL, default=0.0]

Starting temperature in annealing amoeba fitting.

Double cooling [INPUT, OPTIONAL, default=0.95]

Cooling step in annealing amoeba fitting.

Integer tempsteps [INPUT, OPTIONAL, default=100]

Number of exploratory steps at each temperature.

Double1d initialpars [INPUT, OPTIONAL, default=from model]

Initial parameters in iterative fitters.

Double1d highlimits [INPUT, OPTIONAL, default=null]

Upper limits of the parameters (only in AmoebaFitter)

Double1d lowlimits [INPUT, OPTIONAL, default=null]

Lower limits of the parameters (only in AmoebaFitter)

Int1d keepfixed [INPUT, OPTIONAL, default=null]

Keep these parameters fixed. (Parameter numbering starts at 0)

Double1d fixedvalues [INPUT, OPTIONAL, default=null]

Values at which the parameters should be kept.

Boolean gui [INPUT, OPTIONAL, default=false]

Allows to handle this task using a gui.

History

- 15-10-2006 DK

3.134. GaussianFilter

Full Name:	herschel.ia.numeric.toolbox.filter.GaussianFilter
Alias:	GaussianFilter
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.filter import GaussianFilter

Description

Creates a Gaussian filter, that can be applied to numeric arrays of rank 1.

Example

Example 1: Apply GaussianFilter on a Int1d

```
x=Int1d([1,3,2,4,6,5])
f=GaussianFilter(0.2)
print f(x) # [0.10650697891920073,1.1065069789192006,
             2.6804790632423976, 2.319520936757602,
             3.9999999999999996, 5.680479063242397]
```

API Summary

Jython Syntax

```
<f>=GaussianFilter(<sigma> [,<center>=true|false]
[,<edge>=Convolution.ZEROES|CIRCULAR|REPEAT])
<x>=<f>(<x>)
```

Properties

```
real sigma [INPUT, MANDATORY, default=no default value]
boolean center [INPUT, NOT_MANDATORY, default=false]
Convolution.ZEROES|CIRCULAR|REPEAT center [INPUT, NOT_MANDATORY,
default=Convolution.ZEROES]
```

API details

Properties

real sigma [INPUT, MANDATORY, default=no default value]

It must be a real.

boolean center [INPUT, NOT_MANDATORY, default=false]

Set center to true or false.

Convolution.ZEROES|CIRCULAR|REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]

Set edge to true or false

- edge=Convolution.ZEROES: Set result to zero at edges.

Convolution.ZEROES|CIRCULAR|REPEAT center [INPUT, NOT_MANDATORY, default=Convolution.ZEROES]
• edge=Convolution.CIRCULAR: Wrap around at edges (circular convolution). • edge=Convolution.REPEAT: Repeat the edge values of input array.

See also

- BoxCarFilter
- Convolution

3.135. GaussianSmoothingTask

Full Name:	herschel.ia.toolbox.image.GaussianSmoothingTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import GaussianSmoothingTask

Description

A Task to smooth an image using a convolution with a gaussian.

A Task to smooth an image by convolving it with a gaussian function.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double sigma [INPUT, MANDATORY, default=Default value : Double(3.0)]`
`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image before smoothing.

`Double sigma [INPUT, MANDATORY, default=Default value : Double(3.0)]`

The standard deviation of the gaussian.

`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

The image after smoothing.

3.136. HAMMING

Full Name:	herschel.ia.numeric.toolbox.xform.Hamming
Alias:	HAMMING
Type:	Java Class - 
Import:	from herschel.ia.numeric.toolbox.xform import Hamming

Description

Hamming function for multiplying with input arrays.

This function performs an element-by-element multiplication of a input array with a Hamming function having the same number of elements. The Hamming function is defined by the following equation:

```
Hamming[i] = 0.54 + 0.46 * cos(2#( i-# ) / N)
where:
Hamming[i] is the value of the Hamming function at array index i, where 0#i\
#(N-1)
N is the length of the array for which the Hamming function is calculated.
# is the phase shift, defined as # = N/2.0
```

Input:

The input array should be a `Double1d` or `Float1d` and have length N#3.

Output:

The output array is a floating point array of length N that contains the element-by-element multiplication of the input array with the Hamming function.

Symmetry:

This equation produces a Hamming function with a unique point at index `i=0`. For even length arrays, the maximum amplitude is located at `N/2.0`. For odd length arrays, there are two amplitude maxima located at `ceiling(N/2.0)` and `floor(N/2.0)`.

Syntax:

To apply an element-by-element multiplication of the array `x` by the Hamming function, use any of the following options while coding in Jython:

1. `p = HAMMING(x) #Area normalized`
2. `p = HAMMING.AREA(x) #Area normalized, alternative syntax`
3. `q = HAMMING.AMPLITUDE(x) #Amplitude normalized`
4. `r = HAMMING.ENERGY(x) #Energy normalized`

Example

Example 1: Apply Hamming function to an input array x in java:

```
<pre>
 Double1d x = new Double1d(100,1.0); // create array of 100 1's
 Double1d p = (Double1d)x.apply(Hamming.AREA); // Area normalized
 Double1d p2 = (Double1d)x.apply(Hamming.PROCEDURE);  //Area normalized,
 alternative syntax
```

Example 1: Apply Hamming function to an input array x in java:

```
 Double1d q = (Double1d)x.apply(Hamming.AMPLITUDE);  //Amplitude normalized
 Double1d r = (Double1d)x.apply(Hamming.ENERGY);  //Energy normalized
</pre>
 In Jython:
<pre>
x = Double1d(100,1.0)
p = HAMMING(x) #Area normalized
p2 = HAMMING.AREA(x) #Area normalized, alternative syntax
q = HAMMING.AMPLITUDE(x) #Amplitude normalized
r = HAMMING.ENERGY(x) #Energy normalized
</pre>
```

API Summary

Jython Syntax
<y>=HAMMING.[<normalization> = AREA|AMPLITUDE|ENERGY] (<x>)

Properties
Double1d or Float1d **x** [INPUT, MANDATORY, default=No default value]
Double1d or Float1d **y** [OUTPUT, OPTIONAL, default=No default value]
AMPLITUDE|ENERGY|AREA|PROCEDURE **normalization** [INPUT, OPTIONAL, default=No default value]

Limitations

This function on operates on `Double1d` and `Float1d` arrays. It does not work for `Complex1d` arrays.

API details

Properties

Double1d or Float1d x [INPUT, MANDATORY, default=No default value]

The input array to be multiplied by the Hamming function.

Double1d or Float1d y [OUTPUT, OPTIONAL, default=No default value]

Outputs an array containing the element-by-element multiplication of the input array with the Hamming function.

AMPLITUDE|ENERGY|AREA|PROCEDURE normalization [INPUT, OPTIONAL, default=No default value]

The type of normalization of the Hamming function.

See also

- HAMMING

History

- 2007-Nov-23 (ZW) Added two new types of normalization.

- Changed the private constructor to take in one String parameter
- in order to have 3 types of normalization of the window functions.
- Changed some javadoc.
- 2008-Mar-07 (PK) Added static PROCEDURE member, modified n2 division and refactored.
- Used double to calculate n2 peak of the Hamming function thus affecting the symmetry of function.
- 2008-Mar-13 (PK) Changed to use a unique static member for each normalization.

3.137. HANNING

Full Name:	herschel.ia.numeric.toolbox.xform.Hanning
Alias:	HANNING
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.xform import Hanning

Description

Hanning function for multiplying with input arrays.

This function performs an element-by-element multiplication of a input array with a Hanning function having the same number of elements. The Hanning function is defined by the following equation:

```
Hanning[i] = 0.50 + 0.50 * cos(2#( i-# ) / N)
where:
Hanning[i] is the value of the Hanning function at array index i, where 0#i\
#(N-1)
N is the length of the array for which the Hanning function is calculated.
# is the phase shift, defined as # = N/2.0
```

Input:

The input array should be a `Double1d` or `Float1d` and have length N#3.

Output:

The output array is a floating point array of length N that contains the element-by-element multiplication of the input array with the Hanning function.

Symmetry:

This equation produces a Hanning function with a unique point at index `i=0`. For even length arrays, the maximum amplitude is located at `N/2.0`. For odd length arrays, there are two amplitude maxima located at `ceiling(N/2.0)` and `floor(N/2.0)`.

Syntax:

To apply an element-by-element multiplication of the array `x` by the Hanning function, use any of the following options while coding in Jython:

1. `p = HANNING(x) #Area normalized`
2. `p = HANNING.AREA(x) #Area normalized, alternative syntax`
3. `q = HANNING.AMPLITUDE(x) #Amplitude normalized`
4. `r = HANNING.ENERGY(x) #Energy normalized`

Example

Example 1: Apply Hanning function to an input array x in java:

```
<pre>
 Double1d x = new Double1d(100,1.0); // create array of 100 1's
 Double1d p = (Double1d)x.apply(Hanning.AREA); // Area normalized
 Double1d p2 = (Double1d)x.apply(Hanning.PROCEDURE);  //Area normalized,
 alternative syntax
```

Example 1: Apply Hanning function to an input array x in java:

```
 Double1d q = (Double1d)x.apply(Hanning.AMPLITUDE);  //Amplitude normalized
 Double1d r = (Double1d)x.apply(Hanning.ENERGY);  //Energy normalized
</pre>
 In Jython:
<pre>
x = Double1d(100,1.0)
p = HANNING(x) #Area normalized
p2 = HANNING.AREA(x) #Area normalized, alternative syntax
q = HANNING.AMPLITUDE(x) #Amplitude normalized
r = HANNING.ENERGY(x) #Energy normalized
</pre>
```

API Summary

Jython Syntax
<y>=HANNING.[<normalization> = AREA|AMPLITUDE|ENERGY] (<x>)

Properties
Double1d or Float1d **x** [INPUT, MANDATORY, default=No default value]
Double1d or Float1d **y** [OUTPUT, OPTIONAL, default=No default value]
AMPLITUDE|ENERGY|AREA|PROCEDURE **normalization** [INPUT, OPTIONAL, default=No default value]

Limitations

This function on operates on `Double1d` and `Float1d` arrays. It does not work for `Complex1d` arrays.

API details

Properties

Double1d or Float1d x [INPUT, MANDATORY, default=No default value]

The input array to be multiplied by the Hanning function.

Double1d or Float1d y [OUTPUT, OPTIONAL, default=No default value]

Outputs an array containing the element-by-element multiplication of the input array with the Hanning function.

AMPLITUDE|ENERGY|AREA|PROCEDURE normalization [INPUT, OPTIONAL, default=No default value]

The type of normalization of the Hanning function.

See also

- HANNING

History

- 2007-Nov-23 (ZW) Added two new types of normalization.

- Changed the private constructor to take in one String parameter
- in order to have 3 types of normalization of the window functions.
- Changed some javadoc.
- 2008-Mar-07 (PK) Added static PROCEDURE member, modified n2 division and refactored.
- Used double to calculate n2 peak of the Hanning function thus affecting the symmetry of function.
- 2008-Mar-13 (PK) Changed to use a unique static member for each normalization.

3.138. HduHeaders

Full Name:	herschel.ia.io.fits.HduHeaders
Alias:	HduHeaders
Type:	Java Class -
Import:	from herschel.ia.io.fits import HduHeaders
Category:	class

Description

A class for working with FITS HDUs. HDU data is loaded only when requested.

Restrictions:

- Only HCSS FITS files (version 4) can be modified and saved.
- Avoid working with several datasets at the same time if you are planning to modify them.
- Only FITS files can be modified (fits from an InputStream cannot be updated).
- If the HDU is a compositeDataset, the children are returned when asking for the HDU dataset.
- If you modify a CompositeDataset, only its Metadata can be updated. If you want to add or remove a CompositeDataset child, you must work with it directly (add/remove CompositeDataset children directly).
- If you remove a CompositeDataset, its children are removed also.
- When modifying/updating/removing HDUs, a temporary file is created.

Examples

Example 1: default usage:

```
fa = FitsArchive();
#fa.reader = FitsArchive.STANDARD_READER #for reading non HCSS FITS files.
fh = fa.getFitsHduHeaders(path)
print fh
#get last dataset
ds = fh.headers[fh.numHeaders-1].dataset
print ds
#or
h = fh.headers[fh.numHeaders-1]
ds = h.dataset
print ds
```

Example 2: modify a metadata/fits header:

```
fa = FitsArchive();
fh = fa.getFitsHduHeaders(path)
print fh
#get last dataset
h = fh.headers[fh.numHeaders-1]
ds = h.dataset
print ds
ds.description = 'new description'
h.updateDataset(ds)
#or
fh.updateDataset(fh.numHeaders-1, ds)
```

Example 3: add a new dataset/HDU:

```
fa = FitsArchive();
fh = fa.getFitsHduHeaders(path)
ds = ArrayDataset(Int1d([1,2,3]))
fh.addDataset('new_dataset_id',ds)
```

Example 4: remove a HDU:

```
fa = FitsArchive();
fh = fa.getFitsHduHeaders(path)
#remove last header
fh.removeHeader(fh.numHeaders-1)
```

3.139. help

Full Name:	herschel.ia.toolbox.util.HelpTask
Alias:	help
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import HelpTask
Category:	task

Description

The help task for interactive help.

Users can access the help system by calling help from the command line. Help opens a pop up window displaying the specific help topic for the specifed item or the main entry of documentation if no entry is specified or specific help can found.

The current implementation displays the specific help only for tasks, Plot and Image.

Examples

Example 1: overview help

```
help()
```

Example 2: help on help

```
help(help)
```

Example 3: help on plot (1)

```
p = PlotXY
help(p)
```

Example 4: help on plot (2)

```
help("plotxy")
```

Example 5: help on display (1)

```
d = Display()
help(d)
```

Example 6: help on display (2)

```
help("display")
```

API Summary

Jython Syntax
help() help(item)

Property
`Object item [INPUT, NO, default=help]`

Limitations

The current implementation displays the specific help only for tasks, Plot and Image.

Miscellaneous

No miscellaneous

API details

Property

`Object item [INPUT, NO, default=help]`
The item to look for in the help. Currently supported: Task(s), Plot, Image (see examples)

See also

- references

History

- `2004-07-13 - NdC`: first release.

3.140. Histogram

Full Name:	herschel.ia.numeric.toolbox.basic.Histogram
Alias:	Histogram
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Histogram

Description

Create a histogram on a Numeric data type, using a user-specified

bin size.

Examples

Example 1: Apply Histogram on a Double1d

```
from herschel.ia.numeric import *
from herschel.ia.numeric.toolbox.random import RandomGauss
from herschel.ia.numeric.toolbox.basic import Histogram
from herschel.ia.numeric.toolbox.basic import BinCentres
hist=Histogram (1)
x=Double1d( [1,2,3,4] )
print hist(x)  [1,1,1,1]
```

Example 2: Plot histogram over bin centres

```
d = Double1d(200000,10)
d[99000:110000] = 15
d[99900:101000] = 20.0
rnd = RandomGauss()
for ix in range(200000):
   d[ix] += rnd.calc(1.0)
p = PlotXY (d)
binsize = STDDEV (d) * 2.35
print binsize
hist=Histogram(binsize)
bins=BinCentres(binsize)
p=PlotXY(bins(d),hist(d)
# or you could try:  p = PlotXY (Histogram(binsize), BinCentres (binSize)
style=p.getStyle()
style.setChartType(Style.HISTOGRAM)
```

API Summary

Jython Syntax

```
<histogram>=Histogram(binSize)
<H>=histogram(<x>)
```

Properties

```
type binSize [INPUT, MANDATORY, default=no default value]
any array type x [INPUT, MANDATORY, default=no default value]
```

API details

Properties

type binSize [INPUT, MANDATORY, default=no default value]

Size of the histogram bins. See BinCentres for a description of the array of x-values for the centers of a histogram

any array type x [INPUT, MANDATORY, default=no default value]

See also

- BinCentres

3.141. Histogram

Full Name:	herschel.ia.toolbox.image.gui.Histogram
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import Histogram

Description

An implementation of making a histogram. You can choose the kind of area

of which to make a histogram, give the cut levels and the number of bins, used for the histogram. When you push the plot-button, the histogram will be shown and the eventual region will be marked on the image.

API Summary

Constructors
Histogram(ImageAnalysisToolbox toolbox) The standard constructor for Histogram. This constructor creates a window
Histogram(boolean useAsComponent, ImageAnalysisToolbox toolbox) Constructor for Histogram. When the new Histogram is

Methods
PlotXY **getPlot**() Returns the PlotXY shown in this Histogram.
JPanel **getPanel**() Returns the panel of this Histogram.
int **getUsedArea**() Returns the used type of area for this Histogram.
int **getUsedRegion**() Returns the used type of region for this Histogram.
int **getNumberOfEdges**() Returns the number of edges, if the region of interest is a
int **getUsedCutLevels**() Returns the used type of cut levels for this Histogram.
double[] **getCutlevels**() Returns the used cut levels (min, max) for this Histogram.
int **getNumberOfBins**() Returns the used number of bins for this Histogram.
int **getBinWidth**() Returns the width of the bins for this Histogram.
int **getNbOfClicks**() Returns the number of valid clicks (i.e. in the image) you made.
boolean **checkInput**() Returns true if the given input is correct; false otherwise. The

Methods
`boolean` **`checkArea()`** Checks whether the input area is correct. The error message is
`boolean` **`checkParameters()`** Checks whether the input parameters are correct. The error message
`boolean` **`checkCutLevels()`** Checks whether the input cut levels are correct. The error message
`boolean` **`checkNumberOfBins()`** Checks whether the input number of bins is correct. The error
`ArrayList` **`getImageFigures()`** Returns all ImageFigures used for this AreaHistogram (the eventual

API details

Constructors

`Histogram(ImageAnalysisToolbox toolbox)`

in which you can make a histogram. In that window you give the input and retrieve the output.

Argument

`ImageAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

`Histogram(boolean useAsComponent, ImageAnalysisToolbox toolbox)`

component-based, a window for plotting the histogram is opened; otherwise nothing will be shown.

Arguments

`boolean` **`useAsComponent`** `[INPUT, MANDATORY]`

`ImageAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

Methods

`PlotXY getPlot()`

Return

PlotXY

Returns the PlotXY shown in this Histogram.

`JPanel getPanel()`

Return

JPanel

Returns the panel of this Histogram.

`int getUsedArea()`

Return

int

Returns the used type of area for this Histogram.

int getUsedRegion()

Return

int

Returns the used type of region for this Histogrem.

int getNumberOfEdges()

polygon; otherwise -1 is returned.

Return

int

Returns the number of edges, if the region of interest is a polygon; otherwise -1 is returned.

int getUsedCutLevels()

Return

int

Returns the used type of cut levels for this Histogram.

double[] getCutlevels()

Return

double[]

Returns the used cut levels (min, max) for this Histogram.

int getNumberOfBins()

Return

int

Returns the used number of bins for this Histogram.

int getBinWidth()

Return

int

Returns the width of the bins for this Histogram.

int getNbOfClicks()

Return

int

Returns the number of valid clicks (i.e. in the image) you made.

boolean checkInput()

eventual appropriate error message is also constructed here.

Return

boolean checkInput()

boolean

Returns true if the given input is correct; false otherwise.

boolean checkArea()

adapted, when errors occur.

Return

boolean

Returns true if the input area is correct; false otherwise.

boolean checkParameters()

is adapted, when errors occur.

Return

boolean

Returns true if the input parameters are correct; false otherwise.

boolean checkCutLevels()

is adapted, when errors occur.

Return

boolean

Returns true if the input cut levels are correct; false otherwise.

boolean checkNumberOfBins()

message is adapted, when errors occur.

Return

boolean

Returns true if the input number of bins is correct; false otherwise.

ArrayList getImageFigures()

region of interest).

Return

ArrayList

Returns all ImageFigures used for this AreaHistogram (the eventual region of interest).

3.142. HistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.HistogramPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import HistogramPanel

Description

A panel for the HistogramTask.

API Summary

Constructor
HistogramPanel() The constructor of a new HistogramPanel.

Methods
JPanel **getImagePanel()** Returns a panel that displays the image for this
JPanel **getButtonPanel()** Returns the button panel for this HistogramPanel.
setSiteEventHandler(SiteEventHandler handler) Sets the site event handler for this HistogramPanel to the
setTask(TaskApi task) Associates the given task with this HistogramPanel.
setVariableSelection(VariableSelection selection) Sets the variable selection for this HistogramPanel to the given
Display **getDisplay()** Returns the display for this HistogramPanel.
Image **getImage()** Returns the image associated with this HistogramPanel.
TaskApi **getTask()** Returns the task associated with this
Map **getMap()** Returns the map associated with this HistogramPanel.
SiteEventHandler **getHandler()** Returns the site event handler associated with this
Double **getInputLowCut()** Returns the input low cut level for this
Double **getInputHighCut()** Returns the input high cut level for this
getInputNbOfBins() Returns the input number of bins for this
PlotXY **getHistogram()**

Methods
Returns the histogram associated with this
setPoint(MouseEvent me)
Sets the position of the given mouse event at the
setPoint(Double point)
Sets the given screen coordinates at the appropriate place
updateFigure()
Updates the figure associated with this
ArrayList **getClickedPoints**()
Returns the clicked points for this HistogramPanel.
int **getNbOfClickedPoints**()
Returns the number of times there was clicked before
increaseClicks()
Increases the number of clicks made before on the image
drawFigure()
Draws the figure on the image associated with this
trigger()
Triggers the execution of the task associated
updateHistogram()
Updates the histogram associated with this

API details

Constructor

HistogramPanel()

Methods

JPanel getImagePanel()
HistogramPanel.
Return
JPanel
Returns a panel that displays the image for this HistogramPanel.

JPanel getButtonPanel()
Return
JPanel
Returns the button panel for this HistogramPanel.

setSiteEventHandler(SiteEventHandler handler)
given site event handler.
Argument

setSiteEventHandler(SiteEventHandler handler)

SiteEventHandler **handler** [INPUT, MANDATORY]

setTask(TaskApi task)

Argument

TaskApi **task** [INPUT, MANDATORY]

setVariableSelection(VariableSelection selection)

variable selection.

Argument

VariableSelection **selection** [INPUT, MANDATORY]

Display getDisplay()

Return

Display

Returns the display for this HistogramPanel.

Image getImage()

Return

Image

Returns the image associated with this HistogramPanel.

TaskApi getTask()

HistogramPanel.

Return

TaskApi

Returns the task associated with this HistogramPanel.

Map getMap()

Return

Map

Returns the map associated with this HistogramPanel.

SiteEventHandler getHandler()

HistogramPanel.

Return

SiteEventHandler

Returns the site event handler associated with this HistogramPanel.

Double getInputLowCut()

HistogramPanel.

Double getInputLowCut()

Return

Double

Returns the input low cut level for this HistogramPanel.

Double getInputHighCut()

HistogramPanel.

Return

Double

Returns the input high cut level for this HistogramPanel.

getInputNbOfBins()

HistogramPanel.

PlotXY getHistogram()

HistogramPanel.

Return

PlotXY

Returns the histogram associated with this HistogramPanel.

setPoint(MouseEvent me)

appropriate place (i.e. at the index that corresponds with the number of times there was clicked before) in the list of clicked points for this HistogramPanel in pixel coordinates.

Argument

`MouseEvent` **me** `[INPUT, MANDATORY]`

setPoint(Double point)

(i.e. at the index that corresponds with the number of times there was clicked before) in the list of clicked points for this HistogramPanel in pixel coordinates.

Argument

`Double` **point** `[INPUT, MANDATORY]`

updateFigure()

HistogramPanel.

ArrayList getClickedPoints()

Return

ArrayList

Returns the clicked points for this HistogramPanel.

int getNbOfClickedPoints()

on the image for this HistogramPanel.

Return

`int getNbOfClickedPoints()`

int

Returns the number of times there was clicked before on the image for this HistogramPanel.

`increaseClicks()`

for this HistogramPanel.

`drawFigure()`

HistogramPanel.

`trigger()`

with this HistogramPanel.

`updateHistogram()`

HistogramPanel.

3.143. HistogramTask

Full Name:	herschel.ia.toolbox.image.HistogramTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import HistogramTask

Description

An abstract Task to make histograms.

An abstract Task for making a histogram of an image as a whole or of a certain region of interest, which is bounded by a circle, an ellipse, a rectangle or a polygon. Its subclasses/subtasks are ImageHistogramTask, CircleHistogramTask, EllipseHistogramTask, RectangleHistogramTask and PolygonHistogramTask.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Double bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`

The lower cut level.

`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`

The upper cut level.

`Double bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`

The number of bins.

3.144. IaProcessImpl

Full Name:	herschel.ia.dataflow.IaProcessImpl
Type:	Java Class -
Import:	from herschel.ia.dataflow import IaProcessImpl

Description

Implementation of the interface {@link IaProcess}

It has additional helper methods to make a developer's life easier. It is the base class for building event-based processes. A developer may inherit from this class in order to use it within the dataflow framework.

Examples

Example 1: how to implement a event-based process

```
<pre>
import herschel.ia.dataflow.*;
import herschel.ia.numeric.Complex1d;
public class ProcessFFT extends IaProcessImpl implements InputListener {
private final ProcessInput _input;
private final ProcessOutput _output;
public ProcessFFT(String name) {
     super(name);
     _input = createInput("input", Complex1d.class);
     _output = createOutput("output", Complex1d.class);
     // It will always listen to events.
     _input.addInputListener( this );
 }
 public void inputHandler(ProcessInputEvent ev) {
             Product product = (Complex1d) ev.getData();
             // a doStuff method should be defined and would do the
 processing.
             Product new_product = doStuff(product);
             _output.dataReady( new_product );
  }
}
</pre>
```

Example 2: how to implement a event-based process with Launchable

```
<pre>
import herschel.ia.dataflow.*;
import herschel.ia.numeric.Complex1d;
public class ProcessFFT extends IaProcessImpl implements InputListener,
 Launchable {
private final ProcessInput _input;
private final ProcessOutput _output;
public ProcessFFT(String name) {
     super(name);
     _input = createInput("input", Complex1d.class);
     _output = createOutput("output", Complex1d.class);
 }
 public void inputHandler(ProcessInputEvent ev) {
             Product product = (Complex1d) ev.getData();
             // a doStuff method should be defined and would do the
 processing.
             Product new_product = doStuff(product);
             _output.dataReady( new_product );
  }
  public void start() {
```

Example 2: how to implement a event-based process with Launchable

```
        _input.addInputListener( this );
    }
    public void stop() {
        _input.removeInputListener( this );
     }
}
</pre>
```

Example 3: how to use a process.

```
<pre>
from myProcesses import ProcessFFT
c = Complex1d.range(100);
p = ProcessFFT("fft")
p.getConnector("input").pass(c)
p.getConnector("output").pass(c);
print c
</pre>
```

Limitations

no limitation

See also

- reference

History

- 26-5-2005 jcg: change javadoc format for help support.

3.145. IFFT

Full Name:	herschel.ia.numeric.toolbox.xform.IFFT.entry.urm.xml
Alias:	IFFT
Type:	Unknown (XML-based documentation) -
Import:	from herschel.ia.numeric.toolbox.xform import IFFT.entry.urm.xml

Description

Gives the inverse of the Fast Fourier Transform.

Example

Example 1: Apply IFF

```
from java.lang.Math import PI # Frequency modulated signal: parameters ts =
 1E-6 # Sampling period (sec) fc = 200000 # Carrier frequency (Hz) fm = 2000
 # Modulation frequency (Hz) beta = .0003 # Modulation index (Hz) n = 5000
 # Number of samples # Create signal in complex form t = Double1d.range(n)
 * ts signal = SIN(2 * PI * fc * t * (1 + beta * COS(2 * PI * fm * t)))
 z_signal=Complex1d(signal) # Get spectrum spectrum = ABS(FFT(z_signal))
 # Repeat with apodizing z_signal=Complex1d(HAMMING(signal)) spectrum2 =
 ABS(FFT(z_signal))
```

API Summary

Jython Syntax

```
<y>=IFFT(<x>)
```

Properties

```
Complex1d x [INPUT, MANDATORY, default=no default value]
```

```
Array of float, double, complex. y [INPUT, MANDATORY, default=no
default value]
```

Limitations

Does not work for Complex1d.

API details

Properties

`Complex1d x [INPUT, MANDATORY, default=no default value]`

Complex1d arrays only.

`Array of float, double, complex. y [INPUT, MANDATORY, default=no default value]`

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type. The result is in radians.

See also

- FFT

3.146. ImageAbsTask

Full Name:	herschel.ia.toolbox.image.ImageAbsTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageAbsTask

Description

A Task that takes the absolute intensity values of an image.

A Task that replaces the intensity values in an image with their absolute value.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Image absolute [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image absolute [OUTPUT, MANDATORY, default=No default value]`

The output image, being the image with the absolute values of the input image.

3.147. ImageAddTask

Full Name:	herschel.ia.toolbox.image.ImageAddTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageAddTask

Description

A Task to add images.

A Task that allows to add two images pixel-to-pixel or co-add based on the Wcs coordinates, or to add a scalar to all intensity values in an image.

API Summary

Properties
`Image image1 [INPUT, MANDATORY, default=No default value]`
`Image image2 [INPUT, OPTIONAL, default=No default value]`
`Integer ref [INPUT, OPTIONAL, default=No default value]`
`Number scalar [INPUT, OPTIONAL, default=No default value]`
`Image sum [OUTPUT, OPTIONAL, default=No default value]`

API details

Properties

`Image image1 [INPUT, MANDATORY, default=No default value]`

The first image term/addend.

`Image image2 [INPUT, OPTIONAL, default=No default value]`

The second image term/addend.

`Integer ref [INPUT, OPTIONAL, default=No default value]`

The reference frame for the calculation of the sum.

`Number scalar [INPUT, OPTIONAL, default=No default value]`

The scalar term/addend.

`Image sum [OUTPUT, OPTIONAL, default=No default value]`

The sum.

3.148. ImageAnalysis

Full Name:	herschel.ia.toolbox.image.gui.ImageAnalysis
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import ImageAnalysis

Description

A class to deal with ImageAnalysis (in the current version :

ProfilePlotting, AperturePhotometry, AreaHistogra, and ContourPlotting).

API Summary

Constructor
`ImageAnalysis(ImageAnalysisToolbox toolbox, String title)`
Standard constructor for ImageAnalysis. A new window with the

Methods
`JFrame getFrame()`
Method that returns the JFrame of this ImageAnalysis.
`ImageAnalysisToolbox getToolbox()`
Returns the ImageAnalysisToolbox, associated with this
`Container getComponent()`
Returns the Content Pane of the JFrame of this
`ArrayList getImageFigures()`
Returns all ImageFigures used for this ImageAnalysis.

API details

Constructor

`ImageAnalysis(ImageAnalysisToolbox toolbox, String title)`

given title and associated with the given ImageAnalysisToolbox is crested.

Arguments

`ImageAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

`String` **`title`** `[INPUT, MANDATORY]`

Methods

`JFrame getFrame()`

Return

JFrame

Returns the JFrame of this ImageAnalysis.

ImageAnalysisToolbox getToolbox()

ImageAnalysis.

Return

ImageAnalysisToolbox

Returns the ImageAnalysisToolbox, associated with this ImageAnalysis.

Container getComponent()

ImageAnalysis.

Return

Container

Returns the Content Pane of the JFrame of this ImageAnalysis.

ArrayList getImageFigures()

Return

ArrayList

Returns all ImageFigures used for this ImageAnalysis.

3.149. ImageAnalysisToolbox

Full Name:	herschel.ia.toolbox.image.gui.ImageAnalysisToolbox
Alias:	ImageAnalysisToolbox
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import ImageAnalysisToolbox

Description

A class to perform image analysis.

In the current version you can do the following : 1) You can draw a 2D profile plot of a straight line, starting at a point you clicked on with the mouse, and ending at the current mouse position (in the image) or at the 2nd point you clicked on with the mouse. 2) You can perform aperture photometry on the image, either with an annular or a rectangular sky aperture, and with a chosen sky estimation algorithm. 3) You can make a histogram, either of the whole image, or of a selected region (bounded by a circle, an ellipse, a rectangle or a polygon). You can choose the cut levels and the number of bins for the histogram. 4) You can perform contour plotting on the image. You can give the contour values for which you wish the contours to be generated manually or let them be calculated (then you should specify the number of contour levels, the extreme contour values and the distribution of the contour levels). You can also increase/decrease the minimum length for contours to be drawn. In all cases the image analysis is performed on the shown layer and shown in a JTabbedPane on the right hand side.

Example

Example 1: A basic example on how to open a toolbox for image analysis on an

```
ImageDataset imds :
iat = ImageAnalysisToolbox(idms)
or
iat = ImageAnalysisToolbox() iat.setImage(imds)
```

API Summary

Constructors
ImageAnalysisToolbox() The standard constructor for ImageAnalysisToolbox. This
ImageAnalysisToolbox(Image image) Constructor for ImageAnalysisToolbox. This constructor shows an
ImageAnalysisToolbox(Array2dData array2d) Constructor for ImageAnalysisToolbox. This constructor shows an
ImageAnalysisToolbox(Array3dData array3d) Constructor for ImageAnalysisToolbox. This constructor shows an

Methods
setLayer() Sets the image of this ImageAnalysisToolbox, using a file chooser.
addLayer() Sets the image of this ImageAnalysisToolbox, using a file chooser.
exit() Exits this ImageanalysisToolbox (closes (the window of) this

Methods
setImage(Image image) Sets the image you wish to analyze with this ImageAnalysisToolbox,
setImage(Array2dData array2d) Sets the image you wish to analyse with this ImageAnalysisToolbox,
setImage(Array3dData array3d) Sets the image you wish to analyze with this ImageAnalysisToolbox,
addImage(Image image) Adds the given image to the image, analyzed with
addImage(Array2dData array2d) Adds the given image to the images, analyzed with
addImage(Array3dData array3d) Adds the given image to the images, analyzed with
closeActiveTab() Closes the active tab, if there is one.
closeAllTabs() Closes all tabs, if there are any.
setEnabled(boolean enabled) Disables/enables all tabs on the tabbed pane and the menus for
boolean **isEnabled**() Returns whether the tabbed pane and menus for closing tabs and
JFrame **getFrame**() Method that returns the frame of this ImageAnalysisToolbox.
JPanel **getPanel**() Method that returns the panel of this ImageAnalysisToolbox.
JTabbedPane **getTabbedPane**() Method that returns the tabbed pane of this ImageAnalysisToolbox.
Image **getImage**() Returns the displayed image.
Display **getDisplay**() Returns the Display of the image you are analyzing with
int **getSelectedLayer**() Returns the index of the shown layer, or the index of the selected
int **getSelectedTab**() Returns the index of the tab, selected for the shown layer.
JTabbedPane **getTabOnPane**() Returns the tab (tabbed pane) on the tabbed pane, that is
boolean **isInImage**(MouseEvent me) Checks whether the given MouseEvent has occurred within the image
boolean **isInImage**(double pixX, double pixY) Checks whether the point with given pixel coordinates (pixX, pixY)
boolean **hasWCS**()

Methods
Returns true if sky coordinates are available for the image,
plotProfile2D()
Draws straight line on the image, starting at the point in the you
aperturePhotometry()
Performs aperture photometry on the image you are analyzing with
histogram()
Makes an area histogram. You can choose the kind of area you want
ArrayList **getAllFigures**()
Returns all ImageFigures used for this ImageAnalysisToolbox.
addFigures(ArrayList figures)
Adds the ImageFigures belonging to the selected tab up front to
updateImage()
Updates the image, when another tab is made active.
removeAllAnnotations()
Makes all annotations invisible.
createSpaceOnTabbedPane()
Creates a free space at the end of the ArrayList of ArrayLists of
createSpaceInTab()
Creates a free space at index 0 in the ArrayList of CanvasFigures,

API details

Constructors

ImageAnalysisToolbox()
constructor shows an empty ImageAnalysisToolbox window in 2 parts, with only a title, buttons for maximizing, minimizing and closing the window, a menu bar (File, Image, Help) and a color bar and a status bar at the bottom.

ImageAnalysisToolbox(Image image)
ImageAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, a menu bar (file, Image, Help) and a color bar and a status bar at the bottom. Also the image, you wish to analyze is displayed.
Argument
Image **image** [INPUT, MANDATORY]

ImageAnalysisToolbox(Array2dData array2d)
empty ImageAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, and a menubar (file, Image, Help). Also the image you wich to analyse, is displayed.
Argument
Array2dData **array2d** [INPUT, MANDATORY]

ImageAnalysisToolbox(Array3dData array3d)
empty ImageAnalysisToolbox window in 2 parts, with a title, buttons for maximizing, minimizing and closing the window, and a menu ar (file, Image, Help). Also the image you wish to analyze, is displayed.

ImageAnalysisToolbox(Array3dData array3d)

Argument

Array3dData **array3d** [INPUT, MANDATORY]

Methods

setLayer()

addLayer()

exit()

ImageAnalysisToolbox).

setImage(Image image)

to the given image.

Argument

Image **image** [INPUT, MANDATORY]

setImage(Array2dData array2d)

to the given ImageDataset.

Argument

Array2dData **array2d** [INPUT, MANDATORY]

setImage(Array3dData array3d)

to the given ImageDataset.

Argument

Array3dData **array3d** [INPUT, MANDATORY]

addImage(Image image)

this ImageAnalysisToolbox.

Argument

Image **image** [INPUT, MANDATORY]

addImage(Array2dData array2d)

this ImageAnalysisToolbox.

Argument

Array2dData **array2d** [INPUT, MANDATORY]

addImage(Array3dData array3d)

this ImageAnalysisToolbox.

Argument

Array3dData **array3d** [INPUT, MANDATORY]

closeActiveTab()

closeAllTabs()

setEnabled(boolean enabled)

closing tabs and performing image analysis, of this ImageAnalysisToolbox.

Argument

`boolean` **`enabled`** `[INPUT, MANDATORY]`

boolean isEnabled()

performing image analysis, of this ImageAnalysisToolbox are enabled/disabled.

Return

boolean

Returns true is the tabbed pane and menus for closing tabs and performing image analysis, of this ImageAnalysisToolbox are enabled; false otherwise.

JFrame getFrame()

Return

JFrame

The frame of this ImageAnalysisToolbox.

JPanel getPanel()

Return

JPanel

The panel of this ImageAnalysisToolbox.

JTabbedPane getTabbedPane()

Return

JTabbedPane

The tabbed pane of this ImageAnalysisToolbox.

Image getImage()

Return

Image

Returns the displayed image.

Display getDisplay()

this ImageAnalysisToolbox.

Return

Display

Returns the display of the image you are analyzing with this ImageAnalysisToolbox.

int getSelectedLayer()

tab on the tabbed pane.

int getSelectedLayer()

Return

int

Returns the index of the shown layer,or the index of the selected tab on the tabbed pane.

int getSelectedTab()

Return

int

The index of the tab, selected for the shown layer.

JTabbedPane getTabOnPane()

selected.

Return

JTabbedPane

Returns the tab (tabbed pane) on the tabbed pane, that is selected.

boolean isInImage(MouseEvent me)

we are analyzing with this ImageAnalysisToolbox.

Argument

`MouseEvent` **me** [INPUT, MANDATORY]

Return

boolean

Returns true if the given MouseEvent has occurred within the image that is being analyzed with this ImageAnalysisToolbox; false otherwise.

boolean isInImage(double pixX, double pixY)

lies in the image we are analyzing with this ImageAnalysisToolbox.

Arguments

double **pixX** [INPUT, MANDATORY]

double **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the point with given pixel coordinates (pixX, pixY) lies in the image we are analyzing with this ImageAnalysisToolbox; false otherwise.

boolean hasWCS()

analyzed with this ImageAnalysisToolbox; false otherwise.

Return

boolean

Returns true if sky coordinates are available for the image, analyzed with this ImageAnalysisToolbox; false otherwise.

plotProfile2D()

clicked on with the mouse the 1st time, and ending at the current mouse position (in the image), or at the 2nd point (in the image) you clicked on, and plots the intensity along that straight line. When you click or move outside of the image, no straight line is drawn and no plot is shown.

aperturePhotometry()

this ImageAnalysisToolbox, either with an annular sky aperture or a rectangular sky aperture, and with a chosen sky estimation algorithm.

histogram()

to make the histogram of (the whole image, or a region bounded by a circle, an ellipse, a rectangle or a polygon), given the cut levels and the number of bins for the histogram.

ArrayList getAllFigures()

Return

ArrayList

Returns all ImageFigures used for this ImageAnalysisToolbox.

addFigures(ArrayList figures)

the list of ImageFigures, used for this ImageAnalysisToolbox.

Argument

`ArrayList` **figures** `[INPUT, MANDATORY]`

updateImage()

removeAllAnnotations()

createSpaceOnTabbedPane()

ArrayLists of ImageFigures, that belong to the image analysis on the shown layer.

createSpaceInTab()

belonging to a new image analysis on the shown layer.

See also

- ???
- ???
- ???
- ???
- ???
- ???

3.150. ImageArithmeticsTask

Full Name:	herschel.ia.toolbox.image.ImageArithmeticsTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageArithmeticsTask

Description

An abstract Task for image arithmetics.

An abstract Task for image arithmetics. The subclasses/subtasks are ImageAddTask (adding), ImageSubtractTask (subtracting), ImageMultiplyTask (multiply), ImageDivideTask (dividing) and ImageModuloTask (modulo).

API Summary

Properties
`Image image1 [INPUT, MANDATORY, default=No default value]`
`Image image2 [INPUT, OPTIONAL, default=No default value]`
`Integer ref [INPUT, OPTIONAL, default=Default value : 0]`
`Number scalar [INPUT, OPTIONAL, default=No default value]`

API details

Properties

`Image image1 [INPUT, MANDATORY, default=No default value]`

The input first image.

`Image image2 [INPUT, OPTIONAL, default=No default value]`

The input second image.

`Integer ref [INPUT, OPTIONAL, default=Default value : 0]`

The input reference frame for the calculation.

`Number scalar [INPUT, OPTIONAL, default=No default value]`

The input scalar.

3.151. ImageAxis

Full Name:	herschel.ia.gui.image.ImageAxis
Type:	Java Class -
Import:	from herschel.ia.gui.image import ImageAxis
Category:	Display

Description

Axes for image display.

This class can change the look of the axes of an image display. A grid can also be enabled.

API Summary

Methods
disable() Disables the axis
enable() Enables the axis
setLabel(String newLabel) Set the label of the axis
String **getLabel**() Returns the label
setBackground(Color newBackground) Sets the Background.
Position **getOrientation**() Returns the orientation of the axis
setOrientation(Position orientation) Sets the orientation of the axis
setAxisStroke(int stroke) Sets the axis stroke
int **getAxisStroke**() Returns the stroke of the axis.
setAxisColor(Color color) Sets the color of the axis
Color **getAxisColor**() Returns the color of the axis.
setLabelFont(Font labelFont) Sets the font for the label.
Font **getLabelFont**() Returns the font for the label
Color **getLabelFontColor**() Returns the color of the label
setLabelFontColor(Color labelFontcolor)

Methods
Sets the color for the label
showColorTable(boolean showColorTable) Shows or hides the color table for this axis.
setWorldCoordinates(boolean wcs) Show or hide the world coordinates
setTickLabelFont(int newSize) Sets the size of the font for the ticks.
setTickLabelFont(Font tickLabelFont) Sets the font of the ticks
Font **getTickLabelFont**() Returns the font of the ticks.
Color **getTickLabelFontColor**() Returns the color of the font of the ticks
setTickLabelFontColor(Color color) Sets the color of the ticks font
setInnerTickLength(int length) Sets the length of the inner ticks
int **getInnerTickLength**() Returns the inner tick length
setOuterTickLength(int length) Sets the length of the outer ticks
int **getOuterTickLength**() Returns the outer tick length
setMainTicks(int ticks) Sets the number of main ticks
int **getMainTicks**() Returns the number of main ticks.
setMinorTicks(int ticks) Sets the number of minor ticks
int **getMinorTicks**() Returns the number of minor ticks.
setTickDistance(int pixels) Sets the tick distance
int **getTickDistance**() Returns the number of pixels between two ticks on the axis.
showGridLines(boolean gl) Enables or disables the grid
boolean **getShowGridLines**() Returns the status of the grid
setGridColor(Color gridColor) Sets the color of the grid

Methods
`Color getGridColor()` Returns the color of the grid

API details

Methods

`disable()`
Disables the axis

`enable()`
Enables the axis

`setLabel(String newLabel)`
The label of the axis is set. If no label is needed, an empty string should be given as label **Argument** `String newLabel [INPUT, MANDATORY]`

`String getLabel()`
Returns the label of the axis. **Return** **String** The label of the axis.

`setBackground(Color newBackground)`
Sets the Background. **Argument** `Color newBackground [INPUT, MANDATORY]`

`Position getOrientation()`
Returns the orientation of the axis **Return** **Position** The orientation (as a Position)

`setOrientation(Position orientation)`
Changes the orientation of the axis. **Argument** `Position orientation [INPUT, MANDATORY]`

`setAxisStroke(int stroke)`
Sets the stroke (line width) of the axis (in pixels) **Argument** `int stroke [INPUT, MANDATORY]`

int getAxisStroke()

Returns the stroke (line width) of the axis.

Return

int

The stroke of the axis.

setAxisColor(Color color)

Sets the color of the axis.

Argument

`Color` **color** [INPUT, MANDATORY]

Color getAxisColor()

Returns the color of the axis

Return

Color

Color The color of the axis.

setLabelFont(Font labelFont)

Sets the font for the label.

Argument

`Font` **labelFont** [INPUT, MANDATORY]

Font getLabelFont()

Returns the font for the label of the axis.

Return

Font

The font for the label.

Color getLabelFontColor()

Returns the color of the label.

Return

Color

the color of the label.

setLabelFontColor(Color labelFontcolor)

Sets the color for the label

Argument

`Color` **labelFontcolor** [INPUT, MANDATORY]

showColorTable(boolean showColorTable)

Shows (true) or hides (false) the color table for this axis.

Argument

showColorTable(boolean showColorTable)

boolean **showColorTable** [INPUT, MANDATORY]

setWorldCoordinates(boolean wcs)

Shows the world coordinates (True) or the pixel coordinates (False)

Argument

boolean **wcs** [INPUT, MANDATORY]

setTickLabelFont(int newSize)

Sets the size of the font for the ticks.

Argument

int **newSize** [INPUT, MANDATORY]

setTickLabelFont(Font tickLabelFont)

Sets the font of the ticks

Argument

Font **tickLabelFont** [INPUT, MANDATORY]

Font getTickLabelFont()

Returns the font of the ticks.

Return

Font

The font of the ticks,

Color getTickLabelFontColor()

Returns the color of the font of the ticks

Return

Color

The color of the ticks font.

setTickLabelFontColor(Color color)

Sets the color of the ticks font

Argument

Color **color** [INPUT, MANDATORY]

setInnerTickLength(int length)

Sets the distance that the ticks enter in the image

Argument

int **length** [INPUT, MANDATORY]

int getInnerTickLength()

Returns the distance that the ticks enter in the image.

Return

int

int getInnerTickLength()

The distance that the ticks enter in the image.

setOuterTickLength(int length)

Sets the distance that the ticks go out of the image

Argument

`int` **`length`** `[INPUT, MANDATORY]`

int getOuterTickLength()

Returns the distance that the ticks go out of the image.

Return

int

The distance that the ticks go out of the image.

setMainTicks(int ticks)

Sets the number of main ticks. Main ticks are labeled and are slightly longer than the other (minor) ticks. If the number of ticks is less than two, automatic selection of the number of ticks is enabled.

Argument

`int` **`ticks`** `[INPUT, MANDATORY]`

int getMainTicks()

Returns the number of main (labeled) ticks.

Return

int

The number of main ticks.

setMinorTicks(int ticks)

Sets the number of minor ticks between 2 main ticks. Minor ticks are not labeled. Negative values will take the standard value of 4 minor ticks.

Argument

`int` **`ticks`** `[INPUT, MANDATORY]`

int getMinorTicks()

Returns the number of minor (non-labeled) ticks, between two main ticks.

Return

int

The number of minor ticks between two main ticks.

setTickDistance(int pixels)

Sets the distance (in pixels) between two (minor) ticks. Negative (or 0) values will enable automatic selection of the distance.

Argument

`int` **`pixels`** `[INPUT, MANDATORY]`

int getTickDistance()

Returns the number of pixels between two ticks on the axis. A negative value means the automatic selection of the tick distance is enabled.

Return

int

int The distance (in pixels) between two ticks.

showGridLines(boolean gl)

Shows or hides the grid for this axis. The grid can only be displayed if the magnification is at least 4!

Argument

`boolean` **`gl`** `[INPUT, MANDATORY]`

boolean getShowGridLines()

Returns true if the grid is shown.

Return

boolean

True if the grid is shown

setGridColor(Color gridColor)

Sets the color of the grid for this axis.

Argument

`Color` **`gridColor`** `[INPUT, MANDATORY]`

Color getGridColor()

Returns the color of the grid

Return

Color

The color of the grid.

3.152. ImageCeilTask

Full Name:	herschel.ia.toolbox.image.ImageCeilTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageCeilTask

Description

A Task to ceil intensity values.

A Task that takes the ceiled intensity values of an image.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Image` **`ceiled`** `[OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image ceiled [OUTPUT, MANDATORY, default=No default value]`

The output ceiled image.

3.153. ImageContour

Full Name:	herschel.ia.dataset.image.ImageContour
Type:	Java Class -
Import:	from herschel.ia.dataset.image import ImageContour

Description

A class to deal with ImageContours.

API Summary

Constructors
`ImageContour()` This constructor for ImageContour creates a new ImageContour.
`ImageContour(Image image)` This constructor for ImageContour creates a new ImageContour,
`ImageContour(Image image, int nbOfContourValues)` This constructor for ImageContour creates a new ImageContour,
`ImageContour(Image image, int nbOfContourLevels, double[] extremeContourValues, String distribution)` This constructor for ImageContour creates a new ImageContour,

Methods
`addContourLevel(ContourLevel contourLevel, double contourValue)` Adds the given ContourLevel for the given contour value to this
`addContourLevel(ContourLevel contourLevel, String key)` Adds the given ContourLevel to this ImageContour with the given
`setWcs(Wcs wcs)` Sets the Wcs for this ImageContour.
`Wcs` **`getWcs()`** Returns the Wcs for this ImageContour.
`boolean` **`hasWcs()`** Checks whether a Wcs is attached to this ImageContour.
`boolean` **`hasValidWcs()`** Checks whether a valid Wcs is attached to this ImageContour.

API details

Constructors

`ImageContour()`

`ImageContour(Image image)`
associated with the given image.

ImageContour(Image image)

Argument

Image **image** [INPUT, MANDATORY]

ImageContour(Image image, int nbOfContourValues)

associated with the given image and with the given number of contour levels.

Arguments

Image **image** [INPUT, MANDATORY]

int **nbOfContourValues** [INPUT, MANDATORY]

ImageContour(Image image, int nbOfContourLevels, double[] extremeContourValues, String distribution)

associated with the given image and with the given number of contour levels, the given extreme contour values and the given distribution of the contour levels.

Arguments

Image **image** [INPUT, MANDATORY]

int **nbOfContourLevels** [INPUT, MANDATORY]

double[] **extremeContourValues** [INPUT, MANDATORY]

String **distribution** [INPUT, MANDATORY]

Methods

addContourLevel(ContourLevel contourLevel, double contourValue)

ImageContour.

Arguments

ContourLevel **contourLevel** [INPUT, MANDATORY]

double **contourValue** [INPUT, MANDATORY]

addContourLevel(ContourLevel contourLevel, String key)

key.

Arguments

ContourLevel **contourLevel** [INPUT, MANDATORY]

String **key** [INPUT, MANDATORY]

setWcs(Wcs wcs)

Argument

Wcs **wcs** [INPUT, MANDATORY]

Wcs getWcs()

Return

Wcs

Returns the Wcs for this ImageContour.

boolean hasWcs()

Return

`boolean hasWcs()`

boolean

Returns true if a Wcs is attached to this ImageContour, false otherwise.

`boolean hasValidWcs()`

Return

boolean

Returns true if a valid Wcs is attached to this ImageContour; false otherwise.

3.154. ImageDivideTask

Full Name:	herschel.ia.toolbox.image.ImageDivideTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageDivideTask

Description

A Task to divide images.

A Task that allows to divide two images pixel-to-pixel or to divide based on the Wcs coordinates, or to divide all intensity values in an image by a scalar.

API Summary

Properties
`Image image1 [INPUT, MANDATORY, default=No default value]`
`Integer ref [INPUT, OPTIONAL, default=No default value]`
`Number scalar [INPUT, OPTIONAL, default=No default value]`
`Image quotient [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image1 [INPUT, MANDATORY, default=No default value]`

The image dividend.

`Integer ref [INPUT, OPTIONAL, default=No default value]`

The reference frame for the calculation of the quotient.

`Number scalar [INPUT, OPTIONAL, default=No default value]`

The scalar divisor.

`Image quotient [OUTPUT, MANDATORY, default=No default value]`

The quotient.

3.155. ImageExp10Task

Full Name:	herschel.ia.toolbox.image.ImageExp10Task
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageExp10Task

Description

exp10, OUTPUT, Image, MANDATORY, No default value

The output image, being the exp10 scaled image.

API Summary

Property
`Image image [INPUT, MANDATORY, default=No default value]`

API details

Property

`Image image [INPUT, MANDATORY, default=No default value]`
The input image.

3.156. ImageExpNTask

Full Name:	herschel.ia.toolbox.image.ImageExpNTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageExpNTask

Description

A Task that allows to change the intensity values of an image

according to an expN scaling.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double n [INPUT, MANDATORY, default=Default value : 2.0]`
`Image expN [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Double n [INPUT, MANDATORY, default=Default value : 2.0]`

The input exponent.

`Image expN [OUTPUT, MANDATORY, default=No default value]`

The output image, being the expN scaled image.

3.157. ImageExpTask

Full Name:	herschel.ia.toolbox.image.ImageExpTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageExpTask

Description

A Task that allows to change the intensity values of an image

according to an exp scaling.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Image` **`exp`** `[OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image exp [OUTPUT, MANDATORY, default=No default value]`

The ouput image, being the exp scaled image.

3.158. ImageFloorTask

Full Name:	herschel.ia.toolbox.image.ImageFloorTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageFloorTask

Description

A Task to floor intensity values.

A Task that takes the floored intensity values of an image.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Image floored [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image floored [OUTPUT, MANDATORY, default=No default value]`

The output floored image.

3.159. ImageHistogramExplorer

Full Name:	herschel.ia.gui.image.ImageHistogramExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import ImageHistogramExplorer

Description

An explorer for ImageHistogramProducts.

API Summary

Constructors
ImageHistogramExplorer() The constructor of a new ImageHistogramExplorer.
ImageHistogramExplorer(Object object) The constructor of a new ImageHistogramExplorer associated

Methods
String **getName**() Returns the name for this ImageHistogramExplorer.
String **getDescription**() Returns the description for this ImageHistogramExplorer.
boolean **canHandle**(Class className) Checks whether this ImageHistogramExplorer can handle objects of the
setObject(Object object) Sets the object for this ImageHistogramExplorer to the given object.
ImageHistogramProduct **getObject**() Returns the object for this ImageHistogramExplorer.
Class **getVariableType**() Returns the expected variable type for this ImageHistogramExplorer.
JComponent **getComponent**() Returns the component that is responsible for displaying the
JTable **getParameterTable**() Returns the parameter table for this
JComponent **getHistogram**() Returns the histogram for this ImageHistogramExplorer.
addDataObjectListener(DataObjectListener listener) Add the given listener to this ImageHistogramExplorer to
removeDataObjectListener(DataObjectListener listener) Removes the given listener for this ImageHistogramExplorer, so

API details

Constructors

ImageHistogramExplorer()

ImageHistogramExplorer(Object object)

with the given object.

Argument

`Object` **object** [INPUT, MANDATORY]

Methods

String getName()

Return

String

Returns the name for this ImageHistogramExplorer.

String getDescription()

Return

String

Returns the description for this ImageHistogramExplorer.

boolean canHandle(Class className)

given class.

Argument

`Class` **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this ImageHistogramExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

`Object` **object** [INPUT, MANDATORY]

ImageHistogramProduct getObject()

Return

ImageHistogramProduct

Returns the object for this ImageHistogramExplorer.

Class getVariableType()

Return

Class getVariableType()

Class

Returns the expected variable type for this ImageHistogramExplorer.

JComponent getComponent()

data object for this ImageHistogramExplorer.

Return

JComponent

Returns the component that is responsible for displaying the data object for this ImageHistogramExplorer.

JTable getParameterTable()

ImageHistogramExplorer.

Return

JTable

Returns the parameter table for this ImageHistogramExplorer.

JComponent getHistogram()

Return

JComponent

Returns the histogram component for this ImageHistogramExplorer.

addDataObjectListener(DataObjectListener listener)

receive data object events from it.

Argument

`DataObjectListener` **`listener`** `[INPUT, MANDATORY]`

removeDataObjectListener(DataObjectListener listener)

that it no longer receives data object events by this explorer.

Argument

`DataObjectListener` **`listener`** `[INPUT, MANDATORY]`

3.160. ImageHistogramProduct

Full Name:	herschel.ia.dataset.image.ImageHistogramProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import ImageHistogramProduct

Description

A class to deal with the results of an image histogram.

API Summary

Constructor
`ImageHistogramProduct()` The constructor of a new ImageHistogramProduct.

Methods
`setCutLevels``(double lowCut, double highCut)` Sets the cut levels for this ImageHistogramProduct
`setNbOfBins``(int bins)` Sets the number of bins for this
`setHistogram``(Double1d values, Double1d frequencies)` Sets the histogram for this ImageHistogramProduct to a
`setUnit``(Unit unit)` Sets the unit for this ImageHistogramProduct to the given unit.
`double` **`getLowCut`**`()` Returns the minimum cut level for this
`double` **`getHighCut`**`()` Returns the maximum cut level for this
`int` **`getNbOfBins`**`()` Returns the number of bins for this
`TableDataset` **`getHistogram`**`()` Returns the histogram for this ImageHistogramProduct.
`Double1d` **`getValues`**`()` Returns the values for the histogram for this
`Double1d` **`getFrequencies`**`()` Returns the frequencies for the histogram for this
`String` **`getUnit`**`()` Returns the unit for this ImageHistogramProduct.

API details

Constructor

`ImageHistogramProduct()`

Methods

setCutLevels(double lowCut, double highCut)

to the given cut levels.

Arguments

double **lowCut** [INPUT, MANDATORY]

double **highCut** [INPUT, MANDATORY]

setNbOfBins(int bins)

ImageHistogramProduct to the given number of bins.

Argument

int **bins** [INPUT, MANDATORY]

setHistogram(Double1d values, Double1d frequencies)

histogram with the given values and frequencies.

Arguments

Double1d **values** [INPUT, MANDATORY]

Double1d **frequencies** [INPUT, MANDATORY]

setUnit(Unit unit)

Argument

Unit **unit** [INPUT, MANDATORY]

double getLowCut()

ImageHistogramProduct.

Return

double

Returns the minimum cut level for this ImageHistogramProduct.

double getHighCut()

ImageHistogramProduct.

Return

double

Returns the maximum cut level for this ImageHistogramProduct.

int getNbOfBins()

ImageHistogramProduct.

Return

int

Returns the number of bins for this ImageHistogramProduct.

TableDataset getHistogram()

Return

TableDataset getHistogram()

TableDataset

Returns the histogram for this ImageHistogramProduct.

Double1d getValues()

ImageHistogramProduct.

Return

Double1d

Returns the values for the histogram for this

Double1d getFrequencies()

ImageHistogramProduct.

Return

Double1d

Returns the frequencies for the histogram for this ImageHistogramProduct.

String getUnit()

Return

String

Returns the unit for this ImageHistogramProduct.

3.161. ImageHistogramTask

Full Name:	herschel.ia.toolbox.image.ImageHistogramTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageHistogramTask

Description

A Task to make a histogram of an image.

A Task to make a histogram of an image as a whole.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`
`Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`
`ImageHistogramProduct histogram [OUTPUT, MANDATORY, default=Default value : new ImageHistogramProduct()]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`

The lower cut level.

`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`

The upper cut level.

`Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`

The number of bins.

`ImageHistogramProduct histogram [OUTPUT, MANDATORY, default=Default value : new ImageHistogramProduct()]`

The histogram.

3.162. ImageLog10Task

Full Name:	herschel.ia.toolbox.image.ImageLog10Task
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageLog10Task

Description

A Task that allows to change the intensity values of an image according to a log10 scaling.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Image` **`log10`** `[OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image log10 [OUTPUT, MANDATORY, default=No default value]`

The ouput image, being the log10 scaled image.

3.163. ImageLogNTask

Full Name:	herschel.ia.toolbox.image.ImageLogNTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageLogNTask

Description

A Task that allows to change the intensity values of an image

according to a logN scaling.

API Summary

Properties
`Image.class` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double.class` **`n`** `[INPUT, MANDATORY, default=No default value]`
`Image.class` **`logN`** `[OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image.class image [INPUT, MANDATORY, default=No default value]`

The input image.

`Double.class n [INPUT, MANDATORY, default=No default value]`

The input n.

`Image.class logN [OUTPUT, MANDATORY, default=No default value]`

The output logN scaled image.

3.164. ImageLogTask

Full Name:	herschel.ia.toolbox.image.ImageLogTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageLogTask

Description

A Task that allows to cahnge the intensity values of an image

according to a log scaling.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Image log [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image log [OUTPUT, MANDATORY, default=No default value]`

The ouput image, being the log scaled image.

3.165. ImageModuloTask

Full Name:	herschel.ia.toolbox.image.ImageModuloTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageModuloTask

Description

A Task for modulo calculation with images.

A Task that allows to take the modulus of an image, either with another image or with a scalar.

API Summary

Properties
`Image image1 [INPUT, MANDATORY, default=No default value]`
`Image image2 [INPUT, OPTIONAL, default=No default value]`
`Integer ref [INPUT, OPTIONAL, default=No default value]`
`Number scalar [INPUT, OPTIONAL, default=No default value]`
`Image remainder [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image1 [INPUT, MANDATORY, default=No default value]`

The image dividend.

`Image image2 [INPUT, OPTIONAL, default=No default value]`

The image divisor.

`Integer ref [INPUT, OPTIONAL, default=No default value]`

The reference frame for the calculation of the modulus.

`Number scalar [INPUT, OPTIONAL, default=No default value]`

The scalar divisor.

`Image remainder [OUTPUT, MANDATORY, default=No default value]`

The remainder after division.

3.166. ImageMultiplyTask

Full Name:	herschel.ia.toolbox.image.ImageMultiplyTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageMultiplyTask

Description

A Task multiply images.

A Task that allows to multiply two images pixel-to-pixel or to multiply based on the Wcs coordinates, or to multiply all intensity values in an image with a scalar.

API Summary

Properties
`Image image1 [INPUT, MANDATORY, default=No default value]`
`Image image2 [INPUT, OPTIONAL, default=No default value]`
`Integer ref [INPUT, OPTIONAL, default=No default value]`
`Number scalar [INPUT, OPTIONAL, default=No default value]`
`Image product [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image1 [INPUT, MANDATORY, default=No default value]`

The image multiplier.

`Image image2 [INPUT, OPTIONAL, default=No default value]`

The image multiplicand.

`Integer ref [INPUT, OPTIONAL, default=No default value]`

The reference frame for the calculation of the product.

`Number scalar [INPUT, OPTIONAL, default=No default value]`

The scalar multiplicand.

`Image product [OUTPUT, MANDATORY, default=No default value]`

The product.

3.167. ImagePowerTask

Full Name:	herschel.ia.toolbox.image.ImagePowerTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImagePowerTask

Description

A Task that changes images according to a power scaling.

A Task that changes the intensity values of an image according to a power scaling.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Double n [INPUT, MANDATORY, default=Default value : 2.0]`
`Image powered [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Double n [INPUT, MANDATORY, default=Default value : 2.0]`

The input power.

`Image powered [OUTPUT, MANDATORY, default=No default value]`

The output image.

3.168. ImageRoundTask

Full Name:	herschel.ia.toolbox.image.ImageRoundTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageRoundTask

Description

A Task to round intensity values.

A Task that takes the rounded intensity values of an image.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Image rounded [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image rounded [OUTPUT, MANDATORY, default=No default value]`

The output rounded image.

3.169. ImageSaverTask

Full Name:	herschel.ia.toolbox.image.ImageSaverTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageSaverTask
Category:	task/image

Description

A Task to save images.

A task which translates and creates a grey colors JPEG file from an image, not taking into account annotations or cut levels.

Example

Example 1: Saving a grey color image to a file

```
ImageSaverTask()(image = im, filename = "test.jpg")
```

API Summary

Jython Syntax

```
ImageSaverTask()(image, filename)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
String filename [INPUT, MANDATORY, default=No default value]
```

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image to save.

`String filename [INPUT, MANDATORY, default=No default value]`

The filename for the save image.

3.170. ImageSqrtTask

Full Name:	herschel.ia.toolbox.image.ImageSqrtTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageSqrtTask

Description

A Task that changes images according to a sqrt scaling.

A Task that changes the intensity values of an image according to a sqrt scaling.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Image sqrt [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image sqrt [OUTPUT, MANDATORY, default=No default value]`

The output image, being the image changed according to a sqrt scaling.

3.171. ImageSquareTask

Full Name:	herschel.ia.toolbox.image.ImageSquareTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageSquareTask

Description

A Task that changes images according to a square scaling.

A Task that allows to change the intensity values of an image according to a square scaling.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Image square [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The input image.

`Image square [OUTPUT, MANDATORY, default=No default value]`

The output image, being the squared image.

3.172. ImageSubtractTask

Full Name:	herschel.ia.toolbox.image.ImageSubtractTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImageSubtractTask

Description

A Task to subtract images.

A Task that allows to subtract two images pixel-to-pixel or based on the Wcs coordinates, or subtract a scalar from all intensity values of an image.

API Summary

Properties
`Image image1 [INPUT, MANDATORY, default=No default value]`
`Image image2 [INPUT, OPTIONAL, default=No default value]`
`Integer ref [INPUT, OPTIONAL, default=No default value]`
`Number scalar [INPUT, OPTIONAL, default=No default value]`
`Image difference [OUTPUT, OPTIONAL, default=No default value]`

API details

Properties

`Image image1 [INPUT, MANDATORY, default=No default value]`

The image minuend.

`Image image2 [INPUT, OPTIONAL, default=No default value]`

The image subtrahend.

`Integer ref [INPUT, OPTIONAL, default=No default value]`

The reference frame for the calculation of the difference.

`Number scalar [INPUT, OPTIONAL, default=No default value]`

The scalar subtrahend.

`Image difference [OUTPUT, OPTIONAL, default=No default value]`

The difference.

3.173. importCubeTask

Full Name:	herschel.ia.toolbox.cube.ImportCubeTask
Alias:	importCubeTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import ImportCubeTask
Category:	task/cube

Description

The ImportCube task for Cubes.

ImportCubeTask is a task which imports a Cube from a fits file

Example

Example 1: Import a fits file

```
importCubeTask(cube = im, filename = "myFile.fits")
```

API Summary

Jython Syntax

```
importCubeTask(cube, filename)
```

Properties

```
Cube cube [IO, MANDATORY, default=No default value]
string filename [INPUT, MANDATORY, default=no default value]
```

API details

Properties

`Cube cube [IO, MANDATORY, default=No default value]`

The input Cube to load

`string filename [INPUT, MANDATORY, default=no default value]`

The file to import

See also

- ???

3.174. importCubeTask

Full Name:	herschel.ia.toolbox.cube.ImportSpectralCubeTask
Alias:	importCubeTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import ImportSpectralCubeTask
Category:	task/cube

Description

The ImportCube task for Cubes.

ImportCubeTask is a task which imports a Cube from a fits file

Example

Example 1: Import a fits file

```
importCubeTask(cube = im, filename = "myFile.fits")
```

API Summary

Jython Syntax

```
importCubeTask(cube, filename)
```

Properties

```
Cube cube [IO, MANDATORY, default=No default value]
string filename [INPUT, MANDATORY, default=no default value]
```

API details

Properties

`Cube cube [IO, MANDATORY, default=No default value]`

The input Cube to load

`string filename [INPUT, MANDATORY, default=no default value]`

The file to import

See also

- ???

3.175. importImage

Full Name:	herschel.ia.toolbox.image.ImportImageTask
Alias:	importImage
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ImportImageTask
Category:	task/image

Description

A Task to import images into an Image object.

A task which imports an Image from a file (bmp, fpx, gif, jpg, png, pnm, tiff or fits format). It is also possible to import a FITS file.

Example

Example 1: Import a jpg file

```
importFile(image = im, filename = "myFile.jpg")
```

API Summary

Jython Syntax

```
importFile(image, filename)
```

Properties

```
Image image [INOUT, MANDATORY, default=No default value]
string filename [INPUT, MANDATORY, default=no default value]
```

API details

Properties

`Image image [INOUT, MANDATORY, default=No default value]`

The input Image to load.

`string filename [INPUT, MANDATORY, default=no default value]`

The file to import.

3.176. importUfDirToPal

Full Name:	herschel.ia.toolbox.util.ImportUfDirToPalTask
Alias:	importUfDirToPal
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import ImportUfDirToPalTask
Category:	task

Description

Imports an observation context from an user friendly directory structure into a pool.

The importUfDirToPal task is used to ingest observations from a user friendly directory structure into a HIPE pool, so that they can be used with HIPE. The whole observation described by the XML file will be imported and a reference returned in a new variable.

Example

Example 1: Simple example

```
pref = importUfDirToPal(pool=poolDst, dirin="/sourcedir", xml="111111111-
herschel.ia.obs.ObservationContext-0.xml")
```

API Summary

Jython Syntax

```
pref = importUfDirToPal(<pool>, <dirin>, <xml>)
```

Properties

```
ProductPool pool [INPUT, MANDATORY, default=No default value]
String dirin [INPUT, MANDATORY, default=No default value]
String xml [INPUT, MANDATORY, default=No default value]
ProductRef productRef [OUTPUT, MANDATORY, default=no default
value]
```

Limitations

The whole observation described by the XML file will be imported.

API details

Properties

ProductPool pool [INPUT, MANDATORY, default=No default value]

The pool to export products from.

String dirin [INPUT, MANDATORY, default=No default value]

The directory where the user friendly structure of products resides.

String xml [INPUT, MANDATORY, default=No default value]
The xml file that describes the observation to import.

ProductRef productRef [OUTPUT, MANDATORY, default=no default value]
The Product ref imported to the pool.

See also

- ???

History

- 16-01-09 Created
- `2009-03-17 - Added`: output

3.177. info

Full Name:	herschel.ia.inspector.InfoTask
Alias:	info
Type:	Java Task -
Import:	from herschel.ia.inspector import InfoTask
Category:	task/general

Description

A task for inspecting the content of a variable in the jython session.

Users can inspect a variable by calling info from the command line. The task opens a pop up window displaying the structure of the specifed item.

Examples

Example 1: info of myvariable

```
info("myvariable")
```

Example 2: info of myvariable setting a refresh rate of 2 seconds (2000 milliseconds)

```
info("myvariable", 2000)
```

API Summary

Jython Syntax
info("item")
info("item", 5)

Properties
String **item** [INPUT, YES, default=No default value]
Object **refresh** [INPUT, NO, default=5000]

Limitations

No limitation

Miscellaneous

No miscellaneous

API details

Properties

String item [INPUT, YES, default=No default value]

The item to display the structure

Object refresh [INPUT, NO, default=5000]
The refresh rate (milliseconds)

See also

- references

History

- `2004-12-05 - NdC`: first release.

3.178. Int1d

Full Name:	herschel.ia.numeric.Int1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Int1d

Description

A rectangular numeric int array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.179. Int2d

Full Name:	herschel.ia.numeric.Int2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Int2d

Description

A rectangular numeric int array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.180. Int3d

Full Name:	herschel.ia.numeric.Int3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Int3d

Description

A rectangular numeric int array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.181. Int4d

Full Name:	herschel.ia.numeric.Int4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Int4d

Description

A rectangular numeric int array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.182. Int5d

Full Name:	herschel.ia.numeric.Int5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Int5d

Description

A rectangular numeric int array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.183. IntegratedMapDisplay

Full Name:	herschel.ia.gui.cube.IntegratedMapDisplay
Type:	Java Class -
Import:	from herschel.ia.gui.cube import IntegratedMapDisplay

Description

A class to deal with IntegratedMap.

API Summary

Methods
`setBegin(Double begin)` Sets the begin of the straight line, associated with this
`setEnd(Double end)` Sets the end of the straight line, associated with this
`ImageFigure getLine()` Returns the straight line (ImageFigure), of which we wish to plot
`Double getBegin()` Returns the begin of the drawn line in PixelCoordinates.
`Double getEnd()` Returns the end of the drawn line in PixelCoordinates.
`ArrayList getIntegratedMaps()` Returns the end of the drawn line in PixelCoordinates.
`String getSkyCoordinates()` Returns the String version of the sky coordinates

API details

Methods

`setBegin(Double begin)`

Velocity position map to the given point (in UserCoordinates).

Argument

`Double` **`begin`** `[INPUT, MANDATORY]`

`setEnd(Double end)`

Velocity position map to the given point (in UserCoordinates).

Argument

`Double` **`end`** `[INPUT, MANDATORY]`

`ImageFigure getLine()`

the intensity in this IntegratedMapDisplay.

Return

`ImageFigure getLine()`

ImageFigure

The straight line (ImageFigure), of which we wish to plot the intensity in this IntegratedMapDisplay.

`Double getBegin()`

Return

Double

Returns the begin of the drawn line in PixelCoordinates.

`Double getEnd()`

Return

Double

Returns the end of the drawn line in PixelCoordinates.

`ArrayList getIntegratedMaps()`

Return

ArrayList

Returns the end of the drawn line in PixelCoordinates.

`String getSkyCoordinates()`

(WorldCoordinates) of the begin and end of the drawn straight line.

Return

String

Returns the String version of the sky coordinates (WorldCoordinates) of the begin and end of the drawn straight line.

3.184. IntegrateMapFromCubeTask

Full Name:	herschel.ia.toolbox.cube.IntegrateMapFromCubeTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import IntegrateMapFromCubeTask

Description

Task which integrate one or more "map" or images from a Cube.

In the CubeSpectrumAnalysisToolbox (via the integratedmapdisplay GUI), this task integrate one or more set of contiguous images defined by ranges. The output of this task is a set of simpleImages

API Summary

Properties
`Cube` **`cube`** `[INPUT, MANDATORY, default=no default value]`
`integer` **`bbStack`** `[INPUT, default value 1, default=no default value]`
`Double2d` **`startArray`** `[INPUT, MANDATORY, default=no default value]`
`Double2d` **`endArray`** `[INPUT, MANDATORY, default=no default value]`
`SimpleImages[]` **`imageArray`** `[OUTPUT, MANDATORY, default=no default value]`

API details

Properties

`Cube cube [INPUT, MANDATORY, default=no default value]`

The Cube containing the spectra to extract

`integer bbStack [INPUT, default value 1, default=no default value]`

give the number of image to inegrate

`Double2d startArray [INPUT, MANDATORY, default=no default value]`

The Array of spectral values for the start of each integration the dimension of this array is the number of integration to execute

`Double2d endArray [INPUT, MANDATORY, default=no default value]`

The Array of spectral values for the end of each integration the dimension of this array is the number of integration to execute

`SimpleImages[] imageArray [OUTPUT, MANDATORY, default=no default value]`

an array of simpleimages containning the result of each integration

See also

- ???

- ???

History

- 2009-05-20 - AG: creation

3.185. Integrator

Full Name:	herschel.ia.numeric.toolbox.integr.Integrator
Alias:	Integrator
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.integr import Integrator

Description

Interface for all integrators.

Available integrators are: - GaussHermiteIntegrator (closed interval) - GaussianQuad4Integrator (open interval) - GaussianQuad5Integrator (open interval) - GaussJacobiIntegrator (closed interval) - GaussLaguerreIntegrator (closed interval) - GaussLegendreIntegrator (open interval) - RectangularIntegrator (open interval) - TrapezoidalIntegrator (open interval) - SimpsonIntegrator (open interval) - RombergIntegrator (open interval)

Example

Example 1: Integration using the Romberg's method

```
from herschel.ia.numeric.all import *
class MyFunction(RealFunction):
  def calc(self,x):
  return x*x
f = MyFunction()
i = RombergIntegrator(-3, 3)
print i.integrate(f)   # 18.0
```

API Summary

Jython Syntax

```
<r>=SomeOpenIntervalIntegrator(<a>,<b>).integrate(<f>)
<r>=SomeShutIntervalIntegrator().integrate(<f>)
```

Properties

```
double a [INPUT, MANDATORY, default=no default value]
double b [INPUT, MANDATORY, default=no default value]
RealFunction f [INPUT, MANDATORY, default=no default value]
double r [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

double a [INPUT, MANDATORY, default=no default value]

Lower limit of integration, only mandatory for open interval integrators.

double b [INPUT, MANDATORY, default=no default value]

Upper limit of integration, only mandatory for open interval integrators. It must be greater or equal than the lower limit.

RealFunction f [INPUT, MANDATORY, default=no default value]
The function must implement the calc method; see example below.

double r [OUTPUT, MANDATORY, default=no default value]
Returns the integral of the given function between the specified limits; that is, the area behind the function within these limits.

3.186. InverseDFT2dTask

Full Name:	herschel.ia.toolbox.image.InverseDFT2dTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import InverseDFT2dTask

Description

A Task for two dimensional inverse Fast Fourier Transforms.

A Task that calculates the inverse 2D Discrete Fourier Transform and the power spectrum.

API Summary

Properties
`Numeric2dData image [INPUT, MANDATORY, default=No default value]`
`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`
`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Numeric2dData image [INPUT, MANDATORY, default=No default value]`

The input image.

`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`

The transform.

`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

The spectrum.

3.187. InverseFFT2dTask

Full Name:	herschel.ia.toolbox.image.InverseFFT2dTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import InverseFFT2dTask

Description

A Task for two dimensional inverse Fast Fourier Transforms.

A Task that calculates the inverse 2D Fast Fourier Transform and the power spectrum.

API Summary

Properties
`Numeric2dData image [INPUT, MANDATORY, default=No default value]`
`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`
`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Numeric2dData image [INPUT, MANDATORY, default=No default value]`

The input image.

`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`

The transform.

`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

The spectrum.

3.188. INVERSE

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixInverse
Alias:	INVERSE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixInverse

Description

Returns the inverse of a square matrix.

Example

Example 1: Apply INVERSE to a Float2d matrix

```
x=Float2d([ [1,2],[3,4] ])
print INVERSE(x) # [ [-2.0,1.0], [1.5,-0.5] ]
```

API Summary

Jython Syntax
<y>=INVERSE(<x>)

Properties
any square matrix **x** [INPUT, MANDATORY, default=no default value]
double **y** [INPUT, NOT_MANDATORY, default=false]

Miscellaneous

Does not work for complex matrices.

API details

Properties

any square matrix x [INPUT, MANDATORY, default=no default value]
Any square matrix

double y [INPUT, NOT_MANDATORY, default=false]
Returns a double

3.189. IS_FINITE

Full Name:	herschel.ia.numeric.toolbox.basic.IsFinite
Alias:	IS_FINITE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import IsFinite

Description

Returns a boolean array where each element is true if the corresponding element input array is a finite number, false otherwise.

Example

Example 1: Apply IS_FINITE on a Double1d

```
x=Double1d([1,Double.NaN,Double.NEGATIVE_INFINITY,Double.POSITIVE_INFINITY])
print IS_FINITE(x) # [true,false,false,false]
```

API Summary

Jython Syntax

```
<y>=IS_FINITE (<x>)
```

Properties

```
any array or number x [INPUT, MANDATORY, default=no default value]
```

```
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element input array is a finite number, false otherwise

See also

- IS_NAN
- IS_INFINITE

3.190. IS_INFINITE

Full Name:	herschel.ia.numeric.toolbox.basic.IsInfinite
Alias:	IS_INFINITE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import IsInfinite

Description

Returns a boolean array where each element is true if the corresponding element

input array is infinitely large in magnitude, false otherwise.

Example

Example 1: Apply IS_INFINITE on a Double1d

```
x=Double1d([1,Double.NaN,Double.NEGATIVE_INFINITY,Double.POSITIVE_INFINITY])
print IS_INFINITE(x) # [false,false,true,true]
```

API Summary

Jython Syntax

```
<y>=IS_INFINITE (<x>)
```

Properties

```
any array or number x [INPUT, MANDATORY, default=no default value]
```

```
boolean array or a boolean y [OUTPUT, MANDATORY, default=no
default value]
```

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element input array is input array is infinitely large in magnitude, false otherwise.

See also

- IS_NAN
- IS_FINITE

3.191. IS_NAN

Full Name:	herschel.ia.numeric.toolbox.basic.IsNaN
Alias:	IS_NAN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import IsNaN

Description

Returns a boolean array where each element is true if the corresponding element

input array is flagged as Not a Number, false otherwise.

Example

Example 1: Apply IS_FINITE on a Double1d

```
x=Double1d([1,Double.NaN,Double.NEGATIVE_INFINITY,Double.POSITIVE_INFINITY])
print IS_NAN(x) # [false,true,false,false]
```

API Summary

Jython Syntax

```
<y>=IS_NAN(<x>)
```

Properties

```
any array or number x [INPUT, MANDATORY, default=no default value]
```

```
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element of the input array is flagged as Not a Number, false otherwise.

See also

- IS_FINITE
- IS_INFINITE

3.192. KURTOSIS

Full Name:	herschel.ia.numeric.toolbox.basic.Kurtosis
Alias:	KURTOSIS
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Kurtosis

Description

Yields the kurtosis of the elements in the input array.

The kurtosis is the fourth moment divided by the standard deviation raised to the fourth power. In addition, this implementation subtracts 3 since 3 is the kurtosis for a normal distribution.

Example

Example 1: Apply KURTOSIS on a Float1d

```
x=Float1d([1,3,2,3,4])
print KURTOSIS(x) # -1.74840236686
```

API Summary

Jython Syntax

```
<y>=KURTOSIS(<x>)
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
double y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double y [OUTPUT, MANDATORY, default=no default value]

Returns a double

See also

- MEAN
- MEDIAN
- SKEWNESS
- STDDEV

- VARIANCE

3.193. LayerStruct

Full Name:	herschel.ia.gui.explorer.table.LayerStruct
Type:	Java Class -
Import:	from herschel.ia.gui.explorer.table import LayerStruct

Description

LayerStruct is a data structure used in OverPlotter/TablePlotter. It has many methods which allow users to access extracted data and flags. It has many setters (setXXX) and getters (getXXX). It works as a bookkeeping as well. It keeps tracks on the properties of the active layer. It saves all the personalities of an active layer when the layer becomes inactive and re-display them when the layer becomes active.

API Summary

Constructors
`LayerStruct(TableDataset table)`
`LayerStruct(paramType table, String layerName)`
`LayerStruct(TableDataset table, Bool2d flags)`
`LayerStruct(TableDataset table, Bool1d flags)`

Methods
`TableDataset getExtractedTableDataset()`
`BooleanArray getFlags()`

API details

Constructors

`LayerStruct(TableDataset table)`

Argument

`TableDataset table [INPUT, MANDATORY, default=no default value]`

The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.

`LayerStruct(paramType table, String layerName)`

Arguments

`paramType table [INPUT, MANDATORY, default=no default value]`

The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.

`String layerName [INPUT, MANDATORY, default=no default value]`

The layerName must be unique for each table.

LayerStruct(TableDataset table, Bool2d flags)

Arguments

TableDataset **table** [INPUT, MANDATORY, default=no default value]

The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.

Bool2d **flags** [INPUT, MANDATORY, default=no default value]

The flags must have the same rank and size as the table parameter

LayerStruct(TableDataset table, Bool1d flags)

Arguments

TableDataset **table** [INPUT, MANDATORY, default=no default value]

The input table contains one or more column and each column can have numeric or none numeric data or multi-dimension. However, such columns will be filtered out when the table is plotted.

Bool1d **flags** [INPUT, MANDATORY, default=no default value]

The the length of the flags array must equal to the number of columns of the input table

Methods

TableDataset getExtractedTableDataset()

Return

TableDataset

- return the extracted dataset

BooleanArray getFlags()

Return

BooleanArray

- return the flags. The flags can be either Bool1d or Bool2d.

History

- 2006-11-06 - first: version
- 2008-04-22 re-factored this code
- 2008-12-14 - adding: URM documentation

3.194. LevenbergMarquardtFitter

Full Name:	herschel.ia.numeric.toolbox.fit.LevenbergMarquardtFitter
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.fit import LevenbergMarquardtFitter

Description

An introduction to the use of fit package can be found

here. At least once have a look at the background on the organization and structure of the package.

Example

Example 1: The fit/demo directory contains worked

```
<a href="../../../ia/numeric/toolbox/fit/demo/jython.html">examples</a>
on almost all aspects of of the package.
```

Limitations

1. LevenbergMarquardtFitter is **not** guaranteed to find the global minimum.

2. The calculation of the evidence is an Gaussian approximation which is only exact for linear models with a fixed scale.

3.195. LinearInterpolator

Full Name:	herschel.ia.numeric.toolbox.interp.LinearInterpolator
Alias:	LinearInterpolator
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.interp import LinearInterpolator

Description

Creates an linear interpolation function from a set of knots (x,y), that can

be applied to numeric arrays of rank 1.

Example

Example 1: Create and apply a LinearInterpolator on a Double1d

```
# create some knots
x=Double1d.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
f=LinearInterpolator(x,SQUARE(x))
u=Double1d([1,1.5,2,2.5,3,3.5])
print f(u) # [1.0,2.5,4.0,6.5,9.0,12.5]
```

API Summary

Jython Syntax

```
<f>=LinearInterpolator(<x>,<y>[,allowExtrapolation])
```

Properties

```
Double1d x [INPUT, MANDATORY, default=no default value]
Double1d y [INPUT, NOT_MANDATORY, default=false]
boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]
```

API details

Properties

`Double1d x [INPUT, MANDATORY, default=no default value]`

The knots are Double1d

`Double1d y [INPUT, NOT_MANDATORY, default=false]`

The knots are Double1d

`boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]`

Extrapolation is not allowed by default. Boolean.TRUE allows extrapolation.

See also

- CubicSplineInterpolator

- NearestNeighborInterpolator

3.196. ListContext

Full Name:	herschel.ia.pal.ListContext
Type:	Java Class -
Import:	from herschel.ia.pal import ListContext

Description

Groups products (or other Contexts that in turn group products) in a

list-like structure. Products grouped in a ListContext can be subsequently retrieved by an index (with index=0 corresponding to the first product in the list), through the 'refs' method. Note: users may be confused as to why there is a need to have to go though a 'refs' method to add or access products from a ListContext. This is due to a technical limitation in the design which will be addressed in due course.

Examples

Example 1: Adding a product to a ListContext product = Product() ref =

```
storage.save(product) # the ref is a ProductRef object listcontext =
ListContext() listcontext.refs.add(ref)
```

Example 2: Getting the first product from a ListContext ref_first =

```
lcontext.refs.get(0) product = ref_first.product
```

Example 3: Saving a ListContext to ProductStorage (same way as any other

```
product) ref_context = storage.save(listcontext)
```

API Summary

Method
`List getRefs()`
get the list of ProductRefs stored. From this list, you can put

API details

Method

`List getRefs()`

products into the ListContext, or retrieve products by index.

Return

List

A list of product refs.

Examples

Putting a product into the the ListContext ref =

```
storage.save(product) # the ref is a ProductRef object
```

```
List getRefs()
```

```
listcontext.refs.add(ref)
```

Getting the first product from the ListContext ref_first =

```
lcontext.refs.get(0)
```

See also

- *Product Access Layer*

3.197. localStoreWriter

Full Name:	herschel.ia.toolbox.util.LocalStoreWriterTask
Alias:	localStoreWriter
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import LocalStoreWriterTask
Category:	task

Description

Saves a product in a Local Store, either already defined, or a new

one chosen by name by the user.

Example

Example 1: Saving a product into a local store

```
p = Product()
store = PoolManager.getPool("mib")
localStoreWriter(product = p, store = store)
```

API Summary

Jython Syntax

```
localStoreWriter(<product>,<store>)
```

Properties

```
Product product [INPUT, MANDATORY, default=null]
String store [INPUT, MANDATORY, default=null]
```

API details

Properties

`Product product [INPUT, MANDATORY, default=null]`

Product to be saved.

`String store [INPUT, MANDATORY, default=null]`

LocalStore id where to save the product.

History

- 2008-11-11 - JSS: first release
- 2009-02-20 - JDS: cleanup

3.198. LOG10

Full Name:	herschel.ia.numeric.toolbox.basic.Log10
Alias:	LOG10
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Log10

Description

Gives the logarithm with base 10 of a number or a numeric array: $y=LOG_{10}(x)$.

Example

Example 1: Apply LOG10 on a Double1d

```
x=Double1d([1,10,100])
print LOG10(x)        # [0.0,0.9999999999999999,1.9999999999999998]
print ROUND(LOG10(x)) # [0.0,1.0,2.0]
```

API Summary

Jython Syntax

```
<y>=LOG10(<x>)
```

Properties

```
any array or number x [INPUT, MANDATORY, default=no default value]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no
default value]
```

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array (or a number if the input is not an array) where each element is the the logarithm with base 10 of the corresponding element of the input array

See also

- LOG
- LogN

3.199. LOG

Full Name:	herschel.ia.numeric.toolbox.basic.Log
Alias:	LOG
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Log

Description

Computes the function $y=LOG_e(x)$, the natural logarithm.

Example

Example 1: Apply LOG on a Double1d

```
x=Double1d([0,1])
print LOG(EXP(x)) #  [0.0,1.0]
```

API Summary

Jython Syntax
`<y>=LOG(<x>)`

Properties
`any array or number x [INPUT, MANDATORY, default=no default value]`
`boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]`

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array (or a number if the input is not an array) where each element is the the natural logarithm of the corresponding element of the input array

See also

- LOG10
- LogN

3.200. LogN

Full Name:	herschel.ia.numeric.toolbox.basic.LogN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import LogN

Description

Gives the logarithm with base N of a number or a numeric array: $y=LOG_n(x)$.

Example

Example 1: Apply LogN on a Int1d

```
x=2**Int1d.range(6)           # [1,2,4,8,16,32]
print Int1d(ROUND(LogN(2)(x))) # [0,1,2,3,4,5]
print Int1d(ROUND(x.apply(LogN(2)))) # [0,1,2,3,4,5]
```

API Summary

Jython Syntax

```
<y>=LogN(<base>)(<x>)
```

Properties

```
any array or number x [INPUT, MANDATORY, default=no default value]
a number base [INPUT, MANDATORY, default=no default value]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no
default value]
```

Limitations

Does not work for complex values.

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

a number base [INPUT, MANDATORY, default=no default value]

The base n

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array (or a number if the input is not an array) where each element is the the logarithm with base n of the corresponding element of the input array

See also

- LOG
- LOG10

3.201. Long1d

Full Name:	herschel.ia.numeric.Long1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Long1d

Description

A rectangular numeric long array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.202. Long2d

Full Name:	herschel.ia.numeric.Long2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Long2d

Description

A rectangular numeric long array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.203. Long3d

Full Name:	herschel.ia.numeric.Long3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Long3d

Description

A rectangular numeric long array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.204. Long4d

Full Name:	herschel.ia.numeric.Long4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Long4d

Description

A rectangular numeric long array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.205. Long5d

Full Name:	herschel.ia.numeric.Long5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Long5d

Description

A rectangular numeric long array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.206. LUDecomposition

Full Name:	herschel.ia.numeric.toolbox.matrix.LUDecomposition
Alias:	LUDecomposition
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.matrix import LUDecomposition

API Summary

Jython Syntax

`A=Double2d()`
`B=Double2d()`
`res=B.apply(LUDecomposition(A))`

`lud = LUDecomposition(A)`
`res = B.apply(lud)`
`lowerTriangularFactor = lud.l`
`uppperTrangularFactor = lud.u`
`determinant = lud.det`
`integerPivot = lud.pivot`
`doublePivot = lud.doublePivot`
`flag = lud.isSingular();`

Property
`Double2d A [INPUT, MANDATORY, default=no default value]`

API details

Property

`Double2d A [INPUT, MANDATORY, default=no default value]`
Input must be a Double2d or Float2d array.

3.207. ManualContourPanel

Full Name:	herschel.ia.toolbox.image.gui.ManualContourPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import ManualContourPanel

Description

A panel for the ManualContourTask.

API Summary

Constructor
ManualContourPanel() The constructor of a new ManualContourPanel.

Methods
getButtonPanel() Returns the button panel for this ManualContourPanel.
setSiteEventHandler(SiteEventHandler handler) Sets the site event handler for this ManualContourPanel to the
setTask(TaskApi task) Associates the given task to this ManualContourPanel.
setVariableSelection(VariableSelection selection) Sets the variable selection for this ManualContourPanel to the given
Map **getSelectionMap**() Returns the selection map associated with this ManualContourPanel.
SiteEventHandler **getHandler**() Returns the site event handler associated with this
TaskApi **getTask**() Returns the task associated with this
Double1d **getContourValues**() Returns the contour values for this ManualContourPanel.
DefaultListModel **getContourValuesModel**() Returns the default list model for the contour values for this

API details

Constructor

ManualContourPanel()

Methods

getButtonPanel()

setSiteEventHandler(SiteEventHandler handler)

given site event handler.

Argument

`SiteEventHandler` **handler** `[INPUT, MANDATORY]`

setTask(TaskApi task)

Argument

`TaskApi` **task** `[INPUT, MANDATORY]`

setVariableSelection(VariableSelection selection)

variable selection.

Argument

`VariableSelection` **selection** `[INPUT, MANDATORY]`

Map getSelectionMap()

Return

Map

Returns the selection map associated with this ManualContourPanel.

SiteEventHandler getHandler()

ManualContourPanel.

Return

SiteEventHandler

Returns the site event handler associated with this ManualContourPanel.

TaskApi getTask()

ManualContourPanel.

Return

TaskApi

Returns the task associated with this ManualContourPanel.

Double1d getContourValues()

Return

Double1d

Returns the contour values for this ManualContourPanel.

DefaultListModel getContourValuesModel()

ManualContourPanel.

Return

DefaultListModel

Returns the default list model for this contour values for this ManualContourPanel.

3.208. ManualContourTask

Full Name:	herschel.ia.toolbox.image.ManualContourTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ManualContourTask

Description

A Task for making contours.

A Task for making an ImageContour for a given image for a given list of contour values.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double1d` **`values`** `[INPUT, MANDATORY, default=No default value]`
`ImageContour` **`contours`** `[OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double1d values [INPUT, MANDATORY, default=No default value]`

The contour values.

`ImageContour contours [OUTPUT, MANDATORY, default=No default value]`

The ImageContour.

3.209. ManyToOneSpectrumTask

Full Name:	herschel.ia.toolbox.spectrum.ManyToOneSpectrumTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import ManyToOneSpectrumTask

Description

Abstract base class for tasks processing {@link SpectrumContainer} data structures or arrays of such by typically applying the processing on a suitable selection of point spectra included in the containers.

Various selection mechanisms are in place that allow selecting specific individual spectra for the processing.

This family of operations typically reduces the spectra to a single spectrum, hence we call it a 'many-to-one operation'. Typical example is the average.

API Summary

Properties
`Object` **`ds`** `[INPUT, OPTIONAL, default=no default value.]`
`Object` **`selection`** `[INPUT, OPTIONAL, default=None.]`
`PyDictionary|Map>` **`selection_lookup`** `[INPUT, OPTIONAL, default=no default value.]`
`PyList` **`selection_index`** `[INPUT, OPTIONAL, default=No default value.]`
`Object` **`segments`** `[INPUT, OPTIONAL, default=no default value.]`
`Boolean` **`overwrite`** `[INPUT, OPTIONAL, default=False.]`
`String` **`variant`** `[INPUT, OPTIONAL, default=no default value.]`
`SpectrumContainer` **`result`** `[OUTPUT, OPTIONAL, default=no default value]`

API details

Properties

`Object ds [INPUT, OPTIONAL, default=no default value.]`

Input data to be processed by the task. Several types are possible:

- SpectrumContainer
- Array of SpectrumContainer, i.e. SpectrumContainer[]
- List of SpectrumContainer, i.e. List
- Any product with implementations of `SpectrumContainer` inside.

`Object selection [INPUT, OPTIONAL, default=None.]`

Specify what point spectra to restrict to. Different ways to specify these selections are possible:

Object selection [INPUT, OPTIONAL, default=None.]

- Specify a list of indices (in jython) of the point spectra for which the processing should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments to restrict to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

History

- `2009-05-20 - meli`: Initial.

3.210. MapContext

Full Name:	herschel.ia.pal.MapContext
Type:	Java Class -
Import:	from herschel.ia.pal import MapContext

Description

Groups products (or other Contexts that in turn group products) in a

map-like structure. Products grouped in a MapContext can be subsequently retrieved by key, through the 'refs' method. Note: users may be confused as to why there is a need to have to go though a 'refs' method to add or access products from a MapContext. This is due to a technical limitation in the design which will be addressed in due course.

Examples

Example 1: Adding a product to a MapContext product = Product() ref =

```
storage.save(product) # the ref is a ProductRef object mapcontext =
MapContext() mapcontext.refs.put("john",ref)
```

Example 2: Getting a product from a MapContext ref_john =

```
mapcontext.refs.get("john") product = ref_john.product
```

Example 3: Saving a MapContext to ProductStorage (same way as any other

```
product) ref_context = storage.save(mapcontext)
```

API Summary

Method
`Map getRefs()` get the 'map' of ProductRefs stored. From this 'map', you can put

API details

Method

`Map getRefs()`

products into the MapContext, or retrieve products by key.

Return

Map

A map of product refs.

Examples

Putting a product into the the MapContext ref =

```
storage.save(product) # the ref is a ProductRef object
```

`Map getRefs()`

```
storage.save(product) # the ref is a ProductRef object
mapcontext.refs.put("john", ref)
```

Getting the product from the MapContext with key "john"

```
ref_john = lcontext.refs.get("john")
```

See also

- *Product Access Layer*

3.211. MATMUL

Full Name:	herschel.ia.numeric.toolbox.matrix.doc.MATMUL.entry.urm.xml
Alias:	MATMUL
Type:	Unknown (XML-based documentation) -
Import:	from herschel.ia.numeric.toolbox.matrix.doc import MATMUL.entry.urm.xml

Description

Returns matrix multiplication.

This wrapper is deprecated. Please, use the MatrixMultiply Java class.

Limitations

This is a Jython wrapper function for the numeric MatrixMultiply(b)(A).

See also

- MatrixMultiply

3.212. MatrixMultiply

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixMultiply
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixMultiply

Description

Performs matrix multiplication of numeric or logical matrices.

Examples

Example 1: Different syntax forms

```
# spelled out:
c=MatrixMultiply(b)(a)
c=a.apply(MatrixMultiply(b))
&amp;nbsp;
# reusing an instance of a matrix-multiplier:
f=MatrixMultiply(b)
c=f(a)
c=a.apply(f)
```

Example 2: Matrix multiplication examples

```
[1 2 3]        [1 2]
A=[2,3,4]  and B=[2 3]
               [3 4]
X=[1 2]    and Y=[1 2 3]
                          [14 20]
A.apply(MatrixMultiply(B)) = [20 29]
x.apply(MatrixMultiply(A)) = [5 8 11]
A.matrixMultiply(y)        = [14 20]
```

API Summary

Jython Syntax

```
<y>=MatrixMultiply(<b>)(<a>)
```

Properties

```
Array1dData or Array2dData a [INPUT, MANDATORY, default=no default value]
```

```
Array1dData or Array2dData b [INPUT, MANDATORY, default=no default value]
```

Miscellaneous

Complex arrays are not supported yet.

API details

Properties

Array1dData or Array2dData a [INPUT, MANDATORY, default=no default value]

Input must be a rank one or rank two array of any type, but String or Complex.

Array1dData or Array2dData b [INPUT, MANDATORY, default=no default value]

Input must be an array of the same type as 'a'. If 'a' has rank one, 'b' must have rank two. If 'b' has rank one, 'a' must have rank two.

3.213. MatrixSolve

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixSolve
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixSolve

Description

Solves systems of linear equations of type A x = b.

A system of equations is a collection of equations that you deal with all together at once.

Example

Example 1: Apply MatrixSolve

```
# Solve
#  | 1  2 | |x1|   | 8|
#  |      | |  | = |  |
#  | 3  4 | |x2|   |18|
#
A=Float2d([ [1,2],[3,4] ])
b=Double1d([8.0,18.0])
print MatrixSolve(b)(A) # [2.0,3.0]
print A.apply(MatrixSolve(b))
```

API Summary

Jython Syntax

```
<x>=MatrixSolve(<b>)(<A>)
```

Properties

```
any matrix A [INPUT, MANDATORY, default=no default value]
Double1d b [INPUT, MANDATORY, default=no default value]
```

Miscellaneous

Does not work for complex matrices.

API details

Properties

any matrix A [INPUT, MANDATORY, default=no default value]

Any matrix

Double1d b [INPUT, MANDATORY, default=no default value]

Input must be a Double1d array.

3.214. MAX

Full Name:	herschel.ia.numeric.toolbox.basic.Max
Alias:	MAX
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Max

Description

Yields the numerically largest element in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply MAX on a Int2d

```
x=Int2d( [ [1,2], [-1,3] ])
print MAX(x)    # 3
print MAX(x, 0) # [1,3]
print MAX(x, 1) # [2,3]
```

API Summary

Jython Syntax

```
<y>=MAX(<x>, [,<dim>])
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
integer dim [INPUT, MANDATORY, default=no default value]
float or double array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array can be any scalar array.

integer dim [INPUT, MANDATORY, default=no default value]

The dimension to compute the calculation along

float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a scalar of the same type as the type of the input array.

See also

- MIN

3.215. MEAN

Full Name:	herschel.ia.numeric.toolbox.basic.Mean
Alias:	MEAN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Mean

Description

Yields the average value of the elements in the input array.

Example

Example 1: Apply MEAN on a Float1d

```
x=Float1d([1,3,2,3,4])
print MEAN(x) # 2.6
```

API Summary

Jython Syntax

```
<y>=MEAN(<x>)
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
double y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double y [OUTPUT, MANDATORY, default=no default value]

Returns a double

See also

- MEDIAN
- STDDEV
- VARIANCE

3.216. MeanSmoothingTask

Full Name:	herschel.ia.toolbox.image.MeanSmoothingTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import MeanSmoothingTask

Description

A Task to smooth an image by applying a mean filter.

A Task to smooth an image using the average (mean) filter. The average filter computes the sum of all pixel values in the filter window and then divides the sum by the number of pixels in the filter window. In order to filter pixels located near the edges of the image.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]`
`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image before smoothing.

`Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]`

The width of the filter window.

`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

The image after smoothing.

3.217. MEDIANDEV

Full Name:	herschel.ia.numeric.toolbox.basic.MedianAbsoluteDeviation
Alias:	MEDIANDEV
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import MedianAbsoluteDeviation

Description

The median standard deviation is one of several ways to estimate an error of the median.

Algorithm:

For an array **data** the algorithm calculates the new array | data[i] - median(data) | for all values i (| indicates the absolute, positve value of the difference). The algorithm returns the median of the resulting array.

Example

Example 1: apply MedianStandardDeviation to an Int1d array

```
data = Int1d.range(9)
median = MEDIAN(data)
dev = data.apply( MedianStandardDeviation(median) )
```

API Summary

Properties
any 1-5d array of integral type **x** [INPUT, MANDATORY, default=no default value]
the median of the input array x **median** [INPUT, MANDATORY, default=no default value]

API details

Properties

any 1-5d array of integral type x [INPUT, MANDATORY, default=no default value]

the median of the input array x median [INPUT, MANDATORY, default=no default value]

3.218. MEDIAN

Full Name:	herschel.ia.numeric.toolbox.basic.Median
Alias:	MEDIAN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Median

Description

Yields the central value of the elements in integral and floating point arrays.

Example

Example 1: Apply MEDIAN on a Float1d

```
x=Float1d([1,3,2,3,4])
print MEDIAN(x) # 3.0
```

API Summary

Jython Syntax
<y>=MEDIAN(<x>)

Properties
any scalar array **x** [INPUT, MANDATORY, default=no default value]
double **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double y [OUTPUT, MANDATORY, default=no default value]

Returns a double

See also

- MEAN
- STDDEV
- VARIANCE

3.219. MedianSmoothingTask

Full Name:	herschel.ia.toolbox.image.MedianSmoothingTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import MedianSmoothingTask

Description

A Task to smooth an image by applying a median filter.

A Task to smooth an image using the median filter. The median filter computes the median of all pixel values in the filter window.

API Summary

Properties
`Image image [INPUT, MANDATORY, default=No default value]`
`Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]`
`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image before smoothing.

`Integer width [INPUT, MANDATORY, default=Default value : Integer(3)]`

The width of the filter window.

`Image smoothed [OUTPUT, MANDATORY, default=No default value]`

The image after smoothing.

3.220. MetaQuery

Full Name:	herschel.ia.pal.query.MetaQuery
Type:	Java Class -
Import:	from herschel.ia.pal.query import MetaQuery

Description

Meta data query formulates a query on the meta data of a Product.

Typically this type of query is slower than an Attribute Query, but faster than a full query on the Product Access Layer.

Example

Example 1: Example of an query on meta data q=MetaQuery(MyProduct.class,"p",

```
"p.meta['creator'].value == 'Me'")
```

See also

- Querying

3.221. MIN

Full Name:	herschel.ia.numeric.toolbox.basic.Min
Alias:	MIN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Min

Description

Yields the numerically smallest element in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply MIN on a Int2d

```
x=Int2d( [ [1,2], [-1,3] ])
print MIN(x)    # -1
print MIN(x, 0) # [-1, 2]
print MIN(x, 1) # [ 1,-1]
```

API Summary

Jython Syntax

```
<y>=MIN(<x>, [,<dim>])
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
integer dim [INPUT, MANDATORY, default=no default value]
float or double array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array can be any scalar array.

integer dim [INPUT, MANDATORY, default=no default value]

The dimension to compute the calculation along

float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a scalar of the same type as the type of the input array.

See also

- MAX

3.222. MosaicTask

Full Name:	herschel.ia.toolbox.image.MosaicTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import MosaicTask

Description

A Task for making mosaics.

A Task to make mosaics in a naive way (i.e. by simple co-adding).

API Summary

Properties
`ArrayList` **`images`** `[INPUT, MANDATORY, default=No default value]`
`boolean` **`oversample`** `[INPUT, OPTIONAL, default=Default value : true]`
`SimpleImage` **`mosaic`** `[OUTPUT, MANDATORY, default=No default value]`

Miscellaneous

Two blank maps are created : one to represent the total signal and one to represent the total exposure. For each pixel in each image, the pixel value is added into the total signal map and its exposure (or one) in the total exposure map (if not flagged out), taking only the overlap into account. After all pixels in each image have been mapped, the total signal map is divided by the total exposure map to produce the mosaic.

API details

Properties

`ArrayList images [INPUT, MANDATORY, default=No default value]`

The input images.

`boolean oversample [INPUT, OPTIONAL, default=Default value : true]`

Indication whether oversampling should be done.

`SimpleImage mosaic [OUTPUT, MANDATORY, default=No default value]`

The mosaic.

3.223. MultiplySpectrum

Full Name:	herschel.ia.toolbox.spectrum.MultiplySpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import MultiplySpectrum

Description

Task for multiplying the flux data included in a spectrum container with a scalar or for multiplying two spectrum containers with each other on a spectrum-by-spectrum basis.

For the scalar mode, 'ds' and 'param' need to specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1,3,4,2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1,3,4]` and for the pair operation `segments1=[1,3,4], segments1=[3,6,5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the orginal datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import MultiplySpectrum
multiply = MultiplySpectrum()
multipliedByFactor = multiply(ds=spectra, param=2.1)
multipliedByFactor = multiply(ds=spectra, selection={"bbtype":[6031]},
 param=2.1)
multipliedByFactor = multiply(ds=spectra, selection=[0,1,2,3], param=2.1)
multipliedByFactor = multiply(ds=spectra, selection={"Chopper":
([-4.4,5.9],0.2), "bbtype":[6031]}, param=2.1)
multipliedByFactor = multiply(ds=spectra, selection={"LoFrequency":
(4000.0,5000.0)}, param=2.1)
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2)
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2, selection={"bbtype":
[6031]})
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2, selection=[0,1,2,3])
```

Example 1: from Jide:

```
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2, selection={"Chopper":
([-4.4,5.9],0.2), "bbtype":[6031]})
multipliedPairWise = multiply(ds1=spectra1, ds2=spectra2,
 selection={"LoFrequency":(4000.0,5000.0)})
```

API Summary

Properties
SpectrumContainer **ds** [INPUT, OPTIONAL, default=no default value.]
Double **param** [INPUT, OPTIONAL, default=no default value.]
Object **selection** [INPUT, OPTIONAL, default=None.]
PyDictionary|Map> **selection_lookup** [INPUT, OPTIONAL, default=no default value.]
PyList **selection_index** [INPUT, OPTIONAL, default=No default value.]
Boolean **overwrite** [INPUT, OPTIONAL, default=False.]
SpectrumContainer **ds1** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **ds2** [INPUT, OPTIONAL, default=no default value.]
Object **segments** [INPUT, OPTIONAL, default=no default value.]
Object **segments1** [INPUT, OPTIONAL, default=no default value.]
Object **segments2** [INPUT, OPTIONAL, default=no default value.]
String **variant** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]

Input container to be processed by the task in case the task is used as a scalar operation.

Double param [INPUT, OPTIONAL, default=no default value.]

The scalar parameter to be considered when the task is used as scalar operation.

Object selection [INPUT, OPTIONAL, default=None.]

Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra for which the add should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.

Object selection [INPUT, OPTIONAL, default=None.]

- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also

- SpectrumTask

History

- `2007-08-17 - meli`: initial.
- `2008-03-16 - meli`: major refactoring of the toolbox.
- `2008-10-03 - meli`: segments, javadoc

3.224. NearestNeighborInterpolator

Full Name:	herschel.ia.numeric.toolbox.interp.NearestNeighborInterpolator
Alias:	NearestNeighborInterpolator
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.interp import NearestNeighborInterpolator

Description

Creates an linear interpolation function from a set of knots (x,y),

that can be applied to numeric arrays of rank 1.

Example

Example 1: Create and apply a NearestNeighborInterpolator on a Double1d

```
# create some knots
x=Double1d.range(10) # [0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0]
f=NearestNeighborInterpolator(x,SQUARE(x))
u=Double1d([1,1.1,2,2.6,3,3.9])
print f(u) # [1.0,1.0,4.0,9.0,9.0,16.0]
```

API Summary

Jython Syntax

```
<f>=NearestNeighborInterpolator(<x>,<y>[,allowExtrapolation])
```

Properties

```
Double1d x [INPUT, MANDATORY, default=no default value]
Double1d y [INPUT, NOT_MANDATORY, default=false]
boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]
```

API details

Properties

`Double1d x [INPUT, MANDATORY, default=no default value]`

The knots are Double1d

`Double1d y [INPUT, NOT_MANDATORY, default=false]`

The knots are Double1d

`boolean allowExtrapolation [INPUT, NOT_MANDATORY, default=false]`

Extrapolation is not allowed by default. Boolean.TRUE allows extrapolation.

See also

- CubicSplineInterpolator

- LinearInterpolator

3.225. Normalize

Full Name:	herschel.ia.numeric.toolbox.basic.Normalize
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Normalize

Description

This functionality normalizes sets of data.

Normalization can be done by multiplying the independent variable, by a scaling factor so that the data will have a user-specified peak value or mean value over a user-specified range of dependent variables.

Input data sets is a list of vectors containing N elements, where a single vector can be written as (x1,x2,x3,...,xN-1, y). The Normalize class provides four valid options of normalization.

-Option 1: It multiplies the y values such that the peak y value will have a user-specified constant value.

-Option 2: It normalizes the y values such that the peak y value will be same as the peak value of a user-specified "fiducial" vector set.

-Option 3: It normalizes the y values so that the mean of the y values over a user-specified x-range will have a user-specified constant value.

-Option 4: It normalizes the y values so that the mean of the y values over a user-specified x-range will be same as the mean of a user-specified "fiducial" vector set over the x-range.

Notes:

-The x-range must be specified as a range in all N-1 dimensions.

-The dimentions of input data must be same as the dimentions of the fiducial array.

-When the peak/mean value of input data set equals to zero, an error [condition] is issued.

-When the peak/mean value of input data set has different sign from user-supplied peak/mean value, an error [condition] is issued.

-When there are no data within the x-range within the fiducial data set, an error [condition] is issued.

-The routine can handle any dimension N where N >= 2, and can handle all real and integer data types.

Example

Example 1: from herschel.ia.numeric import *

```
from herschel.ia.numeric.toolbox.basic import Normalize
# Perform normalizations using four valid types.
# Type 1: Normalize y values to the common peak, a user-specified constant.
# Input data array.
arr = Double2d([(0.,1.,8.,40.),(10.,10.,4.,60.),(2.,0.,4.,120.)])
# User specified peak value.
peak = 45.
# Normalize.
norm1 = Normalize(peak,Normalize.PEAK_CNST)(arr)
# Check output.
print norm1
# Output: [[0.0,1.0,8.0,15.0],[10.0,10.0,4.0,22.5],[2.0,0.0,4.0,45.0]]
# Type 2: Normalize y values to the max of a user-supplied "fiducial" array.
# Input data array.
arr = Int2d([(0,1,8,40),(10,10,4,60),(2,0,4,120)])
```

Example 1: from herschel.ia.numeric import *

```
# User supplied fiducial array.
fiducial = Int2d([(1,1,1,10),(2,2,8,30),(4,6,10,20)])
# Normalize.
norm2 = Normalize(fiducial,Normalize.PEAK_FIDUCIAL)(arr)
# Check output.
print norm2
# Output: [[0,1,8,10],[10,10,4,15],[2,0,4,30]]
# Type 3: Normalize y values to have a user-specified constant mean value
# over a x range.
# Input data array.
arr = Double2d([(0.,1.,8.,40.),(1.,10.,4.,60.),(2.,0.,4.,120.)])
# User specified mean value.
mean = 25.
# User specified x range array.
xrange = Double2d([(0.,0.,0.),(2.,5.,10.)])
norm3 = Normalize(mean,xrange,Normalize.MEAN_CNST)(arr)
# Check output.
print norm3
# Output: [[0.0,1.0,8.0,12.5],[1.0,10.0,4.0,18.75],[2.0,0.0,4.0,37.5]]
# Type 4: Normalize y values to the mean value of a user-supplied fiducial
# array over a x range.
# Input data array.
arr = Double2d([(0.,1.,8.,40.),(10.,10.,4.,60.),(2.,0.,4.,120.)])
# User supplied fiducial array.
fiducial = Double2d([(1.,1.,1.,15.),(2.,2.,8.,45),(4.,6.,10.,25.)])
# User specified x range array.
xrange = Double2d([(0.,0.,0.),(3.,3.,10.)])
# Normalize.
norm4 = Normalize(fiducial,xrange,Normalize.MEAN_FIDUCIAL)(arr)
# Check results.
print norm4
# Output: [[0.0,1.0,8.0,15.0],[10.0,10.0,4.0,22.5],[2.0,0.0,4.0,45.0]]
```

API Summary

Jython Syntax

```
<result>=Normalize(<a>,[<b>,]<type>)(<input>)
```

Properties

its value depends on the Normalize type specified **a** [INPUT, MANDATORY, default=no default value]

its value depends on the Normalize type specified **b** [INPUT, OPTIONAL, default=no default value]

Normalization type **type** [INPUT, MANDATORY, default=no default value]

Input data sets **input** [INPUT, MANDATORY, default=no default value]

the normalized array **result** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

its value depends on the Normalize type specified a [INPUT, MANDATORY, default=no default value]

-Normalize.PEAK_CNST type: common peak value to use when normalizing the input values ('' **argument is not required). -Normalize.PEAK_FIDUCIAL type: 'fiducial' array**

its value depends on the Normalize type specified a [INPUT, MANDATORY, default=no default value]

where the maximun value to use when normalizing the input values, is extracted ('' argument is not required). -Normalize.MEAN_CNST type: mean value, over an 'x' range (argument ''), to be obtained when normalizing the input values (requires '' argument). -Normalize.MEAN_FIDUCIAL type: 'fiducial' array where the mean value, over an 'x' range (argument ''), to be obtained when normalizing the input values, is extracted (requires '' argument).

its value depends on the Normalize type specified b [INPUT, OPTIONAL, default=no default value]

-Normalize.MEAN_CNST type: 'x' range where the normalized input values must have the specified mean value (argument '). -Normalize.MEAN_FIDUCIAL type: 'x' range where the normalized input values must have the specified mean value extracted from a fiducial array (argument ').

Normalization type type [INPUT, MANDATORY, default=no default value]

-Normalize.PEAK_CNST: It multiplies the 'y' values such that the peak 'y' value will have a user-specified constant value -Normalize.PEAK_FIDUCIAL: It normalizes the 'y' values such that the peak 'y' value will be same as the peak value of a user-specified "fiducial" vector set. -Normalize.MEAN_CNST: It normalizes the 'y' values so that the mean of the 'y' values over a user-specified x-range will have a user-specified constant value. -Normalize.MEAN_FIDUCIAL: It normalizes the 'y' values so that the mean of the 'y' values over a user-specified x-range will be same as the mean of a user-specified "fiducial" vector set over the x-range.

Input data sets input [INPUT, MANDATORY, default=no default value]

List of vectors containing N elements, where a single vector can be written as (x1,x2,x3,...,xN-1, y)

the normalized array result [OUTPUT, MANDATORY, default=no default value]

ie: norm[(x1_1,x2_1,x3_1,...,xN-1_1, yNorm_1), (x1_2,x2_2,x3_2,...,xN-1_2, yNorm_2), ...]

3.226. NotPresent

Full Name:	herschel.ia.numeric.toolbox.basic.NotPresent
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import NotPresent

Description

Tests whether none of the bits in specified mask are present in the elements of the input array.

The input array must be of integral type (e.g. bytes,integers) and the result is a boolean array.

Example

Example 1: Apply NotPresent on a Int1d: test if a binary 'and' comparison, item by item, with '3' (as mask) is equals to '0'

```
x=Int1d([0,1,2,3])
print NotPresent(3)(x)
#=> [   (0 & 3) == 0,   (1 & 3) == 0,   (2 & 3) == 0,   (3 & 3) == 0   ] =
 [true,false,false,false]
print x.apply(NotPresent(3)) #Another way for using NotPresent
```

API Summary

Jython Syntax

```
<y>=NotPresent(<bitmask>)(<x>)
```

Properties

```
any array of integral type x [INPUT, MANDATORY, default=no default value]
any array of booleans bitmask [INPUT, MANDATORY, default=no default value]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any array of integral type x [INPUT, MANDATORY, default=no default value]

The input array must be of integral type (e.g. bytes,integers)

any array of booleans bitmask [INPUT, MANDATORY, default=no default value]

An array of booleans

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns a boolean array (or a boolean if the input is not an array) where each element is true if the corresponding element in the input array is present, false otherwise.

See also

- AllPresent
- AnyPresent

3.227. NumberedDataset

Full Name:	herschel.ia.dataset.spectrum.NumberedDataset
Type:	Java Class -
Import:	from herschel.ia.dataset.spectrum import NumberedDataset
Category:	Datasets

Description

NumberedDataset is an alternative for the non-existing VectorDataset.

NumberedDataset is an extension of CompositeDataset, which contains datasets identified by number. It contains an iterator over the stored datasets.

Example

Example 1: In Jide:

```
vds = NumberedDataset()        # create a NumberedDataset
s1 = Spectrum1d()             # create some Datasets
s2 = Spectrum2d()
s3 = ArrayDataset( )
vds.set( s1 )             # set s1 at number "1"
vds.set( s2 )             # set s2 at number "2"
vds.set( s3, 4 )             # set s3 at number "4"
s4 = vds.get( 1 )             # s4 equals s1
print vds.getCount()         # yields 3 (3 sets in vds)
print vds.getLastIndex()         # yields 4 (last one is at 4)
it = vds.iterator()          # iterator over the datasets.
while it.hasNext(): print it.next().__class__   # print classes
vds.remove( 2 )             # leaves sets at 1 and 4
vds.collapse()            # renumbers sets to 1 and 2
vds.remove()            # removes last one, leaves only nr 1
```

Limitations

NumberedDataset still **is** a CompositeDataset and can be addressed as such. If you do so, the special methods of NumberDataset are **not** guarandeed to work.

History

- 06-03-2006 DK.

3.228. ObservationContext

Full Name:	herschel.ia.obs.ObservationContext
Type:	Java Class -
Import:	from herschel.ia.obs import ObservationContext

Description

An Observation Context is a container of Products applicable to an

specific observation. It provides associations to products which are specific to a single observation (e.g. Telemetry Product, and reduced data products) as well as associations to Products that are appliable to multiple observations (such as the calibration products).

Example

Example 1: from herschel.ia.obs import *

```
from herschel.share.fltdyn.time import FineTime
from herschel.ia.pal import MapContext
obs = ObservationContext()
auxContext = MapContext()
obs.auxiliary=auxContext #error
print obs.isInitialized() #0 (false)
obs.id=1L
obs.odNumber=2L
obs.instrument="HIFI"
obs.modelName="0"
obs.startTime=FineTime(1L)
obs.endTime=FineTime(2L)
print obs.isInitialized() #1 (true)
obs.auxiliary=auxContext #ok
print obs.auxiliary #{description="Unknown", meta=[type, creator, creationDate,
 instrument, modelName, startDate, endDate], datasets=[], history=None,
 refs=[]}
productContext = MapContext()
obs.level['level0']=productContext   #Error
print obs.isPrepared() #0 (false)
obs.calibration = MapContext()
print obs.isPrepared() #1 (true)
obs.level['level0']=productContext   #ok
print obs.level['level0']    #{description="Unknown", meta=[type, creator,
 creationDate, instrument, modelName, startDate, endDate], datasets=[],
 history=None, refs=[]}
print obs.isReduced() #0 (false)
obs.level['level1']=MapContext()
print obs.isReduced() #1 (true)
```

API Summary

Jython Syntax

```
<obs>=ObservationContext()
```

Property

```
ObservationContext obs [OUTPUT, MANDATORY, default=no default
value]
```

API details

Property

```
ObservationContext obs [OUTPUT, MANDATORY, default=no default value]
```

Returns an empty ObservationContext

See also

- ???
- MapContext
- ProductRef
- ???

3.229. OpDayGenerator

Full Name:	herschel.ia.pg.od.OpDayGenerator
Type:	Java Class -
Import:	from herschel.ia.pg.od import OpDayGenerator

History

- 19 Oct 2007: better exeption handling.
- 19 Nov 2007: add PCAL plugin

3.230. openFile

Full Name:	herschel.ia.toolbox.util.OpenFileTask
Alias:	openFile
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import OpenFileTask
Category:	task

Description

opens a variable in a viewer

Allows to open via script viewers associated to variables. TODO: document viewers

Examples

Example 1: Opening a jythom script with the editor

```
openFile("/home/user/myscript.py")
```

Example 2: Opening a fits file with its default viewer

```
openFile("/home/user/product.fits")
```

API Summary

Jython Syntax

```
openFile(<variable> [, <viewer>])
```

Properties

```
String file [INPUT, MANDATORY, default=null]
String viewer [INPUT, OPTIONAL, default=null]
```

API details

Properties

`String file [INPUT, MANDATORY, default=null]`

The path of the file to open

`String viewer [INPUT, OPTIONAL, default=null]`

The ID of the viewer to open the file with

History

- 2009-02-08 - JDS: first release

3.231. openVariable

Full Name:	herschel.ia.toolbox.util.OpenVariableTask
Alias:	openVariable
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import OpenVariableTask
Category:	task

Description

opens a variable in a viewer

Allows to open via script viewers associated to variable types. TODO: document viewers

Example

Example 1: Opening a product variable with its default viewer

```
p = Product()
openVariable("p")
```

API Summary

Jython Syntax

```
openVariable(<variable> [, <viewer>])
```

Properties

```
String variable [INPUT, MANDATORY, default=null]
String viewer [INPUT, OPTIONAL, default=null]
```

API details

Properties

`String variable [INPUT, MANDATORY, default=null]`

Path of the file to be opened

`String viewer [INPUT, OPTIONAL, default=null]`

Viewer to open the variable with

History

- `2008-12-15 - JDS`: first release
- `2008-12-16 - JDS`: added optional parameter viewer

3.232. OverPlotter

Full Name:	herschel.ia.gui.explorer.table.OverPlotter
Type:	Java Class -
Import:	from herschel.ia.gui.explorer.table import OverPlotter

Description

Overview of OverPlotter

```
OverPlotter is an extension of TablePlotter and  allow users to overlay dat\
a on top of
each other and to compare. The OverPlotter can be seen as stacking many tra\
nsparent
TablePlotters as layers on top of each other. Most TablePlotter features  w\
ork in
OverPlotter, especially when working on an individual layer. OverPlotter la\
yers have three
states, active, secondary active and in active.   Each OverPlotter layer ha\
s
its own personalities.  The personalities are unchanged no matter the layer\
 is active,
secondary active and inactive. \
```

Example

Example 1: Invoke OverPlotter in command line

```
<pre>
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.gui.explorer.table import *
from herschel.share.component import *
fits=FitsArchive();
path="/home/zhang/Development/data/"
filename = "loadCurve.fits"
try:
    fits.reader = FitsArchive.HCSS_READER
    p = fits.load(path+filename)
except java.io.IOException:
    fits.reader = FitsArchive.STANDARD_READER
    p = fits.load(path+filename)
table =p.default
wm=WindowManager.getDefault()
#Load OverPlotter
overPlotter=OverPlotter(table)
wm.addWindow('test', overPlotter.component, 1)
#add a new layer
table1=table
overPlotter.object=table1
</pre>
```

API Summary

Constructors
OverPlotter()
The default constructor
OverPlotter(TableDataset tds)

Constructors
A constructor

Methods
`JComponent` **`getComponent()`**
This method will return OverPlotter as a pluggable component
`String` **`getDescription()`**
`String` **`getName()`**
`setObject``(TableDataset data)`
Initiate a new instance of OverPlotter or add a new layer to the existing OverPlotter.

API details

Constructors

`OverPlotter()`
This constructor is used to initializes the _layerCounter to 0.

`OverPlotter(TableDataset tds)`
This constructor is used to initiate an instance of TablePlotter from the command line.
Argument
`TableDataset` **`tds`** `[INPUT, MANDATORY]`
Example
- Invoke OverPlotter in command line

```
from herschel.ia.gui.explorer.table import OverPlotter  #import OverPlotter
opl=OverPlotter(tbs)  #dts is a TableDataset defined somewhere else
```

Methods

`JComponent getComponent()`
This method allows TablePlotter to be used as a plug-ins. The user can plug OverPlotter to his/her own applications.
Return
JComponent
the OverPlotter as a component
Example
Use OverPlotter as a plug-in

```
<pre>
from herschel.share.component import *
from javax.swing import *
from java.awt import *
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.dataset.gui import *
from herschel.ia.gui.explorer.table import OverPlotter
fits=FitsArchive();
#Change to your data path
```

JComponent getComponent()

```
path="/home/zhang/Development/data/"
filename = "loadCurve.fits"
#load the table
try:
  fits.reader = FitsArchive.HCSS_READER
  p = fits.load(path+filename)
except java.io.IOException:
  fits.reader = FitsArchive.STANDARD_READER
   p = fits.load(path+filename)
#create a TableDataset
table =p.default
#create a container to hold the controls/components
pane=JPanel()
#set the layout, there are many different kinds users can choose according
 to his/her need
pane.setLayout(BoxLayout(pane, BoxLayout.Y_AXIS))
#add control one, a label
title = JLabel("Please see OverPlotter below")
pane.add(title)
#Add OverPlotter to the pane
overPlotter=OverePlotter(table)
pane.add(overPlotter.component)
pane.setPreferredSize(Dimension(overPlotter.component.width,
 overPlotter.component.height))
#add to your application
frame =JFrame("OverPlotter as Plug-in demo");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
frame.getContentPane().add(pane, BorderLayout.CENTER)
frame.pack
frame.visible=1
</pre>
```

String getDescription()

Return

String

the description of the OverPlotter

String getName()

Return

String

the name of this explorer

setObject(TableDataset data)

When this method is called the first time, it will initiate a new instance of OverPlotter and then assign the class variable _activeOverPlotter to the newly created object. When it is called again, it will add a new OverPlotter layer on to the existing OverPlotter object, _activeOverPlotter.

Argument

`TableDataset` **`data`** `[INPUT, MANDATORY, default=no default value]`

data is a TableDataset. It will be passed as an active data structure to be plotted.

Example

Create a new OverPlotter with one layer and then add the second layer

```
<pre>
 from herschel.ia.gui.explorer.table import OverPlotter
 overPlotter = OverPlotter(dataset1)
```

setObject(TableDataset data)

```
overPlotter.object = dataset2
</pre>
```

History

- `2008-10-02 - First`: release
- `2009-01-14 - Implemented`: SPR-5598 and SPR-5783

3.233. PackedMask

Full Name:	herschel.ia.numeric.toolbox.mask.PackedMask
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.mask import PackedMask

Description

This mask handling class compresses the mask to minimise memory usage. This

implementation has no limit on the number of masks that can be created. Note that the mask data is stored internally by this class.

Example

Example 1: Create a packed mask array and set and unset at a specific location.

```
GLITCH = PackedMask ("Glitch", [1024,1024,3])
print GLITCH.isSet ([5,5,2])  # 0
GLITCH.set ([5,5,2])
print GLITCH.isSet ([5,5,2])  # 1
GLITCH.unset ([5,5,2])
print GLITCH.isSet ([5,5,2])  # 0
```

3.234. PacketSequence

Full Name:	herschel.binstruct.PacketSequence
Alias:	PacketSequence
Type:	Java Class -
Import:	from herschel.binstruct import PacketSequence
Category:	binstruct

Description

a container for telemetry and telecommand source packets

In principle the `PacketSequence` is a general purpose container for telemetry and telecommand packets. However most of the time it will be used to group all the information of one observation or test because those packets have a natural connection.

3.235. pause

Full Name:	herschel.ia.toolbox.util.PauseTask
Alias:	pause
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import PauseTask
Category:	task

Description

Pause the execution of jython code

The pause task is used to pause the execution of jython. When executed a debug dialog appears displaying the current local namespace in a inspector window and a new console for executing statements into the localized namespace.

Users can add/alter the values in the namespace, the values affects the code still to be executed.

Examples

Example 1: Pause the execution of jython code

```
pause()
```

Example 2: pause the execution of my_function beetwen the assignemnt of x and

```
its printing, assigment done to x from the debugger window are reflected by the
 print
def my_function():
 x=100
 pause()
 print x
my_function()
```

Limitations

no Limitation

Miscellaneous

No miscellaneous

See also

- pause

History

- `2007-10-11 - NdC`: first release.

3.236. pointHistoryDisplay

Full Name:	herschel.ia.toolbox.pointing.PointHistoryDisplayTask
Alias:	pointHistoryDisplay
Type:	Java Task -
Import:	from herschel.ia.toolbox.pointing import PointHistoryDisplayTask
Category:	utility task

Description

Task for displaying graphs of pointing information

This takes a PointingProduct and produces graphs of the data contained in the product.

Example

Example 1: PointHistoryDisplayTask

```
<pre>
 ...
 pp = pool.getProduct(urn)
 phdt = PointHistoryDisplayTask()
 graphs = String1d(["RAC-Time", "RAG-Time"])
 phdt(pp, graphs, "mosaic")
 ...
</pre>
```

API Summary

Jython Syntax

```
PointHistoryDisplayTask()(pp, graphs, mosaic)
see examples
```

Properties

```
PointingProduct PointingProduct [INPUT, true, default=null]
String1d plotPairs [INPUT, true, default=null]
String layout [INPUT, true, default="mosaic"]
```

API details

Properties

`PointingProduct PointingProduct [INPUT, true, default=null]`

PointingProduct

`String1d plotPairs [INPUT, true, default=null]`

String1d containing strings specifying parameter pairs to be plotted. These are pairs of Strings separated by a hyphen.

Allowed Strings are

String1d plotPairs [INPUT, true, default=null]

- "Time"
- "RAC" (Commanded RA)
- "RAG" (Gyro-propagated RA)
- "RAF" (Filtered RA)
- "DecC" (Commanded Dec)
- "DecG" (Gyro-propagated RA)
- "DecF" (Filtered RA)
- "PAC" (Commanded Position angle)
- "PAG" (Gyro-propagated position angle)
- "PAF" (Filtered position angle)
- "AngVel1" (first angular velocity value)
- "AngVel2" (second angular velocity value)
- "AngVel3" (third angular velocity value)

String layout [INPUT, true, default="mosaic"]

String with value "mosaic" or "overlay" to specify how multiple plots should be organised.

3.237. Polygon

Full Name:	herschel.ia.toolbox.image.Polygon
Type:	Java Class -
Import:	from herschel.ia.toolbox.image import Polygon

Description

A polygon shape.

API Summary

Constructors
`Polygon()` The construction of a new Polygon without coordinates.
`Polygon(double[] coords)` The construction of a new Polygon with the given vertices
`Polygon(double x, double y)` The construction of a new Polygon with a single starting point.
`Polygon(int size)` The construction of a new Polygon with space for the given

Method
`boolean` **`contains`**`(double x, double y)` Checks whether the point with the given pixel coordinates is inside this Polygon.

Miscellaneous

This class extends the existing diva.util.java2d.Polygon2D.Double, because the contains() method was implemented incorrectly there.

API details

Constructors

`Polygon()`

`Polygon(double[] coords)`

in the format [x0, y0, x1, y1,...].

Argument

`double[]` **`coords`** `[INPUT, MANDATORY]`

`Polygon(double x, double y)`

Arguments

`double` **`x`** `[INPUT, MANDATORY]`

`double` **`y`** `[INPUT, MANDATORY]`

`Polygon(int size)`

number of vertices.

Argument

`int` **`size`** `[INPUT, MANDATORY]`

Method

`boolean contains(double x, double y)`

Arguments

`double` **`x`** `[INPUT, MANDATORY]`

`double` **`y`** `[INPUT, MANDATORY]`

Return

boolean

Return true if the point with the given pixel coordinates is inside this Polygon; false otherwise.

3.238. PolygonHistogramExplorer

Full Name:	herschel.ia.gui.image.PolygonHistogramExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import PolygonHistogramExplorer

Description

An explorer for PolygonHistogramProducts.

API Summary

Constructors
PolygonHistogramExplorer() The constructor of a new PolygonHistogramExplorer.
PolygonHistogramExplorer(Object object) The constructor of a new PolygonHistogramExplorer associated

Methods
String **getName()** Returns the name for this PolygonHistogramExplorer.
String **getDescription()** Returns the description for this PolygonHistogramExplorer.
boolean **canHandle(Class className)** Checks whether this PolygonHistogramExplorer can handle objects of the
setObject(Object object) Sets the object for this PolygonHistogramExplorer to the given object.
PolygonHistogramProduct **getObject()** Returns the object for this PolygonHistogramExplorer.
Class **getVariableType()** Returns the expected variable type for this PolygonHistogramExplorer.
JTable **getParameterTable()** Returns the parameter table for this

API details

Constructors

PolygonHistogramExplorer()

PolygonHistogramExplorer(Object object)

with the given object.

Argument

Object **object** [INPUT, MANDATORY]

Methods

String getName()

Return

String

Returns the name for this PolygonHistogramExplorer.

String getDescription()

Return

String

Returns the description for this PolygonHistogramExplorer.

boolean canHandle(Class className)

given class.

Argument

Class **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this PolygonHistogramExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

Object **object** [INPUT, MANDATORY]

PolygonHistogramProduct getObject()

Return

PolygonHistogramProduct

Returns the object for this PolygonHistogramExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this PolygonHistogramExplorer.

JTable getParameterTable()

EllipseHistogramExplorer.

Return

JTable

Returns the parameter table for this EllipseHistogramExplorer.

3.239. PolygonHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.PolygonHistogramPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import PolygonHistogramPanel

Description

A panel for the PolygonHistogramTask.

API Summary

Constructor
PolygonHistogramPanel() The construction of a new PolygonHistogramPanel.

Methods
drawFigure() Draws the rectangle on the image associated with this
updateFigure() Updates the polygon associated with this
trigger() Triggers the execution of the task associated
updateHistogram() Updates the histogram associated with this

API details

Constructor

PolygonHistogramPanel()

Methods

drawFigure()
RectangleHistogramPanel.

updateFigure()
PolygonHistogramPanel.

trigger()
with this PolygonHistogramPanel.

updateHistogram()
PolygonHistogramPanel.

3.240. PolygonHistogramProduct

Full Name:	herschel.ia.dataset.image.PolygonHistogramProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import PolygonHistogramProduct

Description

A class to deal with the results of a polygon histogram.

API Summary

Constructor
PolygonHistogramProduct() The constructor of a new PolygonHistogramProduct.

Methods
setEdges(Double1d edgesPixels) Sets the edges for this PolygonHistogramProduct to the given
setEdges(Double1d edgesPixels, String1d edgesSky) Sets the edges for this PolygonHistogramProduct to the given list
CompositeDataset **getEdges**() Returns the edges for this PolygonHistogramProduct.
int **getNbOfEdges**() Returns the number of edges for this
TableDataset **getEdgesPixelCoordinates**() Returns the edges for this PolygonHistogramProduct in
Double2d **getEdgesPixelCoordinatesDouble2d**() Returns the pixel coordinates of the edges as Double2d.
TableDataset **getEdgesSkyCoordinates**() Returns the edges for this PolygonHistogramProduct in

API details

Constructor

PolygonHistogramProduct()

Methods

setEdges(Double1d edgesPixels)
pixel coordinates. **Argument** Double1d **edgesPixels** [INPUT, MANDATORY]

setEdges(Double1d edgesPixels, String1d edgesSky)

of pixel and sky coordinates.

Arguments

Double1d **edgesPixels** [INPUT, MANDATORY]

String1d **edgesSky** [INPUT, MANDATORY]

CompositeDataset getEdges()

Return

CompositeDataset

Returns the edges for this PolygonHistogramProduct.

int getNbOfEdges()

PolygonHistogramProduct.

Return

int

Returns the number of edges for this PolygonHistogramTask.

TableDataset getEdgesPixelCoordinates()

pixel coordinates.

Return

TableDataset

Returns the edges for this PolygonHistogramProduct in pixel coordinates.

Double2d getEdgesPixelCoordinatesDouble2d()

Return

Double2d

Returns the pixel coordinates of the edges as a Double2d.

TableDataset getEdgesSkyCoordinates()

sky coordinates.

Return

TableDataset

Returns the edges for this Polygon in sky coordinates.

3.241. PolygonHistogramTask

Full Name:	herschel.ia.toolbox.image.PolygonHistogramTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import PolygonHistogramTask

Description

A Task to make a histogram within a polygon.

A Task to make a histogram of a region of interest, which is bounded by a polygon.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`lowCut`** `[INPUT, OPTIONAL, default=No default value]`
`Double` **`highCut`** `[INPUT, OPTIONAL, default=No default value]`
`Integer` **`bins`** `[INPUT, MANDATORY, default=No default values]`
`PolygonHistogramProduct` **`histogram`** `[OUTPUT, MANDATORY, default=No default value]`
`Double1d` **`frequencies`** `[OUTPUT, MANDATORY, default=No default value]`
`Double1d` **`edgesPixel`** `[INPUT, OPTIONAL, default=No default value]`
`String1d` **`edgesSky`** `[INPUT, OPTIONAL, default=No default value]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double lowCut [INPUT, OPTIONAL, default=No default value]`

The lower cut level.

`Double highCut [INPUT, OPTIONAL, default=No default value]`

The upper cut level.

`Integer bins [INPUT, MANDATORY, default=No default values]`

The number of bins.

`PolygonHistogramProduct histogram [OUTPUT, MANDATORY, default=No default value]`

The histogram.

`Double1d frequencies [OUTPUT, MANDATORY, default=No default value]`

The frequencies.

Double1d edgesPixel [INPUT, OPTIONAL, default=No default value]

The edges of the polygon in pixel coordinates.

String1d edgesSky [INPUT, OPTIONAL, default=No default value]

The edges of the polygon in sky coordinates.

3.242. Polynomial

Full Name:	herschel.ia.numeric.toolbox.basic.Polynomial
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Polynomial

Description

Calculates the y points of a polynomial for a given input array.

The input array is an integral or floating-point numeric array (e.g. Double5d) and the coefficients is a double array.

Example

Example 1: Apply Polynomial on a Int1d

```
print Polynomial(Double1d([1]))(Double1d.range(5)) # [1.0,1.0,1.0,1.0,1.0]
print Polynomial(Double1d([0,1]))(Double1d.range(5)) # [0.0,1.0,2.0,3.0,4.0]
print Polynomial(Double1d([0,0,1]))(Double1d.range(5)) # [0.0,1.0,4.0,9.0,16.0]
#or
print Double1d.range(5).apply(Polynomial(Double1d([1]))) #
 [1.0,1.0,1.0,1.0,1.0]
print Double1d.range(5).apply(Polynomial(Double1d([0,1]))) #
 [0.0,1.0,2.0,3.0,4.0]
print Double1d.range(5).apply(Polynomial(Double1d([0,0,1]))) #
 [0.0,1.0,4.0,9.0,16.0]
```

API Summary

Jython Syntax

```
<y>=Polynomial(<coefficients>)(<x>)
```

Properties

```
an integral or floating-point numeric array x [INPUT, MANDATORY,
default=no default value]
```

```
a double array coefficients [INPUT, MANDATORY, default=no default
value]
```

```
array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

an integral or floating-point numeric array x [INPUT, MANDATORY, default=no default value]

The input array is an integral or floating-point numeric array.

a double array coefficients [INPUT, MANDATORY, default=no default value]

The number of element to shift

```
array y [OUTPUT, MANDATORY, default=no default value]
```

Returns an array.

3.243. poolDataReader

Full Name:	herschel.ia.toolbox.util.PoolDataReaderTask
Alias:	poolDataReader
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import PoolDataReaderTask
Category:	utility task

Description

Task for reading a product from a pool.

Data will be readed through a ProductStorage by default. If no storage is specified, a new one will be created (unless the parameter 'useStorage' is false). Only the product identifier is required (an urn or a tag).

Examples

Example 1: Read by urn

```
...
product = PoolDataReader(id='urn:poolid:producClass:N')
print product
...
```

Example 2: Read by tag

```
...
product = PoolDataReader(id='myTag',storage=myStorage)
print product
...
```

Example 3: Read two products using the same storage

```
...
ps = ProductStorage()
pool = PoolManager.getPool("mypoolname")
ps.register(pool)
product1 = PoolDataReader(id='myTag1',storage=ps)
print product1
product2 = PoolDataReader(id='myTag2',storage=ps)
print product2
...
```

Example 4: Read two products using the same default storage

```
...
pdrt = PoolDataReaderTask()
product1 = pdrt(id='myTag1')
print product1
ps = pdrt.storage
product2 = pdrt(id='myTag2',storage=ps)
print product2
...
```

API Summary

Jython Syntax

```
product=PoolDataReaderTask()(urn='urn:poolid')
```

Jython Syntax
`see examples`

Properties
`String` **`id`** `[INPUT, true, default=null]`
`ProductStorage` **`storage`** `[INOUT, false, default=new ProductStorage]`
`ProductPool` **`pool`** `[INPUT, false, default=null (deprecated) PAL default pool]`
`String` **`poolid`** `[INPUT, false, default=null]`
`Boolean` **`useStorage`** `[INPUT, false, default=true]`
`Product` **`product`** `[OUTPUT, true, default=null]`

API details

Properties

`String id [INPUT, true, default=null]`

Product identifier, either an urn or a tag. A Tag can be used when reading through storage only.

`ProductStorage storage [INOUT, false, default=new ProductStorage]`

If storage is specified, pool and poolid parameters are not allowed. If storage is specified and useStorage parameter is false, an exception will be raised. If no storage is specified, pool or poolid parameters will be used and a new storage will be created (unless useStorage is false) and will be available as an output parameter.

`ProductPool pool [INPUT, false, default=null (deprecated) PAL default pool]`

If storage is specified, the pool parameter is not allowed. If a pool is specified, the poolid parameter is not allowed. DEPRECATED: If no pool is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be available as an ouput parameter

`String poolid [INPUT, false, default=null]`

Pool identifier. {@link herschel.ia.pal.PoolManager} will be used to find the pool. If storage is specified, the poolid parameter is not allowed. If a poolid is specified, the pool parameter is not allowed. DEPRECATED: If no poolid is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be available as an ouput parameter.

`Boolean useStorage [INPUT, false, default=true]`

Enable/Disable storage usage. If this parameter is false and storage is specified, an exception will be raised. If this parameter is true and no storage is specified, a new {@link herschel.ia.pal.ProductStorage} will be created and will be available as an output parmeter.

`Product product [OUTPUT, true, default=null]`

The loaded product.

History

- 22-11-2007 JCS

- `2009-02-17 - JDS`: (default pool, SPR-6006)

3.244. poolDataWriter

Full Name:	herschel.ia.toolbox.util.PoolDataWriterTask
Alias:	poolDataWriter
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import PoolDataWriterTask
Category:	utility task

Description

Task for writing a product into a pool.

Data will be saved through a ProductStorage by default. If no storage is specified, a new one will be created and available (unless the parameter 'useStorage' is false).

Examples

Example 1: write product

```
p = Product()
pdwt = PoolDataWriterTask()
urn = pdwt(product=p)
print urn
```

Example 2: write product with a tag and read it later

```
p = Product()
pdwt = PoolDataWriterTask()
urn = pdwt(product=p,tag='myTag')
print urn
...
pdrt = PoolDataReaderTask()
product = pdrt(id='myTag',pool=ProductPool)
print product
```

Example 3: write two products using the same storage

```
ps = ProductStorage()
pool = PoolManager.getPool("standard")
ps.register(pool)
pdwt = PoolDataWriterTask()
urn1 = pdrt(product=Product(),tag='myTag1',storage=ps)
print urn1
urn2 = pdrt(product=Product(),tag='myTag2',storage=ps)
print urn2
```

Example 4: write two products using the same default storage

```
pdwt = PoolDataWriterTask()
urn1 = pdwt(product=Product(),tag='myTag1')
print urn1
ps = pdwt.storage
urn2 = pdwt(product=Product(),tag='myTag2',storage=ps)
print urn2
```

API Summary

Jython Syntax

```
urn=PoolDataWriterTask()(product=Product())
```

Jython Syntax
`See examples.`

Properties
`Product` **`product`** `[INPUT, true, default=null]`
`String` **`tag`** `[INPUT, false, default=null]`
`ProductStorage` **`storage`** `[INOUT, false, default=new ProductStorage]`
`ProductPool` **`pool`** `[INPUT, false, default=PAL default pool]`
`String` **`poolid`** `[INPUT, false, default=null]`
`String` **`urn`** `[OUTPUT, MANDATORY, default=null]`
`Boolean` **`useStorage`** `[INPUT, false, default=true]`

API details

Properties

`Product product [INPUT, true, default=null]`

The product to be saved.

`String tag [INPUT, false, default=null]`

Product tag. A Tag can be used when saving through storage only.

`ProductStorage storage [INOUT, false, default=new ProductStorage]`

If storage is specified, pool and poolid parameters are not allowed. If storage is specified and useStorage parameter is false, an exception will be raised. If no storage is specified, pool or poolid parameters will be used and a new storage will be created (unless useStorage is false) and will be availabe as an output parameter.

`ProductPool pool [INPUT, false, default=PAL default pool]`

If storage is specified, the pool parameter is not allowed. If a pool is specified, the poolid parameter is not allowed. DEPRECATED: If no pool is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be availabe as an ouput parameter

`String poolid [INPUT, false, default=null]`

Pool identifier. {@link herschel.ia.pal.PoolManager} will be used to find the pool. If storage is specified, the poolid parameter is not allowed. If a poolid is specified, the pool parameter is not allowed. DEPRECATED: If no poolid is specified, the default pool will be used. (see {@link herschel.ia.pal}) A product storage will be created (unless useStorage is false) and that storage will be availabe as an ouput parameter.

`String urn [OUTPUT, MANDATORY, default=null]`

The urn of the product just written.

`Boolean useStorage [INPUT, false, default=true]`

Enable/Disable storage usage. If this parameter is false and storage is specified, an exception will be raised. If this parameter is true and no storage is specified, a new {@link herschel.ia.pal.ProductStorage} will be created and will be available as an output parmeter.

History

- 22-11-2007 JCS
- `2008-12-15 - JDS`: (added deprecation marks to jtags)
- `2009-02-17 - JDS`: (default pool, SPR-6006)

3.245. PositionList

Full Name:	herschel.ia.dataset.image.PositionList
Type:	Java Class -
Import:	from herschel.ia.dataset.image import PositionList

Description

Position list.

A list of positions.

API Summary

Methods
`double` **`getRa`**`(int row)` Returns the RA at a given row.
`double` **`getDec`**`(int row)` Returns the Dec at a given row.
`int` **`getSize`**`()` Returns the number of positions.

API details

Methods

`double getRa(int row)`

Returns the RA for the position for this PositionList at the given row in the list.

Argument

`int` **`row`** `[INPUT, MANDATORY]`

Return

double

Returns the RA for the position for this PositionList at the given row in the list.

`double getDec(int row)`

Returns the Dec for the position for this PositionList at the given row in the list.

Argument

`int` **`row`** `[INPUT, MANDATORY]`

Return

double

Returns the Dec for the position for this PositionList at the given row in the list.

`int getSize()`

Returns the number of positions for this PositionList.

`int getSize()`

Return

int

Returns the number of positions for this PositionList.

3.246. PowerSpectrum

Full Name:	herschel.ia.gui.explorer.table.PowerSpectrum
Type:	Java Class -
Import:	from herschel.ia.gui.explorer.table import PowerSpectrum

Description

PowerSpectrum Generator OverView

The PowerSpectrum generator is a GUI tool which allows users to generate PowerSpectrum inactively. The tool allows users to specify the time data and unit. By default, the tool will look for the time data in the table. If there are several time column data, the lowest column index will be used as a time data to calculate frequency.

API Summary

Constructor
`PowerSpectrum(TableDataset table)`

Methods
`setObject(TableDataset data)`
`JComponent getComponent()`

API details

Constructor

`PowerSpectrum(TableDataset table)`

Argument

`TableDataset` **`table`** `[INPUT, MANDATORY, default=no default value]`

The input table must contain one or more time column data.

Methods

`setObject(TableDataset data)`

Argument

`TableDataset` **`data`** `[INPUT, MANDATORY, default=no default value]`

The input table must contain at least one time column data.

`JComponent getComponent()`

Return

JComponent

- this GUI component

History

- `2007-12-02 - first`: version
- `2008-04-03 - Move`: the code to calculate PowerSpectrum to ia_numeric_toolbox_xform
- `2008-11-21 - Major`: GUI change to allow users to choose time data and its unit

3.247. PowerSpectrum

Full Name:	herschel.ia.toolbox.astro.PowerSpectrum
Type:	Java Class -
Import:	from herschel.ia.toolbox.astro import PowerSpectrum

Examples

Example 1: java example import herschel.ia.numeric.toolbox.xform. *;

```
TableDataset table; //defined somewhere double flimit = 0.1; double
sigma = 0.4; boolean deglitch = true;
TableDataset pw_table = PowerSpectrum.getPowerSpectrum(flimit,
sigma, deglitch, table);
```

Example 2: jython example from herschel.ia.numeric.toolbox.xform import *

```
flimt = 0.1 sima = 0.4 deglitch= 1 pw_table =
PowerSpectrum.powerSpectrum(flimit, sigma, deglitch, table)
```

See also

- ???

History

- Feb 22, 2008 - first version

3.248. Pow

Full Name:	herschel.ia.numeric.toolbox.basic.Pow
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Pow

Description

Gives the array raised to the specified power: $y=(x)^{power}$

Example

Example 1: Apply Pow on a Int1d

```
x=Double1d([1,2,3,4])
print Pow(2)(x) # [1.0,4.0,9.0,16.0]
print x.apply(Pow(2)) # [1.0,4.0,9.0,16.0]
```

API Summary

Jython Syntax

```
<y>=Pow(<power>)(<x>)
```

Properties

```
any array or number x [INPUT, MANDATORY, default=no default value]
a number power [INPUT, MANDATORY, default=no default value]
a number or an array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any array or number x [INPUT, MANDATORY, default=no default value]

An array or a number

a number power [INPUT, MANDATORY, default=no default value]

The power

a number or an array y [OUTPUT, MANDATORY, default=no default value]

The result is a number or an array depending on the input

See also

- Pow

3.249. ProcessDistributor

Full Name:	herschel.ia.dataflow.ProcessDistributor
Type:	Java Class -
Import:	from herschel.ia.dataflow import ProcessDistributor

Description

Decouples the chain of execution of processes, i.e, makes a event-based process to behave as thread-based one.

Helper class that allows one or more Event-based processes (or connectables in general) to be run in a different thread.

This is intented to be used as is.

As typical example, a dataflow containing three Event-based processes A, B, C, whose flow is:

```
A----B
|
---C
```

The user (or developer) sees that he needs better performance running B process in another thread, so a PD (product-distributor) process is added to the dataflow this way:

```
A----PD----B
|
---C
```

In case B and C must run in the same thread, the dataflow can be modified like this:

```
A----PD----B
|
---C
```

The ProcessProductDistributor has an input called "input" and a output called "output".

Example

Example 1: how to use a process distributor.

```
<pre>
from herschel.ia.dataflow import *
from myProcesses import *
df = DataFlow("mydf")
df.createProcess("avg", AverageProcess)
df.createProcess("viewer", ViewerProcess)
p = ProcessDistributor("distrib", Double1d)
df.addProcess(p)
df.connect("avg.output", "distrib.input")
df.connect("distrib.output", "viewer.input")
df.start()
</pre>
```

Limitations

no limitation

See also

- reference

History

- 26-5-2005 jcg: change javadoc format for help support.

3.250. PRODUCT

Full Name:	herschel.ia.numeric.toolbox.basic.Product
Alias:	PRODUCT
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Product

Description

Yields the product of all elements in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply PRODUCT on a Int2d

```
x=Int2d( [ [1,2], [-1,3] ])
print PRODUCT(x)    # -6
print PRODUCT(x, 0) # [-1, 6]
print PRODUCT(x, 1) # [ 2,-3]
```

API Summary

Jython Syntax

```
<y>=PRODUCT(<x>, [,<dim>])
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
integer dim [INPUT, MANDATORY, default=no default value]
float or double array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array can be any scalar array.

integer dim [INPUT, MANDATORY, default=no default value]

The dimension to compute the calculation along

float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a scalar of the same type as the type of the input array.

See also

- SUM

3.251. ProductRef

Full Name:	herschel.ia.pal.ProductRef
Type:	Java Class -
Import:	from herschel.ia.pal import ProductRef

Description

A ProductRef provides a reference to a product that is held in a

product storage or to a product in memory. Typically a ProductRef is returned by the load, save and select methods of a ProductStorage. A Product reference is providing a mechanism to inspect the meta data of a stored product without loading the complete product into memory.

Example

Example 1: Usage of a product reference p=Product(creator="me")

```
ref=storage.save(p)
print ref.type # herschel.ia.dataset.Product print ref.urn #
urn:simple.default:herschel.ia.dataset.Product:23674 print
ref.meta['creator'] # me print ref.product.creator # me (product loaded into
memory!)
```

API Summary

Methods
getProduct() Returns the Product class to which this Product reference is
getType() Returns the Product class to which this Product reference is

API details

Methods

getProduct()

pointing to.

getType()

pointing to.

See also

- *Product Access Layer*

3.252. ProductStorage

Full Name:	herschel.ia.pal.ProductStorage
Type:	Java Class -
Import:	from herschel.ia.pal import ProductStorage

Description

The ProductStorage is a storage mechanism to provide read, write and

query on Products stored in registered Product Pools.

Examples

Example 1: creation and registering myStore=ProductStorage()

```
myStore.register(SimplePool.getInstance()) # the simple.default
pool
```

Example 2: loading a Product from this store

```
ref=myStore.load("urn:simple.default:herschel.ia.dataset.Product:123"
)
```

Example 3: Untitled

```
saving a Product product=Product(...) ref=myStorage.save(product)
```

Example 4: select Products based on a query query=AttribQuery(....)

```
results=myStorage.select(query)
```

API Summary

Constructors
ProductStorage()
Creates an empty ProductStorage with no pools registered.
ProductStorage(ProductPool pool)
Creates a ProductStorage with the given pool registered, which
ProductStorage(ProductPool[] pools)
Creates a ProductStorage with the given pools registered.
ProductStorage(String poolName)
Creates a ProductStorage with the the pools corresponding to the
ProductStorage(String[] poolNames)
Creates a ProductStorage with the the pools corresponding to the

Methods
setSharedMode(boolean sharedMode)
Sets whether the last version info is to be updated before each

Methods
`register(ProductPool pool)` Registers a ProductPool to the ProductStorage. The first
`register(ProductPool array pools)` Registers ProductPools to the ProductStorage. The first
`getPools()` Provides the set of ProductPools registered, in order in which
`getWritablePool()` Returns the writable pool, which is the first registered pool.
`Set getProductClasses()` Provides all Product class definitions found in this pool.
`remove(String urn)` Removes a product of given URN from the storage.
`ProductRef load(String id)` Loads a Product from this store.
`ProductRef save(Product product)` Saves a Product to this store. If the product is a context that
`ProductRef saveHeadOnly(Product product)` Saves a Product to this store. If the product is a context that
`Set select(StorageQuery query)` Returns a set of references to products that match the specified
`Set select(StorageQuery query, Set previous)` Returns a set of references to products that match the specified

API details

Constructors

`ProductStorage()`

`ProductStorage(ProductPool pool)`

happens to be the writable one.

Argument

`ProductPool` **`pool`** `[INPUT, MANDATORY]`

Example

Creating a storage initialized with a pool.

```
storage = ProductStorage(pool) # pool is the writable one
```

`ProductStorage(ProductPool[] pools)`

The first provided pool is the writable one.

Argument

`ProductPool[]` **`pools`** `[INPUT, MANDATORY]`

Example

Creating a storage initialized with pools.

ProductStorage(ProductPool[] pools)

```
pool1 = PoolManager.getPool("pool1")
pool2 = PoolManager.getPool("pool2")
pool3 = PoolManager.getPool("pool3")
storage = ProductStorage([pool1, pool2, pool3]) # pool1 is the
writable one
```

ProductStorage(String poolName)

given name. This pool is the writable one.

Argument

`String` **poolName** [INPUT, MANDATORY]

Example

Creating a storage initialized with pools. # the following line

```
storage = ProductStorage("pool") # is equivalent to
pool = PoolManager.getPool("pool")
storage = ProductStorage(pool)
```

ProductStorage(String[] poolNames)

given names registered. The first provided pool is the writable one.

Argument

`String[]` **poolNames** [INPUT, MANDATORY]

Example

Creating a storage initialized with pools. # the following line

```
storage = ProductStorage(["pool1", "pool2", "pool3"]) # is
equivalent to
pool1 = PoolManager.getPool("pool1")
pool2 = PoolManager.getPool("pool2")
pool3 = PoolManager.getPool("pool3")
storage = ProductStorage([pool1, pool2, pool3])
```

Methods

setSharedMode(boolean sharedMode)

operation on the storage. Set it to `true` if you expect any other ProductStorage may update the products from the outside.

This can be set also in the initialization of the application, by the property `hcss.ia.pal.sharedMode`.

Argument

`boolean` **sharedMode** [INPUT, MANDATORY, default=Whether the storage is]

accessed for writing by several processes at the same time.

register(ProductPool pool)

ProductPool will be the one for which save operations would write products to. The remaining pools registered will be read-only.

Argument

`ProductPool` **pool** [INPUT, MANDATORY, default=The product pool that]

will be added to this product storage.

register(ProductPool pool)

Example

How to register a product pool

```
storage.register(pool)
```

register(ProductPool array pools)

ProductPool will be the one for which save operations would write products to. The remaining pools registered will be read-only.

Argument

`ProductPool array` **`pools`** `[INPUT, MANDATORY, default=The product pools]`

that will be added to this product storage.

Example

How to register many pools at the same time

```
storage.register([pool1, pool2, pool3])
```

getPools()

they were initially registered.

getWritablePool()

Set getProductClasses()

Return

Set

a collection of Product classes

remove(String urn)

Argument

`String` **`urn`** `[INPUT, MANDATORY, default=The URN to the Product that]`

should be removed.

ProductRef load(String id)

Argument

`String` **`id`** `[INPUT, MANDATORY, default=The URN or tag to the Product]`

Return

ProductRef

A reference to the Product corresponding to the input URN or tag.

ProductRef save(Product product)

has descendents, all those descendents will be copied if they do not already exist in the destination storage (this is a deep-copy or cloning operation).

Argument

ProductRef save(Product product)

Product **product** [INPUT, MANDATORY, default=The product that should be]

saved.

Return

ProductRef

A reference to the Product saved into the store.

ProductRef saveHeadOnly(Product product)

has descendents, all those descendents will be copied if they do not already exist in the destination storage (this is a deep-copy or cloning operation).

Argument

Product **product** [INPUT, MANDATORY, default=The product that should be]

saved.

Return

ProductRef

A reference to the Product saved into the store.

Set select(StorageQuery query)

query.

Argument

StorageQuery **query** [INPUT, MANDATORY, default=The query applied to]

this store.

Return

Set

A set of references to Products selected by the query

Set select(StorageQuery query, Set previous)

query.

Arguments

StorageQuery **query** [INPUT, MANDATORY, default=The query applied to]

this store.

Set **previous** [INPUT, MANDATORY]

Return

Set

A set of references to Products selected by the query

See also

- *Product Access Layer*

3.253. ProfileExplorer

Full Name:	herschel.ia.gui.image.ProfileExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import ProfileExplorer

Description

An explorer for Profiles.

API Summary

Constructors
ProfileExplorer()
The constructor of a new ProfileExplorer.
ProfileExplorer(Object object)
The constructor of a new ProfileExplorer associated

Methods
String **getName**()
Returns the name for this ProfileExplorer.
String **getDescription**()
Returns the description for this ProfileExplorer.
boolean **canHandle**(Class className)
Checks whether this ProfileExplorer can handle objects of the
setObject(Object object)
Sets the object for this ProfileExplorer to the given object.
Profile **getObject**()
Returns the object for this ProfileExplorer.
Class **getVariableType**()
Returns the expected variable type for this ProfileExplorer.
JComponent **getComponent**()
Returns the component that is responsible for displaying the
JPanel **getCoordinatePanel**()
Constructs and returns the coordinate table for this ProfileExplorer.
JPanel **getProfilePlotPanel**()
Returns the profile plot component for this
addDataObjectListener(DataObjectListener listener)
Add the given listener to this ProfileExplorer to
removeDataObjectListener(DataObjectListener listener)
Removes the given listener for this ProfileExplorer, so

API details

Constructors

ProfileExplorer()

ProfileExplorer(Object object)

with the given object.

Argument

Object **object** [INPUT, MANDATORY]

Methods

String getName()

Return

String

Returns the name for this ProfileExplorer.

String getDescription()

Return

String

Returns the description for this ProfileExplorer.

boolean canHandle(Class className)

given class.

Argument

Class **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this ProfileExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

Object **object** [INPUT, MANDATORY]

Profile getObject()

Return

Profile

Returns the object for this ProfileExplorer.

Class getVariableType()

Return

`Class getVariableType()`

Class

Returns the expected variable type for this ProfileExplorer.

`JComponent getComponent()`

data object for this ProfileExplorer.

Return

JComponent

Returns the component that is responsible for displaying the data object for this ProfileExplorer.

`JPanel getCoordinatePanel()`

Return

JPanel

Returns the coordinate table for this ProfileExplorer.

`JPanel getProfilePlotPanel()`

ProfileExplorer.

Return

JPanel

Returns the profile plot component for this ProfileExplorer.

`addDataObjectListener(DataObjectListener listener)`

receive data object events from it.

Argument

`DataObjectListener` **`listener`** `[INPUT, MANDATORY]`

`removeDataObjectListener(DataObjectListener listener)`

that it no longer receives data object events by this explorer.

Argument

`DataObjectListener` **`listener`** `[INPUT, MANDATORY]`

3.254. Profile

Full Name:	herschel.ia.dataset.image.Profile
Type:	Java Class -
Import:	from herschel.ia.dataset.image import Profile

Description

A class to deal with the results of profile plotting.

API Summary

Constructor
`Profile()` The constructor of a new Profile.

Methods
`setBegin``(double beginX, double beginY)` Sets the begin to the given pixel coordinates.
`setBegin``(double beginX, double beginY, String beginRA, String beginDec)` Sets the begin to the given pixel and sky
`setEnd``(double endX, double endY)` Sets the end to the given pixel coordinates.
`setEnd``(double endX, double endY, String endRA, String endDec)` Sets the end to the given pixel and sky coordinates.
`setProfile``(Double1d profile)` Sets the profile for this Profile to the given
`setUnit``(Unit unit)` Sets the unit for this Profile.
`Double1d` **`getBeginPixelCoordinates`**`()` Returns the begin for this Profile in pixel coordinates.
`String1d` **`getBeginSkyCoordinates`**`()` Returns the begin for this Profile in sky coordinates.
`Double1d` **`getEndPixelCoordinates`**`()` Returns the end for this Profile in pixel coordinates.
`String1d` **`getEndSkyCoordinates`**`()` Returns the end for this Profile in sky coordinates.
`Double1d` **`getProfile`**`()` Returns the profile for this Profile.
`String` **`getUnit`**`()` Returns the unit for this Profile.

API details

Constructor

Profile()

Methods

setBegin(double beginX, double beginY)

Arguments

double **beginX** [INPUT, MANDATORY]

double **beginY** [INPUT, MANDATORY]

setBegin(double beginX, double beginY, String beginRA, String beginDec)

coordinates.

Arguments

double **beginX** [INPUT, MANDATORY]

double **beginY** [INPUT, MANDATORY]

String **beginRA** [INPUT, MANDATORY]

String **beginDec** [INPUT, MANDATORY]

setEnd(double endX, double endY)

Arguments

double **endX** [INPUT, MANDATORY]

double **endY** [INPUT, MANDATORY]

setEnd(double endX, double endY, String endRA, String endDec)

Arguments

double **endX** [INPUT, MANDATORY]

double **endY** [INPUT, MANDATORY]

String **endRA** [INPUT, MANDATORY]

String **endDec** [INPUT, MANDATORY]

setProfile(Double1d profile)

profile.

Argument

Double1d **profile** [INPUT, MANDATORY]

setUnit(Unit unit)

Argument

Unit **unit** [INPUT, MANDATORY]

Double1d getBeginPixelCoordinates()

Return

Double1d getBeginPixelCoordinates()

Double1d

Returns the begin for this Profile in pixel coordinates.

String1d getBeginSkyCoordinates()

Return

String1d

Returns the begin for this Profile in sky coordinates.

Double1d getEndPixelCoordinates()

Return

Double1d

Returns the end for this Profile in pixel coordinates.

String1d getEndSkyCoordinates()

Return

String1d

Returns the end for this Profile in sky coordinates.

Double1d getProfile()

Return

Double1d

Returns the profile for this Profile.

String getUnit()

Return

String

Returns the unit for this Profile.

3.255. ProfilePanel

Full Name:	herschel.ia.toolbox.image.gui.ProfilePanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import ProfilePanel

Description

A panel for the ProfileTask.

API Summary

Constructor
ProfilePanel() The constructor of a new ProfilePanel.

Methods
JPanel **getImagePanel()** Returns a panel that displays the image for this
JPanel **getButtonPanel()** Returns the button panel for this ProfilePanel.
setSiteEventHandler(SiteEventHandler handler) Sets the site event handler for this ProfilePanel to the
setTask(TaskApi task) Associates the given task with this ProfilePanel.
setVariableSelection(VariableSelection selection) Sets the variable selection for this ProfilePanel to the given
Display **getDisplay()** Returns the display for this ProfilePanel.
Image **getImage()** Returns the image associated with this ProfilePanel.
ImageFigure **getLine()** The straight line for this ProfilePanel.
Double **getBegin()** Returns the begin of the straight line associated
Double **getEnd()** Returns the end of the straight line associated
TaskApi **getTask()** Returns the task associated with this
Map **getMap()** Returns the map associated with this ProfilePanel.
SiteEventHandler **getHandler()** Returns the site event handler associated with this
PlotXY **getPlot()**

Methods
Returns the plot associated with this ProfilePanel.
`int getNbOfClicks()` Returns the number of clicks made on the image
`boolean isInImage(MouseEvent me)` Checks whether the given mouse event occurred inside the
`boolean isInImage(double pixX, double pixY)` Checks whether the given pixel coordinates lies

API details

Constructor

`ProfilePanel()`

Methods

`JPanel getImagePanel()`

ProfilePanel.

Return

JPanel

Returns a panel that displays the image for this ProfilePanel.

`JPanel getButtonPanel()`

Return

JPanel

Returns the button panel for this ProfilePanel.

`setSiteEventHandler(SiteEventHandler handler)`

given site event handler.

Argument

`SiteEventHandler` **`handler`** `[INPUT, MANDATORY]`

`setTask(TaskApi task)`

Argument

`TaskApi` **`task`** `[INPUT, MANDATORY]`

`setVariableSelection(VariableSelection selection)`

variable selection.

Argument

`VariableSelection` **`selection`** `[INPUT, MANDATORY]`

`Display getDisplay()`

Return

Display getDisplay()

Display

Returns the display for this ProfilePanel.

Image getImage()

Return

Image

Returns the image associated with this ProfilePanel.

ImageFigure getLine()

Return

ImageFigure

Reurns the straight line for this ProfilePanel.

Double getBegin()

with this ProfilePanel in pixel coordinates.

Return

Double

Returns the begin of the straight line associated with this ProfilePanel in pixel coordinates.

Double getEnd()

with this ProfilePanel in pixel coordinates.

Return

Double

Returns the end of the straight line associated with this ProfilePanel in pixel coordinates.

TaskApi getTask()

ProfilePanel.

Return

TaskApi

Returns the task associated with this ProfilePanel.

Map getMap()

Return

Map

Returns the map associated with this ProfilePanel.

SiteEventHandler getHandler()

ProfilePanel.

Return

SiteEventHandler getHandler()

SiteEventHandler

Returns the site event handler associated with this ProfilePanel.

PlotXY getPlot()

Return

PlotXY

Returns the plot associated with this ProfilePanel.

int getNbOfClicks()

for this ProfilePanel.

Return

int

Returns the number of clicks made on the image for this ProfilePanel.

boolean isInImage(MouseEvent me)

image for this ProfilePanel.

Argument

`MouseEvent me [INPUT, MANDATORY]`

Return

boolean

Returns true if the given mouse event occurred inside the image for this ProfilePanel.

boolean isInImage(double pixX, double pixY)

inside the image for this ProfilePanel.

Arguments

`double pixX [INPUT, MANDATORY]`

`double pixY [INPUT, MANDATORY]`

Return

boolean

Returns true if the given pixel coordinates lie inside the image for this ProfilePanel; false otherwise.

3.256. ProfilePlotting

Full Name:	herschel.ia.toolbox.image.gui.ProfilePlotting
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import ProfilePlotting

Description

An implementation of profile plotting. In a separate window, you can plot the

intensity along a straight line over an image. There is also shown from which sky coordinate to which sky coordinate (if available) and from which pixel coordinate to which pixel coordinate, this straight line is drawn.

API Summary

Constructors
ProfilePlotting(ImageAnalysisToolbox toolbox) The standard constructor for ProfilePlotting. A new window, associated
ProfilePlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox) Constructor for ProfilePlotting. When the new ProfilePlotting is

Methods
setBegin(Double begin) Sets the begin of the straight line associated with this
setEnd(Double end) Sets the end of the straight line, associated with this
PlotXY **getPlot**() Returns the PlotXY shown in this ProfilePlotting.
ImageFigure **getLine**() Returns the straight line along which we wish to plot
Double **getBegin**() Returns the begin of the drawn line in pixel coordinates.
Double **getEnd**() Returns the end of the drawn line in pixel coordinates.
int **getClicks**() Returns the number of valid clicks (i.e. in the image) you made.
String **getSkyCoordinates**() Returns the String version of the sky coordinates
String **getPixelCoordinates**() Returns the String version of the pixel coordinates of the begin.
boolean **isInImage**(double pixX, double pixY) Checks whether the given pixel coordinates lies
Double1d **getPlotPoints**()

Methods
Returns the intensity of the pixels along the straight line,
`ArrayList` **`getImageFigures()`** Returns all ImageFigures used for this ProfilePlotting (the

API details

Constructors

`ProfilePlotting(ImageAnalysisToolbox toolbox)`

with the given ImageAnalysisToolbox is opened. It is on the image, analyzed with the given ImageAnalysisToolbox, we wish to draw a straight line, and plot the intensity along that straight line in the new ProfilePlotting window.

Argument

`ImageAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

`ProfilePlotting(boolean useAsComponent, ImageAnalysisToolbox toolbox)`

component-based, a window for plotting the intensity is opened; otherwise nothing will be shown.

Arguments

`boolean` **`useAsComponent`** `[INPUT, MANDATORY]`

`ImageAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

Methods

`setBegin(Double begin)`

ProfilePlotting to the given point (in mouse coordinates).

Argument

`Double` **`begin`** `[INPUT, MANDATORY]`

`setEnd(Double end)`

ProfilePlotting to the given point (in mouse coordinates).

Argument

`Double` **`end`** `[INPUT, MANDATORY]`

`PlotXY getPlot()`

Return

PlotXY

Returns the PlotXY shown in this ProfilePlotting.

`ImageFigure getLine()`

the intensity in this ProfilePlotting.

Return

ImageFigure

The straight line along which we wish to plot the intensity in this ProfilePlotting.

Double getBegin()

Return

Double

Returns the begin of the drawn line in pixel coordinates.

Double getEnd()

Return

Double

Returns the end of the drawn line in pixel coordinates.

int getClicks()

Return

int

Returns the number of valid clicks (i.e. in the image) you made.

String getSkyCoordinates()

of the begin and end of the drawn straight line.

Return

String

Returns the String version of the sky coordinates of the begin and end of the drawn straight line.

String getPixelCoordinates()

Return

String

Returns the String version of the pixel coordinates of the begin and end of the drawn line.

boolean isInImage(double pixX, double pixY)

inside the image for this ProfilePlotting.

Arguments

double **pixX** [INPUT, MANDATORY]

double **pixY** [INPUT, MANDATORY]

Return

boolean

Returns true if the given pixel coordinates lie inside the image for this ProfilePlotting; false otherwise.

Double1d getPlotPoints()

associated with this ProfilePlotting.

Return

Double1d getPlotPoints()

Double1d

Returns the intensity of the pixels along the drawn straight line associated with this ProfilePlotting.

ArrayList getImageFigures()

straight line on the image).

Return

ArrayList

Returns all ImageFigures used for this ProfilePlotting (the straight line on the image).

3.257. ProfileTask

Full Name:	herschel.ia.toolbox.image.ProfileTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ProfileTask

Description

A Task to make intensity plots along a straight line.

A Task which determines the intensity of the pixels along a straight line on an image.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`beginX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`beginY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`endX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`endY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`beginRA`** `[INPUT, OPTIONAL, default=Default value : "beginRA"]`
`String` **`beginDec`** `[INPUT, OPTIONAL, default=Default value : "beginDec"]`
`String` **`endRA`** `[INPUT, OPTIONAL, default=Default value : "endRA"]`
`String` **`endDec`** `[INPUT, OPTIONAL, default=Default value : "endDec"]`
`Profile` **`profile`** `[OUTPUT, MANDATORY, default=Default value : Profile()]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double beginX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the begin of the straight line.

`Double beginY [INPUT, OPTIONAL, default=Default value : NaN]`

The y-pixel-coordinate of the begin of the straight line.

`Double endX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the end of the straight line.

`Double endY [INPUT, OPTIONAL, default=Default value : NaN]`

The y-pixel-coordinate of the end of the straight line.

String beginRA [INPUT, OPTIONAL, default=Default value : "beginRA"]

The right ascension of the begin of the straight line.

String beginDec [INPUT, OPTIONAL, default=Default value : "beginDec"]

The declination of the begin of the straight line.

String endRA [INPUT, OPTIONAL, default=Default value : "endRA"]

The right ascension of the end of the straight line.

String endDec [INPUT, OPTIONAL, default=Default value : "endDec"]

The declination of the end of the straight line.

Profile profile [OUTPUT, MANDATORY, default=Default value : Profile()]

The profile.

3.258. Query

Full Name:	herschel.ia.pal.query.Query
Type:	Java Class - 
Import:	from herschel.ia.pal.query import Query

Example

Example 1: Example of an query: q=Query("type=='AbcProduct' and

```
creator=='Scott'")
q=Query("foo.Product","type=='AbcProduct' and creator=='Scott'")
```

3.259. RadialVelocity

Full Name:	herschel.ia.toolbox.astro.RadialVelocity
Type:	Java Class -
Import:	from herschel.ia.toolbox.astro import RadialVelocity
Category:	toolbox

Description

The RadialVelocity utility.

Calculates S/C radial velocity projection in the direction of the pointing, including Sun and Earth velocities to help in Doppler correction of observed frequencies.

The corrections applied from the data obtained from products only include: Sun velocity (standard) and Earth velocity (aprox)

Examples

Example 1: Providing the vectors explicitly

```
from herschel.share.fltdyn.math import Vector3
from herschel.share.fltdyn.time import FineTime
from java.util import Date
v = Vector3([1,2,3]) #or Vector3(1,2,3), the S/C velocity
p = Vector3([4,5,6]) #or Vector3(4,5,6), the S/C pointing vector
t = FineTime(Date()) #now, after 1972 UTC
res = RadialVelocity.getProjection(v, p, t) #get LOS velocity (km/s)
print res
```

Example 2: Providing an OrbitEphemerisProduct and a PointingProduct

```
from herschel.share.fltdyn.time import FineTime
vt = OrbitEphemerisProduct() # ...
pt = PointingProduct() # ...
# Sat Jan 01 01:00:00 CET 1972, 1972-01-01T00:00:10.000000 TAI
 (441763210000000)
t = FineTime(Date(63072000000L))
res = RadialVelocity.getProjection(vt, pt, t) #get LOS velocity (km/s)
```

Example 3: Providing only a storage and returning both the pointing and the projection

```
from herschel.ia.toolbox.astro import RadialVelocity
s = ProductStorage() # ...
t = FineTime(Date(63072000000L))
# ra and dec in sexagesimal degrees (J2000), v in km/s
(ra,dec,v)= RadialVelocity.getPointingAndProjection(s, t) #get pointing
 direction and LOS velocity
```

Example 4: Type conversion: Direction to Vector3

```
#Direction to Vector3
from herschel.share.fltdyn.math import Direction
from herschel.share.fltdyn.math import Vector3
d = Direction(0.2, 0.5) #radians
p = Vector3(d)
# ...
d = Direction.fromDegrees(359, 90.01) #sexagesimal degrees
```

Example 5: Type conversion: Double1d (sized 3) to Vector3

```
d = Double1d([1,2,3])
v = Vector3(d.array)
```

Limitations

The corrections applied from the data obtained from products only include: Suns velocity (standard) Earth velocity (aprox)

See also

- RadialVelocity
- ???
- ???
- ???
- ???
- ProductStorage

History

- `2008-10-10 - JDS`: first interface candidate release
- `2008-12-12 - JDS`: added precession algorithm
- `2009-02-02 - JDS`: fixed degrees and radians
- `2009-02-11 - JDS`: added more documentation (units)
- `2009-05-15 - JDS`: moved to using a constant Sun speed vector in J2000, include Earth velocity

3.260. RandomGauss

Full Name:	herschel.ia.numeric.toolbox.random.RandomGauss
Alias:	RandomGauss
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.random import RandomGauss

Description

Generates or populates an array with normally N(0,1) distributed pseudo random numbers

Examples

Example 1: example usage of a random function

```
f=RandomGauss()
f=RandomGauss(seed=72654)
y=f(x)         # creates a random array with same dimensions and type as x
y=x.apply(f)   # like wise, but java style
x.perform(f)   # changes the values in x
```

Example 2: To generate 100 Double values normally distributed with mean=0 and sigma = 1.0

```
f = RandomGauss()
y = Double1d(100).apply(f)
# To make it with mean=mean and sigma=s
ynew = mean + y*s
```

API Summary

Jython Syntax

```
f = RandomGauss([seed=<seed>])
```

Property

```
OPTIONAL seed [INPUT, the seed for the random generator,
default=no default value]
```

API details

Property

OPTIONAL seed [INPUT, the seed for the random generator, default=no default value]

3.261. RandomPoisson

Full Name:	herschel.ia.numeric.toolbox.random.RandomPoisson
Alias:	RandomPoisson
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.random import RandomPoisson

Description

Generates Poisson random numbers sequence with given mean

Examples

Example 1: example usage of a random function

```
f=RandomPoisson(20)
f=RandomPoisson(15,seed=72654)
y=f(x)         # creates a random array with same dimensions and type as x
y=x.apply(f)   # like wise, but java style
x.perform(f)   # changes the values in x
```

Example 2: To generate 100 Double values who follow Poisson distribution with mean 5

```
f = RandomPoisson(5.0)
y = Long1d(100).apply(f)
```

API Summary

Jython Syntax

```
f = RandomPoisson(<mean>,seed=<seed>)
```

Properties

```
MANDATORY mean [INPUT, the Poisson expectation value (mean),
default=no default value]
```

```
OPTIONAL seed [INPUT, the seed for the random generator,
default=no default value]
```

API details

Properties

MANDATORY mean [INPUT, the Poisson expectation value (mean), default=no default value]

OPTIONAL seed [INPUT, the seed for the random generator, default=no default value]

3.262. RandomUniform

Full Name:	herschel.ia.numeric.toolbox.random.RandomUniform
Alias:	RandomUniform
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.random import RandomUniform

Description

Generates arrays with uniformly-distributed random numbers.

Generates or populates an array with uniformly-distributed pseudo-random numbers in the range 0 <= x < max.

Examples

Example 1: example usage of a random function

```
x=Double1d(20)
f=RandomUniform(15,seed=72654)
y=f(x)         # creates a random array with same dimensions and type as x
y=x.apply(f)   # like wise, but java style
x.perform(f)   # changes the values in x
```

Example 2: To generate 100 Double values in [0,1)

```
f = RandomUniform()
y = Double1d(100).apply(f)
# To generate 100 Integer values in [0,100)
fx = RandomUniform(100.0)
yint = Int1d(100).apply(fx)
```

API Summary

Jython Syntax

```
f = RandomUniform([<max>],seed=<seed>) # between [0,max), default max=1.
```

Properties

OPTIONAL **max** [INPUT, if set the random sequence is in [0, default=max) otherwise in [0]

OPTIONAL **seed** [INPUT, the seed for the random generator, default=no default value]

API details

Properties

OPTIONAL max [INPUT, if set the random sequence is in [0, default=max) otherwise in [0]

OPTIONAL seed [INPUT, the seed for the random generator, default=no default value]

3.263. RangeExtractionGui

Full Name:	herschel.ia.gui.cube.RangeExtractionGui
Type:	Java Class -
Import:	from herschel.ia.gui.cube import RangeExtractionGui

Description

A class to extract a sub Range on a cube containing a large spectrum

API Summary

Method
setPlot() initiate the PlotXY, shown in this Range extraction GUI GUI which contain the global spectrum of the GUI.

API details

Method

setPlot()

3.264. RangeExtractionTask

Full Name:	herschel.ia.toolbox.cube.RangeExtractionTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import RangeExtractionTask

Description

RangeExtractionTask

A class to extract a part of the cube given in parameter and to store the result as a new simplespectralcube the extraction can concentrate on the spectral domain or the spatial domain or both

3.265. RealFunction

Full Name:	herschel.ia.numeric.toolbox.RealFunction
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox import RealFunction

Example

Example 1: Untitled

```
class Inverse(RealFunction):<br>
&amp;nbsp;&amp;nbsp;def calc(self, x):<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;return 1 / x<br>
</code>
```

3.266. REBIN

Full Name:	herschel.ia.numeric.toolbox.interp.Rebin
Alias:	REBIN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.interp import Rebin
Category:	numeric

Description

*

The Rebin class resizes a vector or array to dimensions given by the parameters Di. The supplied dimensions must be integral multiples or factors of the original dimension. The expansion or compression of each dimension is independent of the others, so that each dimension can be expanded or compressed by a different value. If the dimensions of the desired result are not integer multiples of the original dimensions, use the REGRID function. The SAMPLE keyword: Normally, REBIN uses bilinear interpolation when magnifying and neighborhood averaging when minifying. Set the SAMPLE keyword to use nearest neighbor sampling for both magnification and minification. If Rebin is initialted without error, the algorithm in IDL is strickly followed. If the Rebin is initiated with error, the weigted mean is used at compression when SAMPLE is not set.

Examples

Example 1: Apply Rebin on Int1d array

```
from herschel.ia.toolbox.basic.interp import Rebin;
Int1d x;
int dimension;
y = x.apply(new Rebin(dimension));
```

Example 2: Apply Rebin on Double2d array

```
from herschel.ia.toolbox.basic.interp import Rebin;
Double2d x, y;
y=x.apply(new Rebin(d1,d2));
```

Example 3: In Jython

```
from herschel.ia.toolbox.basic.interp import Rebin;
x=Int1d.range(12)
y=x.apply(Rebin(24))
or
y=Rebin(24)(x)
```

Example 4: expansion

```
signal=Double1d.range(10)
dim = 20
rebin = Rebin(dim)
rSignal=rebin(signal)
print rSignal
[0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.0]
```

Example 5: calcualte error

```
signal=Double1d.range(20)+1
```

Example 5: calcualte error

```
inError = SQRT(signal)
dim = 10
rebin = Rebin(dim)
rSignal=rebin(signal)
p_error = rebin.getError(inError)
print p_error
[0.8660254037844387,1.3228756555322954,1.6583123951777,1.9364916731037087,2.179449471770337,
2.3979157616563596,2.598076211353316,2.7838821814150108,2.958039891549808,3.1224989991991996]
```

Example 6: Use SAMPLE keyword

```
signal=Double1d.range(10)
dim = 20
rebin = Rebin(dim, Rebin.SAMPLE)
rSignal=rebin(signal)
print rSignal
[0.0,0.0,1.0,1.0,2.0,2.0,3.0,3.0,4.0,4.0,5.0,5.0,6.0,6.0,7.0,7.0,8.0,8.0,9.0,9.0]
```

Example 7: Use wighted error

```
signal=Double1d.range(20)+1
inError = SQRT(signal)
dim = 10
rebin = Rebin(dim, inError)
rSignal=rebin(signal)
p_error = rebin.getError()
print p_error
[0.6666666666666666,1.7142857142857142,2.7272727272727275,3.7333333333333343,4.736842105263158,
5.739130434782608,6.7407407407407405,7.741935483870969,8.742857142857142,9.743589743589745]
```

API Summary

Jython Syntax

<y>=Rebin(<d1[,d2,d3,d4,d5]>)(<x>)

or

<y>=Rebin(<int d[]>)(<x>)

where <x>=input

<di> = new dimensions of output array

d[] = an array containing the new dimensions

<y>=output resampled onto dimensions.

Properties

type **di INPUT** [the dimension of the corresponding array index, MANDATORY, default=no default value]

array of integer **x** [INPUT, short, default=long or double taken the form of Int1(2]

type **sample Normally** [REBIN uses bilinear interpolation when magnifying and neighborhood averaging, MANDATORY, default=no default value]

Limitations

The supplied dimensions must be integral multiples or factors of the original dimension. Let f = n/m, the ratio of the size of the original vector, X to the size of the result. 1/f must be an integer if n < m (expansion). f must be an integer if compressing, (n > m).

API details

Properties

type di INPUT [the dimension of the corresponding array index, MANDATORY, default=no default value]

array of integer x [INPUT, short, default=long or double taken the form of Int1(2]

type sample Normally [REBIN uses bilinear interpolation when magnifying and neighborhood averaging, MANDATORY, default=no default value]

when minifying. Set the SAMPLE to Rebin.SAMPLE to use nearest neighbor sampling for both magnification and minification. Bilinear interpolation gives higher quality results but requires more time.

History

- `2006-08-30 - first`: version
- `2009-03-23 - error`: propagation (SCR-4833) is added.

3.267. RectangleHistogramExplorer

Full Name:	herschel.ia.gui.image.RectangleHistogramExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import RectangleHistogramExplorer

Description

An explorer for RectangleHistogramProducts.

API Summary

Constructors
`RectangleHistogramExplorer()` The constructor of a new RectangleHistogramExplorer.
`RectangleHistogramExplorer(Object object)` The constructor of a new RectangleHistogramExplorer associated

Methods
`String` **`getName()`** Returns the name for this RectangleHistogramExplorer.
`String` **`getDescription()`** Returns the description for this RectangleHistogramExplorer.
`boolean` **`canHandle(Class className)`** Checks whether this RectangleHistogramExplorer can handle objects of the
`setObject(Object object)` Sets the object for this RectangleHistogramExplorer to the given object.
`RectangleHistogramProduct` **`getObject()`** Returns the object for this RectangleHistogramExplorer.
`Class` **`getVariableType()`** Returns the expected variable type for this RectangleHistogramExplorer.
`JTable` **`getParameterTable()`** Returns the parameter table for this

API details

Constructors

`RectangleHistogramExplorer()`

`RectangleHistogramExplorer(Object object)`

with the given object.

Argument

`Object` **`object`** `[INPUT, MANDATORY]`

Methods

String getName()

Return

String

Returns the name for this RectangleHistogramExplorer.

String getDescription()

Return

String

Returns the description for this RectangleHistogramExplorer.

boolean canHandle(Class className)

given class.

Argument

Class **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this RectangleHistogramExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

Object **object** [INPUT, MANDATORY]

RectangleHistogramProduct getObject()

Return

RectangleHistogramProduct

Returns the object for this RectangleHistogramExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this RectangleHistogramExplorer.

JTable getParameterTable()

RectangleHistogramExplorer.

Return

JTable

Returns the parameter table for this RectangleHistogramExplorer.

3.268. RectangleHistogramPanel

Full Name:	herschel.ia.toolbox.image.gui.RectangleHistogramPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import RectangleHistogramPanel

Description

A panel for the RectangleHistogramPanel.

API Summary

Constructor
RectangleHistogramPanel() The construction of a new RectangleHistogramPanel.

Methods
drawFigure() Draws the rectangle on the image associated with this
updateFigure() Updates the rectangle associated with this
trigger() Triggers the execution of the task associated
updateHistogram() Updates the histogram associated with this

API details

Constructor

RectangleHistogramPanel()

Methods

drawFigure()
RectangleHistogramPanel.

updateFigure()
RectangleHistogramPanel.

trigger()
with this RectangleHistogramPanel.

updateHistogram()
RectangleHistogramPanel.

3.269. RectangleHistogramProduct

Full Name:	herschel.ia.dataset.image.RectangleHistogramProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import RectangleHistogramProduct

Description

A class to deal with the results of a rectangle histogram.

API Summary

Constructor
`RectangleHistogramProduct()` The constructor of a new RectangleHistogramProduct.

Methods
`setUpperLeftCorner``(double upperLeftX, double upperLeftY)` Sets the upper left corner for this RectangleHistogramProduct
`setUpperLeftCorner``(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLefttDec)` Sets the upper left corner for this RectangleHistogramProduct
`setDimensions``(double widthPixels, double heightPixels)` Sets the dimensions for this RectangleHistogramProduct to the
`setDimensions``(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)` Sets the dimensions for this RectangleHistogramProduct to the
`Double1d` **`getUpperLeftCornerPixelCoordinates`**`()` Returns the upper left corner for this RectangleHistogramProduct in
`String1d` **`getUpperLeftCornerSkyCoordinates`**`()` Returns the upper left corner for this RectangleHistogramProduct in
`double` **`getWidthPixels`**`()` Returns the width for this RectangleHistogramProduct in pixels.
`double` **`getWidthArcsec`**`()` Returns the width for this RectangleHistogramProduct in arcsec.
`double` **`getHeightPixels`**`()` Returns the height for this RectangleHistogramProduct in pixels.
`double` **`getHeightArcsec`**`()` Returns the height for this RectangleHistogramProduct in arcsec.

API details

Constructor

`RectangleHistogramProduct()`

Methods

setUpperLeftCorner(double upperLeftX, double upperLeftY)

to the given pixel coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

setUpperLeftCorner(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLefttDec)

to the given pixel and sky coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

String **upperLeftRA** [INPUT, MANDATORY]

String **upperLefttDec** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels)

given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)

given dimensions in pixels and arcsec.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

double **widthArcsec** [INPUT, MANDATORY]

double **heightArcsec** [INPUT, MANDATORY]

Double1d getUpperLeftCornerPixelCoordinates()

pixel coordinates.

Return

Double1d

Returns the upper left corner for this RectangleHistogramProduct in pixel coordinates.

String1d getUpperLeftCornerSkyCoordinates()

sky coordinates.

Return

String1d

Returns the upper left corner for this RectangleHistogramProduct in sky coordinates.

`double getWidthPixels()`

Return

double

Returns the width for this RectangleHistogramProduct in pixels.

`double getWidthArcsec()`

Return

double

Returns the width for this RectangleHistogramProduct in arcsec.

`double getHeightPixels()`

Return

double

Returns the height for this RectangleHistogramProduct in pixels.

`double getHeightArcsec()`

Return

double

Returns the height for this RectangleHistogramProduct in arcsec.

3.270. RectangleHistogramTask

Full Name:	herschel.ia.toolbox.image.RectangleHistogramTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import RectangleHistogramTask

Description

A Task to make a histogram within a rectangle.

A Task to make a histogram of a region of interest, which is bounded by a rectangle.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`lowCut`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`highCut`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Integer` **`bins`** `[INPUT, MANDATORY, default=Default value : new Integer(2000)]`
`RectangleHistogramProduct` **`histogram`** `[OUTPUT, MANDATORY, default=No default value]`
`Double` **`minX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`minY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`minRA`** `[INPUT, OPTIONAL, default=Default value : "minRA"]`
`String` **`minDec`** `[INPUT, OPTIONAL, default=Default value: "minDec"]`
`Double` **`widthPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`heightPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`widthArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`heightArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double lowCut [INPUT, OPTIONAL, default=Default value : NaN]`

The lower cut level.

`Double highCut [INPUT, OPTIONAL, default=Default value : NaN]`

The upper cut level.

`Integer bins [INPUT, MANDATORY, default=Default value : new Integer(2000)]`

The number of bins.

RectangleHistogramProduct histogram [OUTPUT, MANDATORY, default=No default value]

The histogram.

Double minX [INPUT, OPTIONAL, default=Default value : NaN]

The x-pixel-coordinate of the upper left corner of the rectangle.

Double minY [INPUT, OPTIONAL, default=Default value : NaN]

The y-pixel-coordinate of the upper left corner of the rectangle.

String minRA [INPUT, OPTIONAL, default=Default value : "minRA"]

The right ascension of the upper left corner of the rectangle.

String minDec [INPUT, OPTIONAL, default=Default value: "minDec"]

The declination of the upper left corner of the rectangle.

Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]

The width of the rectangle in pixels.

Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]

The height of the rectangle in pixels.

Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The width of the rectangle in arcsec.

Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The height of the rectangle in arcsec.

3.271. RectangularSkyAperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.RectangularSkyAperturePhotometryExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import RectangularSkyAperturePhotometryExplorer

Description

An explorer for RectangularSkyAperturePhotometryProducts.

API Summary

Constructors
`RectangularSkyAperturePhotometryExplorer()` The constructor of a new RectangularSkyAperturePhotometryExplorer.
`RectangularSkyAperturePhotometryExplorer(Object object)` The constructor of a new RectangularSkyAperturePhotometryExplorer

Methods
`boolean` **`canHandle`**`(Class className)` Checks whether this RectangularSkyAperturePhotometryExplorer can
`RectangularSkyAperturePhotometryProduct` **`getObject`**`()` Returns the object for this RectangularSkyAperturePhotometryExplorer.
`Class` **`getVariableType`**`()` Returns the expected variable type for this RectangularAperturePhotometryExplorer.
`JTable` **`getParameterTable`**`()` Constructs and returns the parameter table for this
`JPanel` **`getPlotPanel`**`()` Constructs and returns the plot panel for this

API details

Constructors

`RectangularSkyAperturePhotometryExplorer()`

`RectangularSkyAperturePhotometryExplorer(Object object)`

associated with the given object.

Argument

`Object` **`object`** `[INPUT, MANDATORY]`

Methods

`boolean canHandle(Class className)`

handle objects of the given class.

boolean canHandle(Class className)

Argument

`Class` **className** [INPUT, MANDATORY]

Return

boolean

Returns true if this RectangularSkyAperturePhotometryExplorer can handle objects of the given class; false otherwise.

RectangularSkyAperturePhotometryProduct getObject()

Return

RectangularSkyAperturePhotometryProduct

Returns the object for this RectangularSkyAperturePhotometryExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this RectangularAperturePhotometryExplorer.

JTable getParameterTable()

RectangularSkyAperturePhotometryExplorer.

Return

JTable

Returns the parameter table for this RectangularSkyAperturePhotometryExplorer.

JPanel getPlotPanel()

RectangularSkyAperturePhotometryExplorer.

Return

JPanel

Returns the plot panel for this RectangularSkyAperturePhotometryExplorer.

3.272. RectangularSkyAperturePhotometryPanel

Full Name:	herschel.ia.toolbox.image.gui.RectangularSkyAperturePhotometryPanel
Type:	Java Class -
Import:	from herschel.ia.toolbox.image.gui import RectangularSkyAperturePhotometryPanel

Description

A panel for the RectangularSkyAperturePhotometryTask.

API Summary

Constructor
RectangularSkyAperturePhotometryPanel() The constructor of a new RectangularSkyAperturePhotometryPanel.

Methods
JPanel **getAperturesPanel()** Returns the panel where to specify the target radius and
JComponent **getSkyApertureComponent()** Returns the panel where it is explained how to
drawRectangle() Allows to draw a rectangle on the image associated
int **getNbOfRectanglePoints()** Returns the number of points that were confirmed
increaseNbOfRectanglePoints() Increases the number of rectangle points that were
drawFigures() Draws the circles on the image for this
moveConfirmedFigures() Allows the user to drag the figures bounding the
clearSkyAperture() Clears the rectangular sky aperture for this

API details

Constructor

RectangularSkyAperturePhotometryPanel()

Methods

JPanel getAperturesPanel()
where it is explained how to specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPane.

JPanel getAperturesPanel()

Return

JPanel

Returns the panel where to specify the target radius and where it is explained to specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPanel.

JComponent getSkyApertureComponent()

specify the rectangular sky aperture for this RectangularSkyAperturePhotometryPanel.

Return

JComponent

Returns the panel where it is explained how to specify the rectangular sky aoertyre for this RectangularSkyAperturePhotometryPanel.

drawRectangle()

with this RectangularSkyAperturePhotometryPanel.

int getNbOfRectanglePoints()

before for the rectangle on the image for this RectangularSkyAperturePhotometryPanel.

Return

int

Returns the number of points that were confirmed before for the rectangle on the image for this RectangularSkyAperturePhotometryPanel.

increaseNbOfRectanglePoints()

confirmed before on the image for this RectangularSkyAperturePhotometryPanel.

drawFigures()

RectangularSkyAperturePhotometryPanel.

moveConfirmedFigures()

apertures for this RectangularSkyAperturePhotometryPanel.

clearSkyAperture()

RectangularSkyAperturePhotometryPanel.

3.273. RectangularSkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.RectangularSkyAperturePhotometryProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import RectangularSkyAperturePhotometryProduct

Description

A class to deal with the resuls of aperture photometry with a

rectangular sky aperture.

API Summary

Constructor
`RectangularSkyAperturePhotometryProduct()` The constructor of a new RectangularSkyAperturePhotometryProduct.

Methods
`setUpperLeftCorner``(double upperLeftX, double upperLeftY)` Sets the upper left corner for this RectangularSkyAperturePhotometryProduct
`setUpperLeftCorner``(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLeftDec)` Sets the upper left corner for this RectangularSkyAperturePhotometryProduct.
`setDimensions``(double widthPixels, double heightPixels)` Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the
`setDimensions``(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)` Sets the dimensions for this RectangularSkyAperturePhotometryProduct to the
`Double1d` **`getUpperLeftCornerPixelCoordinates`**`()` Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in
`String1d` **`getUpperLeftCornerSkyCoordinates`**`()` Returns the upper left corner for this RectangularSkyAperturePhotometryProduct
`double` **`getWidthPixels`**`()` Returns the width for this RectangularSkyAperturePhotometryProduct
`double` **`getWidthArcsec`**`()` Returns the width for this RectangularSkyAperturePhotometryProduct
`double` **`getHeightPixels`**`()` Returns the height for this RectangularSkyAperturePhotometryProduct
`getHeightArcsec``()` Returns the height for this RectangularSkyAperturePhotometryProduct

Miscellaneous

Aperture specific information iss stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

RectangularSkyAperturePhotometryProduct()

Methods

setUpperLeftCorner(double upperLeftX, double upperLeftY)

to the given pixel coordinates.

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

setUpperLeftCorner(double upperLeftX, double upperLeftY, String upperLeftRA, String upperLeftDec)

Arguments

double **upperLeftX** [INPUT, MANDATORY]

double **upperLeftY** [INPUT, MANDATORY]

String **upperLeftRA** [INPUT, MANDATORY]

String **upperLeftDec** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels)

given dimensions in pixels.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

setDimensions(double widthPixels, double heightPixels, double widthArcsec, double heightArcsec)

given dimensions in pixels and arcsec.

Arguments

double **widthPixels** [INPUT, MANDATORY]

double **heightPixels** [INPUT, MANDATORY]

double **widthArcsec** [INPUT, MANDATORY]

double **heightArcsec** [INPUT, MANDATORY]

Double1d getUpperLeftCornerPixelCoordinates()

pixel coordinates.

Return

Double1d

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in pixel coordinates.

String1d getUpperLeftCornerSkyCoordinates()

in sky coordinates.

String1d getUpperLeftCornerSkyCoordinates()

Return

String1d

Returns the upper left corner for this RectangularSkyAperturePhotometryProduct in sky coordinates.

double getWidthPixels()

in pixels.

Return

double

Returns the width for this RectangularSkyAperturePhotometryProduct in pixels.

double getWidthArcsec()

in arcsec.

Return

double

Returns the width for this RectangularSkyAperturePhotometryProduct in arcsec.

double getHeightPixels()

in pixels.

Return

double

Returns the height for this RectangularSkyAperturePhotometryProduct in pixels.

getHeightArcsec()

in arcsec.

3.274. RectangularSkyAperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.RectangularSkyAperturePhotometryTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import RectangularSkyAperturePhotometryTask

Description

A Task for aperture photometry.

A Task for aperture photometry, using a circular target aperture and a rectangular sky aperture.

API Summary

Properties
`Image` **`image`** `[INPUT, MANDATORY, default=No default value]`
`Double` **`centerX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`centerY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`centerRA`** `[INPUT, OPTIONAL, default=Default value : "centerRA"]`
`String` **`centerDEC`** `[INPUT, OPTIONAL, default=Default value : "centerDec"]`
`Double` **`radiusPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`radiusArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Integer` **`algorithm`** `[INPUT, MANDATORY, default=Default value : 4]`
`RectangularSkyAperturePhotometryProduct` **`result`** `[OUTPUT, MANDATORY, default=No default value]`
`Double` **`minX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`minY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`minRA`** `[INPUT, OPTIONAL, default=Default value : "upperLeftRA"]`
`String` **`minDec`** `[INPUT, OPTIONAL, default=Default value : "upperLeftDec"]`
`Double` **`widthPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`heightPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`widthArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`heightArcsec`** `[INPUT, OPTIONAL, default=Default value : NaN]`

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image.

`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the target center.

Double centerY [INPUT, OPTIONAL, default=Default value : NaN]

The y-pixel-coordinate of the target center.

String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]

The right ascension of the target center.

String centerDEC [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the target center.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The target radius in pixels.

Double radiusArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The target radius in arcsec.

Integer algorithm [INPUT, MANDATORY, default=Default value : 4]

The sky estimation algorithm.

RectangularSkyAperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

The result.

Double minX [INPUT, OPTIONAL, default=Default value : NaN]

The input x-pixel-coordinate of the upper left corner of the rectangular sky aperture.

Double minY [INPUT, OPTIONAL, default=Default value : NaN]

The input y-pixel-coordinate of the upper left corner of the rectangular sky aperture.

String minRA [INPUT, OPTIONAL, default=Default value : "upperLeftRA"]

The input right ascension of the upper left corner of the rectangular sky aperture.

String minDec [INPUT, OPTIONAL, default=Default value : "upperLeftDec"]

The input declination of the upper left corner of the rectangular sky aperture.

Double widthPixels [INPUT, OPTIONAL, default=Default value : NaN]

The input width of the rectangular sky aperture in pixels.

Double heightPixels [INPUT, OPTIONAL, default=Default value : NaN]

The input height of the rectangular sky aperture in pixels.

Double widthArcsec [INPUT, OPTIONAL, default=Default value : NaN]

The input width of the rectangular sky aperture in arcsec.

Double heightArcsec [INPUT, OPTIONAL, default=Default value : NaN]
The input height of the rectangular sky aperture in arcsec.

3.275. ReplaceFreqRanges

Full Name:	herschel.ia.toolbox.spectrum.ReplaceFreqRanges
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import ReplaceFreqRanges

Description

Task for inserting and/or replacing ranges within the segments of a container.

Various different modes are available for how the replacement / insertion should be processed. For further details see the mode-parameter below.

Example

Example 1: from jide

```
from herschel.ia.toolbox.spectrum import ReplaceFreqRanges
replace = ReplaceFreqRanges()
replacedSlices1 = replace(ds=spectra, by=slices, mode="replace")
replacedSlices2 = replace(ds=spectra, by=slices, mode="avg")
replacedSlices3 = replace(ds=spectra, by=slices, mode="insert")
```

API Summary

Properties
SpectrumContainer **ds** [INPUT, OPTIONAL, default=No default value.]
SpectrumContainer **by** [INPUT, OPTIONAL, default=No default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=No default value.]
String **mode** [INPUT, OPTIONAL, default=replace.]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=No default value.]

The container in which the spectra of the other input container should be inserted.

SpectrumContainer by [INPUT, OPTIONAL, default=No default value.]

The container with the slices that should be inserted into the first input container.

SpectrumContainer result [OUTPUT, OPTIONAL, default=No default value.]

The output container.

String mode [INPUT, OPTIONAL, default=replace.]

A mode specifying how the module should actually do the replacement. Three options are available:

String mode [INPUT, OPTIONAL, default=replace.]

- "replace": Resample the slices to be inserted to the frequency grid of the original container (ds) and replace the original ranges by the resampled slices.
- "avg": Resample the slices to be inserted to the frequency grid of the original container (ds) and replace the original ranges by the average of the original ranges and the resampled slices.
- "insert": Insert brute-force the slices by replacing the ranges of the original spectra by matching the frequency ranges - irrespective of the frequency grid.

History

- `2008-03-27 - meli`: initial.

3.276. ResampleFrequency

Full Name:	herschel.ia.toolbox.spectrum.ResampleFrequency
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import ResampleFrequency

Description

Task for resampling the flux values included in a spectrum container with respect

to a modified frequency grid. Different schemes are available - all should conserve total flux.

The standard (default) scheme uses a trapezoidal integration scheme in combination with linear interpolation. An alternative scheme consists of using an euler integration scheme in combination with nearest neighbor interpolation.

Flags and weights are processed if available:

- For the weights, the same integration/interpolation scheme is applied as is used for the flux. For the weights, we always assume that the weights are defined 'per channel'. (For the flux, we generally assume that it is given on a 'per frequency' basis - if not otherwise stated by the 'isDensity' quantity.
- The flag for an output channel is defined by combining the flags of all the input channels that are considered (for the given output channel) by an bitwise OR logic. This allows to recover all the distinct flag values that are involved in the computation of the given channel.

For output channels that are not supported by any input channels, we set for flux and weights values 'NaN' and as flag 'NotSampled' (64).

Other attributes found in the spectra are copied to the result container without any change.

Example

Example 1: from Jide:

```
resampled1 = resample(ds=spectra, density=True, resolution=1.0)
resampled2 = resample(ds=spectra, density=True, scheme="euler", resolution=1.0)
grid = [4000.0 + 2.0*Double1d.range(500), 5000.0 + 5.0*Double1d.range(200),
 6000.0 + 1.0*Double1d.range(1000), 7000.0 + 2.0*Double1d.range(500)]
resampled3 = resample(ds=spectra, density=True, grid=grid)
```

API Summary

Properties
`SpectrumContainer` **`ds`** `[INPUT, MANDATORY, default=no default value.]`
`SpectrumContainer` **`result`** `[OUTPUT, MANDATORY, default=no default value.]`
`String` **`scheme`** `[INPUT, OPTIONAL, default=no default value.]`
`Double1d[]|Double1d` **`grid`** `[INPUT, OPTIONAL, default=no default value.]`
`double` **`resolution`** `[INPUT, OPTIONAL, default=no default value.]`
`Boolean` **`density`** `[INPUT, OPTIONAL, default=no default value.]`

API details

Properties

SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]

The input container to be resampled.

SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]

The output container with the re-sampled spectra.

String scheme [INPUT, OPTIONAL, default=no default value.]

The scheme to be adopted for the resampling - available are "standard" (or equivalently "trapezoidal") and "simple" (or equivalently "euler").

- The "standard" scheme combines a linear interpolation with a trapezoidal integration scheme.
- The "simple" scheme combines a nearest neighbor interpolation and an Euler integration scheme.

Both scheme are constructed such that total flux is conserved.

Double1d[]|Double1d grid [INPUT, OPTIONAL, default=no default value.]

The new frequency grid the spectra should be resampled to. It is specified as a Double1d for each of the sub-segments. The grid is specified in the same units as the original frequency data. The grid may be non-equidistant.

double resolution [INPUT, OPTIONAL, default=no default value.]

In case no output frequency grid is specified, the task creates an output grid with this given resolution (width) by determining, for each sub-segment, the smallest minimum frequency and the largest maximum frequency. The width is specified in the same units as the original frequency grid. In case the width is specified as a negative number the resulting grid will be in decreasing order.

Boolean density [INPUT, OPTIONAL, default=no default value.]

If set to true the flux data is treated as a flux density (per frequency unit) - otherwise the flux is treated as a per channel quantity.

History

- `2008-01-29 - meli`: initial.
- `2008-03-16 - meli`: major refactoring of the toolbox.
- `2008-09-30 - meli`: trapezoidal scheme used as standard (default); some refactoring (interpolation scheme implicit in integration scheme.
- `2008-10-03 - meli`: flag, weights processing, javadoc, ...

3.277. RESHAPE

Full Name:	herschel.ia.numeric.toolbox.basic.Reshape
Alias:	RESHAPE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Reshape

Description

Creates a new array of the same type as the input array but with a different shape.

The input array can be an array of any rank, and the output array has a rank defined by the specified shape argument.

The shape argument must be an array of integers such as [3,4], and the resulting array must have the same number of elements as the input array. If the shape argument is not specified, the result will always be a 1d-array.

Example

Example 1: Apply RESHAPE on a Int1d

```
x=Int1d.range(12)
print RESHAPE(x,[2,6])   # [[0,1,2,3,4,5],[6,7,8,9,10,11]]
print RESHAPE(x,[6,2])   # [[0,1],[2,3],[4,5],[6,7],[8,9],[10,11]]
print RESHAPE(x,[2,2,3]) # [[[0,1,2],[3,4,5]],[[6,7,8],[9,10,11]]]
print RESHAPE(TRANSPOSE(RESHAPE(x,[4,3])))
# [0,3,6,9,1,4,7,10,2,5,8,11]
```

API Summary

Jython Syntax

```
<y>=RESHAPE(<x>, [<shape>])
```

Properties

```
any array of integral type x [INPUT, MANDATORY, default=no default value]
array of integers shape [INPUT, NOT_MANDATORY, default=1d]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any array of integral type x [INPUT, MANDATORY, default=no default value]

The input array can be an array of any rank

array of integers shape [INPUT, NOT_MANDATORY, default=1d]

The shape argument must be an array of integers such as [3,4]. If the shape argument is not specified, the result will always be a 1d-array.

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
Returns an array with the same number of elements as the input array and with shape equal to is specified or 1d if not specified.

See also

- CONCATENATE

3.278. restore

Full Name:	herschel.ia.toolbox.util.RestoreTask
Alias:	restore
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import RestoreTask
Category:	task

Description

Restore variables from a file to a jython session

The restore task is used to restore one or more variables from a file to a jython session. The restore task is used to save one or more variable from a previously saved file into the specified namespace.

Please note that when used for restoring values using the local() option the result might be unpredictable if not executed as first command. Please note that only Serializable variable can be saved.

Examples

Example 1: restore variables (names and values) from a file

```
restore("foo.sav")
```

Example 2: restore variables (names and values) from a file into a function namespace

```
def my_function():
 restore("test.sav", space=locals())
 print locals()
```

API Summary

Jython Syntax

```
restore(<filename>[, <space>])
```

Properties

```
String filename [INPUT, MANDATORY, default=No default value]
PyStringMap space [INPUT, OPTIONAL, default=globals()]
```

Limitations

Only Serializable variable saved can be restored, when using for restoring variables using locals result might be unpredictable if not executed as first command.

Miscellaneous

No miscellaneous

API details

Properties

String filename [INPUT, MANDATORY, default=No default value]

The name of the file where the variables along with their values are stored

PyStringMap space [INPUT, OPTIONAL, default=globals()]

A jython dictionary (PyStringMap) containing the namespace to load the variables into, The choices available are:

1. "locals()" for reading variables from the local namespace (i.e. from where the function is used)
2. "globals()" for reading variables from the module where the function is used

When no space is specified the top level namespace is updated.

See also

- save

History

- `2004-07-13 - NdC`: first release.
- `2009-01-21 - JDS`: Cleanup

3.279. resume

Full Name:	herschel.ia.toolbox.util.ResumeTask
Alias:	resume
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import ResumeTask
Category:	task

Description

Pause the execution of jython code

The resume task is used to resume the execution of jython after a call of pause. The resume can only be performed interactively from the debbuger window and not programmatically as jython doesn't execute code just after a pause call.

Example

Example 1: resume the jython execution (only available from the debug command window)

```
resume()
```

Miscellaneous

No miscellaneous

See also

- resume

History

- `2007-10-11 - NdC`: first release.

3.280. REVERSE

Full Name:	herschel.ia.numeric.toolbox.basic.Reverse
Alias:	REVERSE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Reverse

Description

Reverses the order of the elements in an array of rank 1

Example

Example 1: Apply REVERSE on a Double1d

```
print REVERSE(Double1d.range(3)) # [2.0,1.0,0.0]
```

API Summary

Jython Syntax
<y>=REVERSE(<x>)

Properties
any array of rank 1 **x** [INPUT, MANDATORY, default=no default value]
boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array of rank 1 x [INPUT, MANDATORY, default=no default value]

The input array can be an array of rank 1

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array with the same number of elements as the input array and with the reverse order.

3.281. Rotate

Full Name:	herschel.ia.numeric.toolbox.basic.Rotate
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Rotate

Description

Rotates numeric 2d and 3d arrays.

The result is an array of the double type. The returned array is usually larger than the original because a rotation of a rectangle array creates empty corners. The corners are filled with 0 by default. (This does not happen with rotations around multiples of 90 degrees).

The rotation is always around the center of the array. That means the result contains the complete rotated array without any cuts.

Possible interpolation is nearest neighbor, bilinear und bicubic.

The interpolation quality depends on the rotation angle and on the interpolation method. It can be measured as a sum over all array indexes. An example is presented with an 200X255 Int2d array. It is constructed as follows:

```
array = Int2d();
for i in range(200):
array.append(Int1d.range(255),1)
```

The sum is 6477000

With a rotation angle of 54 deg clockwise the sums are:

Nearest neighbor: 6477268 (0.004% accurate)

Bilinear: 6447345 (0.46% accurate)

Bicubic: 6451030 (0.40% accurate)

Example

Example 1: apply Rotate to a Int2d array

```
Clockwise nearest neighbour rotation (default)
 result = Rotate(54.0)( array )
Counterclockwise nearest neighbour rotation
 result = Rotate(-54.0)( array )
Clockwise bicubic rotation
 result = Rotate(54.0, Rotate.BICUBIC) ( array ) or
 result = Rotate(54.0, 3) ( array )
Counterclockwise bilinear rotation
 result = Rotate(54.0, Rotate.BILINEAR, False) ( array )
 array = Int2d();
 for i in range(5):
     array.append(Int1d.range(5),1)
print array
[
[0,0,0,0,0],
[1,1,1,1,1],
[2,2,2,2,2],
[3,3,3,3,3],
[4,4,4,4,4]
```

Example 1: apply Rotate to a Int2d array

```
]
print Rotate(90) ( array )
[
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0],
[4.0,3.0,2.0,1.0,0.0]
]
```

API Summary

Jython Syntax

```
<y> = <x>.apply( Rotate(angle, interpolation method, clockwise) )
or
<y> = Rotate(angle, interpolation method, clockwise)( <x> )
```

Properties

any 2- or 3d array of integral type **x** [INPUT, MANDATORY, default=no default value]

the rotation angle in degrees (360 based) **angle** [INPUT, MANDATORY, default=no default value]

the interpolation method **interpolation method** [INPUT, NOT_MANDATORY, default=Rotate.NEAREST_NEIGHBOR]

the direction of rotation **clockwise** [INPUT, NOT_MANDATORY, default=true]

DoubleNd array (same rank as the input array) **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any 2- or 3d array of integral type x [INPUT, MANDATORY, default=no default value]

the rotation angle in degrees (360 based) angle [INPUT, MANDATORY, default=no default value]

the angle can be positive or negative

the interpolation method interpolation method [INPUT, NOT_MANDATORY, default=Rotate.NEAREST_NEIGHBOR]

the possible values for the interpolation method are

Rotate.NEAREST_NEIGHBOR = 1

Rotate.BILINEAR = 2

Rotate.BICUBIC = 3

the direction of rotation clockwise [INPUT, NOT_MANDATORY, default=true]

clockwise is a boolean and has the value True for clockwise rotation and False for counterclockwise rotation.

DoubleNd array (same rank as the input array) y [OUTPUT, MANDATORY, default=no default value]

Returns a Double2d array, if the input was a 2d array. Returns a Double3d array if the input was a 3d array. 3d arrays are treated as a stack of 2d arrays. Rotation is done around the depth (3rd) dimension.

3.282. rotate

Full Name:	herschel.ia.toolbox.image.RotateTask
Alias:	rotate
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import RotateTask
Category:	task/image

Description

A Task that rotates images.

A task which rotates an Image. The Wcs is also adapted. It is possible to use 4 types of interpolation :

- INTERP_NEAREST : nearest-neighbor interpolation. Nearest-neighbor interpolation is simply pixel copying
- INTERP_BILINEAR : Bilinear interpolation requires a neighborhood extending one pixel to the right and below the central sample. If the fractional subsample position is given by (xfrac, yfrac), the resampled pixel value will be:

```
(1 - yfrac) * [(1 - xfrac)*s00 + xfrac*s01] + yfrac * [(1 - xfrac)*s10 + xf\
rac*s11]
```

  A neighborhood extending one sample to the right of, and one sample below the central sample is required to perform bilinear interpolation. This implementation maintains equal subsampleBits in x and y.

- INTERP_BICUBIC : InterpolationBicubic is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
 r(x) = (a + 2)|x|^3 - (a + 3)|x|^2  +  1 , 0 <= |x| < 1
r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a , 1 <= |x| < 2
r(x) = 0, otherwise
```

  with 'a' set to -0.5. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Rifman. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

  A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

- INTERP_BICUBIC_2 : InterpolationBicubic2 is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
 r(x) = (a + 2)|x|^3 - (a + 3)|x|^2  +  1 , 0 <= |x| < 1
r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a , 1 <= |x| < 2
r(x) = 0, otherwise
```

  with 'a' set to -1.0. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Keys. This interpolator may produce somewhat sharper results than InterpolationBicubic, but that result is image dependent. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

  A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

When no interpolation is set, BiLinear interpolation (INTERP_BILINEAR) is taken as standard. INTERP_BICUBIC and INTERP_BICUBIC_2 need an extra parameter (subsample precision, in bits)

to be set. If the subsample precision is not set explicitely, the value will be 16. The errors are not calculated. At the moment, the same operation is executed on the errors. The Wcs is not always calculated exactly.

Examples

Example 1: Rotating an image over 12.2 degrees (counterclockwise for astronomical image, clockwise for other images)

```
rotate(image = im, angle = 12.2)
```

Example 2: Rotating an image over 12.2 degrees (counterclockwise for astronomical image, clockwise for other images), using Bicubic interpolation, using 32 bits

```
rotate(image = im, angle = 12.2, interpolation = Rotate.INTERP_BICUBIC,
 subsampleBits = 32)
```

API Summary

Jython Syntax

```
rotate(image, 12.2, Rotate.INTERP_BICUBIC)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
float angle [INPUT, OPTIONAL, default=0.0]
int interpolation [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]
int subsampleBits [INPUT, OPTIONAL, default=16.0]
Image rotatedImage [OUTPUT, MANDATORY, default=No default value]
```

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The Image to rotate.

`float angle [INPUT, OPTIONAL, default=0.0]`

The amount of degrees to rotate the image (counterclockwise for astronomical image, clockwise for other images).

`int interpolation [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]`

The type of interpolation to use (INTERP_NEAREST, INTERP_NEAREST, INTERP_BICUBIC, INTERP_BICUBIC_2).

`int subsampleBits [INPUT, OPTIONAL, default=16.0]`

The number of bits to use for the interpolation (only if interpolation is INTERP_BICUBIC or INTERP_BICUBIC_2).

`Image rotatedImage [OUTPUT, MANDATORY, default=No default value]`

The rotated Image.

3.283. ROUND

Full Name:	herschel.ia.numeric.toolbox.basic.Round
Alias:	ROUND
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Round

Description

Gives a rounded result of an array.

It can perform the operation over a decimal position (second argument) or over the integer part (no second argument or equals to '0'). For rounding to the closest integer representation, the result is functionally equivalent to adding 1/2 and taking the floor of the result. The function returns a float array for float input array and double array for any other numeric array type.

Example

Example 1: Apply ROUND on a Double1d

```
x=Double1d([-0.5001, -0.5, -0.4999, 0.4999, 0.5, 0.5001, 1.1234, 2.6236])
print ROUND(x)   #[-1.0, 0.0,  0.0,  0.0, 1.0, 1.0, 1.0,   3.0]
print ROUND(x,3) #[-0.5, -0.5, -0.5, 0.5, 0.5, 0.5, 1.123, 2.624]
```

API Summary

Jython Syntax

```
<y>=ROUND(<x>[,<n>])
```

Properties

any array of any rank **x** [INPUT, MANDATORY, default=no default value]

decimal position for rounding **n** [INPUT, OPTIONAL, default=0]

float or double array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any array of any rank x [INPUT, MANDATORY, default=no default value]

The input array can be an array of any type (except String and Complex) of any rank.

decimal position for rounding n [INPUT, OPTIONAL, default=0]

The decimal position for rounding. By default, it rounds to the closest integer value.

float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a float array for float input array and double array for any other numeric array type.

See also

- CEIL
- FLOOR

3.284. RowByRowAverageSpectrum

Full Name:	herschel.ia.toolbox.spectrum.RowByRowAverageSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import RowByRowAverageSpectrum

Description

Task for averaging two {@link SpectrumContainer} datasets on a scan-by-scan basis.

Included here for backwards compatibility reasons.

History

- `2007-08-17 - meli`: initial.
- `2008-03-16 - meli`: major refactoring of the toolbox.

3.285. save

Full Name:	herschel.ia.toolbox.util.SaveTask
Alias:	save
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import SaveTask
Category:	task

Description

Save variables from the jython session to a file

The save task is used to save one or more variable from the jython global namespace into the specified file. Variables can be specified with a string containing a comma separated list of variable names, if no variables are specified the entire namespace is saved.

Please note that only Serializable variable can be saved.

Examples

Example 1: save all global variable names and values

```
save("foo.sav")
```

Example 2: save specific variable names and values

```
save("foo.sav","x,y,z")
```

Example 3: save specified variable names and values from the local namespace of a function

```
def my_function():
 x=100
 y=23
 save("test.sav","x,y", space=locals())
my_function()
```

API Summary

Jython Syntax

```
save(<filename>[, <variable>][, <space>])
```

Properties

```
String filename [INPUT, MANDATORY, default=No default value]
String variable [INPUT, OPTIONAL, default=""]
PyStringMap space [INPUT, OPTIONAL, default=globals()]
```

Limitations

Only Serializable variables can be saved

Miscellaneous

No miscellaneous

API details

Properties

String filename [INPUT, MANDATORY, default=No default value]

The name of the file to store the values of variables.

String variable [INPUT, OPTIONAL, default=""]

The comma-separated list of variables to save.

PyStringMap space [INPUT, OPTIONAL, default=globals()]

A jython dictionary (PyStringMap) containing the namespace of the variables, The choices available are:

1. locals() for using variables from the local namespace (i.e. from where the function is used)
2. globals() for using variables from the module where the function is used

When no space is specified the top level namespace is updated.

See also

- restore

History

- `2004-07-13 - NdC`: first release.
- `2009-01-14 - JDS`: Cleanup

3.286. scale

Full Name:	herschel.ia.toolbox.image.ScaleTask
Alias:	scale
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import ScaleTask
Category:	task/image

Description

A Task to scale images.

A task to scale images. The Wcs is also adapted. It is possible to use 4 types of interpolation :

- INTERP_NEAREST : nearest-neighbor interpolation. Nearest-neighbor interpolation is simply pixel copying
- INTERP_BILINEAR : Bilinear interpolation requires a neighborhood extending one pixel to the right and below the central sample. If the fractional subsample position is given by (xfrac, yfrac), the resampled pixel value will be:

```
(1 - yfrac) * [(1 - xfrac)*s00 + xfrac*s01] + yfrac * [(1 - xfrac)*s10 + xf\
rac*s11]
```

  A neighborhood extending one sample to the right of, and one sample below the central sample is required to perform bilinear interpolation. This implementation maintains equal subsampleBits in x and y.

- INTERP_BICUBIC : InterpolationBicubic is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
r(x) = (a + 2)|x|^3 - (a + 3)|x|^2 +  1 , 0 <= |x| < 1
r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a , 1 <= |x| < 2
r(x) = 0, otherwise
```

  with 'a' set to -0.5. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Rifman. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

  A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

- INTERP_BICUBIC_2 : InterpolationBicubic2 is a subclass of Interpolation that performs interpolation using the piecewise cubic polynomial:

```
 r(x) = (a + 2)|x|^3 - (a + 3)|x|^2 +  1 , 0 <= |x| < 1
r(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a , 1 <= |x| < 2
r(x) = 0, otherwise
```

  with 'a' set to -1.0. This definition is also sometimes known as "cubic convolution", using the parameter 'a' recommended by Keys. This interpolator may produce somewhat sharper results than InterpolationBicubic, but that result is image dependent. (Reference: Digital Image Warping, George Wolberg, 1990, pp 129-131, IEEE Computer Society Press, ISBN 0-8186-8944-7)

  A neighborhood extending one sample to the left of and above the central sample, and two samples to the right of and below the central sample is required to perform bicubic interpolation.

When no interpolation is set, BiLinear interpolation (INTERP_BILINEAR) is taken as standard. INTERP_BICUBIC and INTERP_BICUBIC_2 need an extra parameter (subsample precision, in bits)

to be set. If the subsample precision is not set explicitely, the value will be 16. The errors are not calculated. At the moment, the same operation is executed on the errors.

Examples

Example 1: Scaling an image by a factor 1.4 in x-direction, and -0.4 in y-direction (meaning flipping and scaling it by 0.4)

```
scale(image = im, x = 1.4, y = 0.4)
```

Example 2: Scaling an image by a factor 1.4 in x-direction, and 0.4 in y-direction, using Bicubic interpolation, using 32 bits

```
scale(image = im, x = 1.4, y = 0.4, interpolation = Scale.INTERP_BICUBIC,
 subsampleBits = 32)
```

API Summary

Jython Syntax

```
scale(image, 1.4, 0.4, ScaleTask.INTERP_BICUBIC, 32)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
float x [INPUT, OPTIONAL, default=1.0]
float y [INPUT, OPTIONAL, default=1.0]
int interpolation [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]
int subsampleBits [INPUT, OPTIONAL, default=16.0]
Image scaledImage [OUTPUT, MANDATORY, default=No default value]
```

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The input image that has to be scaled.

float x [INPUT, OPTIONAL, default=1.0]

The scale factor in x-direction.

float y [INPUT, OPTIONAL, default=1.0]

The scale factor in y-direction.

int interpolation [INPUT, OPTIONAL, default=Rotate.INTERP_NEAREST]

The type of interpolation to use (INTERP_NEAREST, INTERP_NEAREST, INTERP_BICUBIC, INTERP_BICUBIC_2).

int subsampleBits [INPUT, OPTIONAL, default=16.0]

The number of bits to use for the interpolation (only if interpolation is INTERP_BICUBIC or INTERP_BICUBIC_2).

Image scaledImage [OUTPUT, MANDATORY, default=No default value]
The scaled image.

3.287. SelectSpectrum

Full Name:	herschel.ia.toolbox.spectrum.SelectSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import SelectSpectrum

Description

Task for selecting individual spectra from a `SpectrumContainer` or fromw arrays of such and pack them in a new spectrum container.

Different selection schemes are available for specifying the selection:

- segments: The most simple scheme is to use 'segments' where you specify which point spectra and subsegments therein you want to select.
- selection_lookup: Specify a PyDictionary with column names as keys and a PyList of admissible values as values.
- selection: General selection model (type SelectionModel) that allows you to specify selections such as select all rows with values in a given column lying in a predefined interval.
- selection_index: Specify a PyList of indices that reference the point spectra included in the container.

The different options to specify selections can be combined. Here, an AND logic is adopted.

Example

Example 1: from jide:

```
from herschel.ia.toolbox.spectrum import SelectSpectrum
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
select = SelectSpectrum()
selected = select(ds=spectra, selection={"bbtype":[6031, 6032], "buffer":[2]})
selected = select(ds=spectra, selection=[0,1,2,3])
selected = select(ds=spectra, selection={"LoFrequency":(4000.0,5000.0)})
selected = select(ds=spectra, selection={"Chopper":([-4.4,5.9],0.1))
from herschel.ia.toolbox.spectrum.selections.models import RangesSelectionModel
selected = select(ds=spectra, selection=RangesSelectionModel("Chopper",
 [-4.4,5.9], 0.1)) # the same as above
```

API Summary

Properties
`Object` **`ds`** `[INPUT, OPTIONAL, default=no default value.]`
`Object` **`segments`** `[INPUT, OPTIONAL, default=no default value.]`
`Object` **`selection`** `[INPUT, OPTIONAL, default=None.]`
`PyDictionary|Map>` **`selection_lookup`** `[INPUT, OPTIONAL, default=no default value.]`
`PyList` **`selection_index`** `[INPUT, OPTIONAL, default=No default value.]`
`SpectrumContainer` **`result`** `[OUTPUT, OPTIONAL, default=no default value]`

API details

Properties

Object ds [INPUT, OPTIONAL, default=no default value.]

Input data to be processed by the task. Several types are possible:

- SpectrumContainer
- Array of SpectrumContainer, i.e. SpectrumContainer[]
- List of SpectrumContainer, i.e. List
- Any product with implementations of `SpectrumContainer` inside.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments to restrict on. There are two options available:

- Specify a PyList of segment indices or
- pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.

Object selection [INPUT, OPTIONAL, default=None.]

Specification of what point spectra select. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra to select.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also

- ManyToOneSpectrumTask

History

- `2007-06-01 - meli`: initial.
- `2007-07-13 - meli`: Javadoc updated.
- `2008-03-16 - meli`: major refactoring of the toolbox.
- `2008-10-03 - meli`: segments, javadoc

3.288. SerialArchive

Full Name:	herschel.ia.io.serial.SerialArchive
Alias:	SerialArchive
Type:	Java Class -
Import:	from herschel.ia.io.serial import SerialArchive
Category:	class

Description

A class for writing and reading serialized objects.

The SerialArchive provides a transparent way to store and retrieve Products as defined within the Herschel Data Processing software.

Example

Example 1: default usage:

```
from herschel.ia.io.serial import SerialArchive
# write/read a Product in a serialized file.
s = SerialArchive()
s.save("output.ser", Product())
p = s.load("output.ser")
```

API Summary

Methods
`Product load(InputStream stream)` Loads a Product from an input stream.
`Product load(String fileName)` Loads a Product from a serialized file.
`save(String fileName, [Optionally derived] Product product)` Saves a Product to a serialized file.
`createOutputStream(String fileName)` Creates an object output stream for storing in a file.
`createOutputStream(OutputStream file)` Creates an object output stream for storing in a file.
`createOutputStream(OutputStream stream)` Creates an object output stream for storing in a file.
`createInputStream(String fileName)` Creates an object input stream for reading from a file.
`createInputStream(String file)` Creates an object input stream for reading from a file.
`createInputStream(InputStream stream)` Creates an object input stream for reading from a file.

API details

Methods

Product load(InputStream stream)

Loads a Product from an input stream using a serial reader.

Argument

`InputStream` **`stream`** `[INPUT, MANDATORY, default=no default value]`

Input stream from where the Product is to be read.

Return

Product

An object of the herschel.ia.dataset.Product family.

Product load(String fileName)

Loads a Product from a serialized file using a serial reader.

Argument

`String` **`fileName`** `[INPUT, MANDATORY, default=no default value]`

Name of the serialized file.

Return

Product

An object of the herschel.ia.dataset.Product family.

save(String fileName, [Optionally derived] Product product)

Saves a Product to a serialized file with sufficient information to preserve the quantities of the data as well as the original dataset type and contents.

Arguments

`String` **`fileName`** `[INPUT, MANDATORY, default=no default value]`

Name of the serialized file

`[Optionally derived] Product` **`product`** `[INPUT, MANDATORY, default=no default value]`

An object of the herschel.ia.dataset.Product family.

createOutputStream(String fileName)

Creates an object output stream for storing objects in a file, by calling its writeObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropiately, even in the presence of different package names or class versions.

Argument

`String` **`fileName`** `[INPUT, MANDATORY]`

createOutputStream(OutputStream file)

Creates an object output stream for storing objects in a file, by calling its writeObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropiately, even in the presence of different package names or class versions.

Argument

createOutputStream(OutputStream file)

`OutputStream` **`file`** `[INPUT, MANDATORY, default=no default value]`

File in which the data is to be serialized

createOutputStream(OutputStream stream)

Creates an object output stream for storing objects in a file, by calling its writeObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropiately, even in the presence of different package names or class versions.

Argument

`OutputStream` **`stream`** `[INPUT, MANDATORY, default=no default value]`

Stream to be wrapped

createInputStream(String fileName)

Creates an object input stream for reading objects from a file, by calling its readObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropiately, even in the presence of different package names or class versions.

Argument

`String` **`fileName`** `[INPUT, MANDATORY]`

createInputStream(String file)

Creates an object input stream for reading objects from a file, by calling its readObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropiately, even in the presence of different package names or class versions.

Argument

`String` **`file`** `[INPUT, MANDATORY, default=no default value]`

Name of the serialized file

createInputStream(InputStream stream)

Creates an object input stream for reading objects from a file, by calling its readObject method. The returned stream treats Products and Datasets as as special case, so that intermediate classes (resolvers) can manage them appropiately, even in the presence of different package names or class versions.

Argument

`InputStream` **`stream`** `[INPUT, MANDATORY, default=no default value]`

Stream to be wrapped

See also

- FitsArchive

3.289. SerialClientPool

Full Name:	herschel.ia.pal.pool.serial.SerialClientPool
Type:	Java Class -
Import:	from herschel.ia.pal.pool.serial import SerialClientPool

Description

A ProductPool implementation used for accessing products stored on a

remote pool across the network.

In order to use this, two things need to be set up first at the remote site:

1. A ProductPool implementation should be created.
2. A SerialServer is running, that talks to the remote ProductPool and relays data back/forth to the SerialClientPool via an accessible network port.

Make a note of the URL at which the server is running (eg "myserver.esa.int"), the ID of the remote product pool, and the port number at which data is relayed.

Examples

Example 1: Creating the remote pool and SerialServer. The SerialServer relays

```
data via port 4444 server=SerialServer(4444,
SimplePool.getInstance())
```

Example 2: Create a SerialClientPool that talks to the remote pool of ID

```
"simple.default" at site of URL "remotehost.esa.int", port 4444
storage=ProductStorage()
storage.register(SerialClientPool("myserver.esa.int", 4444,
"simple.default" ))
```

API Summary

Constructor
SerialClientPool(String hostName, String portNumber, String id)
Creates an instance of a SerialClientPool connected to a remote

API details

Constructor

SerialClientPool(String hostName, String portNumber, String id)

pool.

Arguments

String **hostName** [INPUT, MANDATORY, default=The URL at which the]

remote server is running, eg "remotehost.esa.int"

SerialClientPool(String hostName, String portNumber, String id)

`String` **portNumber** `[INPUT, MANDATORY, default=The port number on which]`

the remote server is running, eg 4444

`String` **id** `[INPUT, MANDATORY, default=The identifier for the remote]`

pool, eg "simple.mypool"

Return

SerialClientPool

An instance of the SerialClientPool.

3.290. SHIFT

Full Name:	herschel.ia.numeric.toolbox.basic.Shift
Alias:	SHIFT
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Shift

Description

Creates a new array out of the input array with a circular shift applied to

all elements along specified dimension.

Examples

Example 1: Apply SHIFT on a Int1d

```
x=Int1d([0,1,2,3,4,5,6,7,8,9,10,11])
print SHIFT(x,-3)     # [3,4,5,6,7,8,9,10,11,0,1,2]
```

Example 2: Apply SHIFT on a Int2d

```
x=Int2d([[0,1,2,3],[4,5,6,7],[8,9,10,11]])
print SHIFT(x,2)     # [[4,5,6,7],[8,9,10,11],[0,1,2,3]]
print SHIFT(x,2,1)   # [[2,3,0,1],[6,7,4,5],[10,11,8,9]]
```

API Summary

Jython Syntax

```
<y>=SHIFT(<x>,<shift>,<dimension=0>)
```

Properties

```
any array of any rank x [INPUT, MANDATORY, default=no default value]
integer shift [INPUT, MANDATORY, default=no default value]
integer dimension [INPUT, MANDATORY, default=no default value]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any array of any rank x [INPUT, MANDATORY, default=no default value]

The input array can be an array of rank 1

integer shift [INPUT, MANDATORY, default=no default value]

The number of element to shift

integer dimension [INPUT, MANDATORY, default=no default value]

The dimension to apply the shift to

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array with the same number of elements as the input array and with the reverse order.

3.291. Short1d

Full Name:	herschel.ia.numeric.Short1d
Type:	Java Class -
Import:	from herschel.ia.numeric import Short1d

Description

A rectangular numeric short array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.292. Short2d

Full Name:	herschel.ia.numeric.Short2d
Type:	Java Class -
Import:	from herschel.ia.numeric import Short2d

Description

A rectangular numeric short array of rank 2.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.293. Short3d

Full Name:	herschel.ia.numeric.Short3d
Type:	Java Class -
Import:	from herschel.ia.numeric import Short3d

Description

A rectangular numeric short array of rank 3.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.294. Short4d

Full Name:	herschel.ia.numeric.Short4d
Type:	Java Class -
Import:	from herschel.ia.numeric import Short4d

Description

A rectangular numeric short array of rank 4.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.295. Short5d

Full Name:	herschel.ia.numeric.Short5d
Type:	Java Class -
Import:	from herschel.ia.numeric import Short5d

Description

A rectangular numeric short array of rank 5.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.296. SIGCLIP

Full Name:	herschel.ia.numeric.toolbox.basic.Sigclip
Alias:	SIGCLIP
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Sigclip

Description

Sigclip finds values in an array that are more than n*(standard deviation) larger than a comparator (the median or mean

value). The comparator can be calculated for a running box of user defined size (behavior = filter - default) or for the whole array (behavior = clip). Sigclip replaces these higher values with the comparator. In filter mode Sigclip does not include the investigated Pixel for the computation of the comparator and the sigma. The user has the choice of 8 parameters: 1. envSize: the size of the environment of surrounding pixels. If i is the coordinate of the tested pixel, the environment will be [i-envSize, i+envSize] in the 1d case. For multi-dimenional arrays the box is extended in all dimensions. The default value for envSize is 3. 2. nsigma: the number of sigmas that a value has to exceed its next smaller value before it will be replaced. The default value is also 3. 3. The returnmode: Sigclip.RETURN_ARRAY (=0) returns a copy of the input array with the replaced values. Sigclip.RETURN_BOOL (=1) returns a boolean array where the clipped locations are marked as true (the other values are false). 4. the mode: Sigclip.MEAN or Sigclip.MEDIAN determines if median or mean of the neighbouring values is used to replace a sigclipped value 5. the treatment at the array edges. This is defined with the method setEdge(int edge). The choices are: Sigclip.TRUNCATE (default: scales down the boxsize), Sigclip.FILL_EDGEVALUE (fills the empty indices of the box with the last value of the array) , Sigclip.FILL_MEAN (fills the empty indices of the box with the mean of the rest of the box array) and Sigclip.FILL_MEDIAN (fills the empty indices of the box with the median of the rest of the box array) 6. the treatment of the outliers. outliers = "positve" removes only the values that are larger than the comparator (median or mean). This is the default. Other possible values are outliers = "negative" or outliers = "both". deprecation: the default value for outliers is deprecated. Currently it is "positive", but it will be changed to "both". The deprecation is best treated in your code by explicitely specifying the value for outliers! 7. The behavior as filter (default) or clip. behavior = "filter" uses the box of size env and lets Sigclip act as a filter. behavior = "clip" calculates the comparator mean or median and the sigma for the whole array and clips all array values according to these values 8. reduce1d reduced1d = true: in 1-dimensional arrays values are removed instead of being clipped. The length of the returned array becomes shorter reduced1d = false (default): values are clipped. The returned array has the same size as the input array

Example

Example 1: apply Sigclip to an Int1d array

```
array = Int1d.range(9)
array.set(5, 20)
print array
[0,1,2,3,4,20,6,7,8]
print array.apply( Sigclip() )
[0,1,2,3,4,5,6,7,8]
print array.apply( Sigclip(return = Sigclip.RETURN_BOOL) )
[false,false,false,false,false,true,false,false,false]
print array.apply( Sigclip( env=4, nsigma=2.5, mode=Sigclip.MEDIAN,
 edge=Sigclip.TRUNCATE, return=Sigclip.RETURN_BOOL) )
false,false,false,false,false,true,false,false,false]
array = Int1d(20, 9)
array.set(7, 20)
array.set(17, 0)
print array
```

Example 1: apply Sigclip to an Int1d array

```
[9,9,9,9,9,9,9,20,9,9,9,9,9,9,9,9,9,0,9,9]
print array.apply( Sigclip( reduce1d = True, outliers = "both") )
[9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9]
```

API Summary

Properties
any 1-3d array of integral type **x** [INPUT, MANDATORY, default=no default value]
the box size that is analysed for every sample **envSize** [INPUT, NOT_MANDATORY, default=3]
the number of sigmas that a value has to exceed the next smaller value to be sigclipped **nsigma** [INPUT, NOT_MANDATORY, default=3]
determines **returnmode** [INPUT, if a numeric or a boolean array is returned, default=NOT_MANDATORY]
determines **mode** [INPUT, if median or mean is used to replace sigclipped values, default=NOT_MANDATORY]
determines **outliers** [INPUT, if only positive, default=only negtive or both outliers are removed]
determines **behavior** [INPUT, if Sigclip acts as a filter or a classical clip, default=NOT_MANDATORY]
determines **reduce1d** [INPUT, if found values are removed from the returned array, default=NOT_MANDATORY]

API details

Properties

any 1-3d array of integral type x [INPUT, MANDATORY, default=no default value]

the box size that is analysed for every sample envSize [INPUT, NOT_MANDATORY, default=3]

the number of sigmas that a value has to exceed the next smaller value to be sigclipped nsigma [INPUT, NOT_MANDATORY, default=3]

determines returnmode [INPUT, if a numeric or a boolean array is returned, default=NOT_MANDATORY]

Sigclip.RETURN_ARRAY (=0) returns a copy of the input array with the replaced values.
Sigclip.RETURN_BOOL (=1) returns a boolean array where the clipped locations are marked as true (the other values are false).

determines mode [INPUT, if median or mean is used to replace sigclipped values, default=NOT_MANDATORY]

Sigclip.MEAN (=0) uses the mean of the environment to replace sigclipped values.
Sigclip.MEDIAN (=1) uses the median of the environment to replace sigclipped values.

determines outliers [INPUT, if only positive, default=only negtive or both outliers are removed]

"positive" (default) only outliers larger than the comparator are found "negative" only outliers smaller than the comparator are found "both" all outliers are found deprecation: the default value for outliers is deprecated. Currently it is "positive", but it will be changed to "both". The deprecation is best treated in your code by explicitely specifying the value for outliers!

determines behavior [INPUT, if Sigclip acts as a filter or a classical clip, default=NOT_MANDATORY]

"filter" (default) Sigclip behaves like a filter (runs a box of size env over the input array). "clip" Sigclip works on the whole array at once.

determines reduce1d [INPUT, if found values are removed from the returned array, default=NOT_MANDATORY]

"False" (default) the size of the returned array is the same as the size of the input array "True" the sigcliped values are removed from 1d arrays.

3.297. SimpleCube

Full Name:	herschel.ia.dataset.image.SimpleCube
Type:	Java Class -
Import:	from herschel.ia.dataset.image import SimpleCube
Category:	Image

Description

A Product describing cubes.

The Simplecube is a way to store and work with Cubes in the HCSS.

Example

Example 1: Creating a SimpleCube

```
s = SimpleCube()
s.setImage(myDouble3dImage)
```

API Summary

Constructors
SimpleCube()
The standard constructor
SimpleCube(SimpleCube copy)
The copy constructor
SimpleCube(String description)
Constructor with a description

Methods
Wcs **getWcs**()
Returns the World Coordinates System.
setImage(AbstractOrdered3dData image, Unit unit, String description)
Sets the cube.
setImage(AbstractOrdered3dData image, Unit unit)
Sets the cube.
setWcs(Wcs wcs)
Sets the world coordinates system.
getIntensity(int depth, double row, double column)
Returns the intensity
double **getIntensityWorldCoordinates**(int depth, double x, double y)
Returns the intensity of the cube
setIntensity(int depth, int row, int column, Number value)
Sets the intensity of the image.
Number **getPixel**(int depth, int row, int column)

Methods
Returns 1 pixel value.
`int` **`getDepth`**`()` Returns the number of layers
`Cube` **`getPreview`**`(float res)` Returns a preview of the cube
`AbstractOrdered3dData` **`getError`**`()` Returns the error of the cube as a Numeric3d.
`AbstractOrdered3dData` **`getExposure`**`()` Returns the exposure of the cube as a Numeric3d.
`Flag` **`getFlag`**`()` Returns the flag of the cube.
`int` **`getHeight`**`()` Returns the height.
`AbstractOrdered3dData` **`getImage`**`()` returns the cube as a Numeric3d.
`Unit` **`getUnit`**`()` Returns the unit.
`int` **`getWidth`**`()` Returns the width
`boolean` **`hasError`**`()` Checks whether the cube has an error.
`boolean` **`hasExposure`**`()` Checks whether the cube has an exposure.
`boolean` **`hasFlag`**`()` Checks when the cube has a flag.
`setUnit``(Unit unit)` Sets the unit.
`removeError``()` Removes the error.
`removeExposure``()` Removes the exposure.
`removeFlag``()` Removes the flag.
`setImage``(AbstractOrdered3dData cube)` Sets the cube.
`setError``(AbstractOrdered3dData error)` Sets the error.
`setError``(AbstractOrdered3dData error, String description)` Sets the error.
`setFlag``(Flag flag, String description)` Sets the flag.

Methods
setFlag(Flag flag) Sets the flag.
setExposure(AbstractOrdered3dData exposure, String description) Sets the exposure.
setExposure(AbstractOrdered3dData exposure) Sets the exposure.
int[] **getDimensions**() Returns the dimension.

API details

Constructors

SimpleCube()

A constructor which creates a standard SimpleCube. The standard SimpleCube consists of a Numeric3d for the image. The SimpleCube has the depth as the first (most slowly varying) index.

Example

Typical example on how to create an SimpleCube.

```
image=SimpleCube(description="ngc 6992", image=im, flag=flag, errors=er,
 unit=unit, wcs=wcs)
```

SimpleCube(SimpleCube copy)

Creates a new SimpleCube with the same contents of the given SimpleCube.

Argument

SimpleCube **copy** [INPUT, MANDATORY]

SimpleCube(String description)

Creates a SimpleCube with a given description.

Argument

String **description** [INPUT, MANDATORY]

Methods

Wcs getWcs()

Returns the World Coordinates System of the image.

Return

Wcs

The World Coordinates System of the image.

setImage(AbstractOrdered3dData image, Unit unit, String description)

Sets the cube. You can give a description to the cube and give a unit. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, the Exposure and the Flag are removed. If the new cube has the same

setImage(AbstractOrdered3dData image, Unit unit, String description)

dimensions, then the old values for the errors, exposure and flag are kept (but the unit of the errors is not).

Arguments

AbstractOrdered3dData **image** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

String **description** [INPUT, MANDATORY]

setImage(AbstractOrdered3dData image, Unit unit)

Sets the cube. You can give a unit. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, exposure and the flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposure and flag are kept (but the unit of the errors is not).

Arguments

AbstractOrdered3dData **image** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

setWcs(Wcs wcs)

Sets the world coordinates system of the cube.

Argument

Wcs **wcs** [INPUT, MANDATORY]

getIntensity(int depth, double row, double column)

Returns the intensity of the SimpleCube at a given point. If the pixel is flagged out, NaN is given back.

Arguments

int **depth** [INPUT, MANDATORY]

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

double getIntensityWorldCoordinates(int depth, double x, double y)

Returns the intensity of the vube at a given point in world coordinates. If the pixel is flagged out, NaN is given back.

Arguments

int **depth** [INPUT, MANDATORY]

double **x** [INPUT, MANDATORY]

double **y** [INPUT, MANDATORY]

Return

double

The intensity

setIntensity(int depth, int row, int column, Number value)

Sets the intensity of the image at a given point.

Arguments

setIntensity(int depth, int row, int column, Number value)

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Number **value** [INPUT, MANDATORY]

Number getPixel(int depth, int row, int column)

Returns 1 pixel value of the image.

Arguments

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

Number

1 pixel value of the image.

int getDepth()

Returns the depth of this SimpleCube. This is the first axis (slowly varying axis).

Return

int

Returns the depth of this SimpleCube.

Cube getPreview(float res)

Returns a new SimpleCube, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.

Argument

float **res** [INPUT, MANDATORY]

Return

Cube

The preview of the cube

AbstractOrdered3dData getError()

Returns the error of the cube as a Numeric3d containing the error of every pixel.

Return

AbstractOrdered3dData

The error

AbstractOrdered3dData getExposure()

Returns the exposure of the cube as a Numeric3d containing the exposure of every pixel.

Return

AbstractOrdered3dData

AbstractOrdered3dData getExposure()

The exposure

Flag getFlag()

Returns the flag of the cube.

Return

Flag

The flag

int getHeight()

Returns the height of this SimpleCube.

Return

int

Returns the height of this SimpleCube.

AbstractOrdered3dData getImage()

Returns the cube as a Numeric3d containing the data of the cube.

Return

AbstractOrdered3dData

The cube as a Numeric3d

Unit getUnit()

Returns the unit of the cube. The unit of the errors of this cube is the same as the unit of the cube.

Return

Unit

The unit

int getWidth()

Returns the width of this SimpleImage.

Return

int

Returns the width of this SimpleImage.

boolean hasError()

Returns true if the cube has an error.

Return

boolean

True if the error is set.

boolean hasExposure()

Returns true if the cube has an exposure.

boolean hasExposure()

Return

boolean

True if the exposure is set.

boolean hasFlag()

Returns true if the cube has a flag.

Return

boolean

True if the cube has a flag.

setUnit(Unit unit)

Sets the unit of the cube. Adapting the unit of the cube will also adapt the unit of the errors on the cube.

Argument

`Unit` **unit** [INPUT, MANDATORY]

removeError()

Removes the error, if an error exists.

removeExposure()

Removes the exposure, if an exposure exists.

removeFlag()

Removes the flag, if a flag exists.

setImage(AbstractOrdered3dData cube)

Sets the cube. If the new cube has other dimensions than the original cube, the errors, exposure and flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposure and flag are kept.

Argument

`AbstractOrdered3dData` **cube** [INPUT, MANDATORY]

setError(AbstractOrdered3dData error)

Sets the errors of the cube. The Numeric3d containing the errors should have the same dimensions as the cube. If this is not the case, the errors will not be adapted.

Argument

`AbstractOrdered3dData` **error** [INPUT, MANDATORY]

setError(AbstractOrdered3dData error, String description)

Sets the errors of the cube. The Numeric3d containing the errors should have the same dimensions as the cube. If this is not the case, the errors will not be adapted. The errors are also described.

Arguments

`AbstractOrdered3dData` **error** [INPUT, MANDATORY]

`String` **description** [INPUT, MANDATORY]

setFlag(Flag flag, String description)

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted. The flag is also described.

Arguments

Flag **flag** [INPUT, MANDATORY]

String **description** [INPUT, MANDATORY]

setFlag(Flag flag)

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted.

Argument

Flag **flag** [INPUT, MANDATORY]

setExposure(AbstractOrdered3dData exposure, String description)

Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.

Arguments

AbstractOrdered3dData **exposure** [INPUT, MANDATORY]

String **description** [INPUT, MANDATORY]

setExposure(AbstractOrdered3dData exposure)

Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.

Argument

AbstractOrdered3dData **exposure** [INPUT, MANDATORY]

int[] getDimensions()

Returns the dimension (depth, width, height) of this SimpleCube.

Return

int[]

Returns the dimension (depth, width, height) of this SimpleCube.

3.298. simpleFitsReader

Full Name:	herschel.ia.toolbox.util.SimpleFitsReaderTask
Alias:	simpleFitsReader
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import SimpleFitsReaderTask
Category:	task

Description

The SimpleFitsReaderTask task.

Creates a product from a FITS file.

Examples

Example 1: SimpleFitsReaderTask with just a file parameter

```
filepath = "path_to_file"
product=simpleFitsReader(file=filepath)
```

Example 2: SimpleFitsReaderTask with an optional reader type

```
filepath = "path_to_file"
readertype = SimpleFitsReaderTask.ReaderType.STANDARD
product=simpleFitsReader(file=filepath, reader=readertype)
```

API Summary

Jython Syntax

```
product=simpleFitsReader(<file>[, <reader>])
```

Properties

```
String file [INPUT, MANDATORY, default=null]
```

```
SimpleFitsReaderTask.ReaderType reader [INPUT, OPTIONAL,
default=SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD]
```

```
Product product [OUTPUT, MANDATORY, default=null]
```

API details

Properties

`String file [INPUT, MANDATORY, default=null]`

The path of the FITS file to be read.

`SimpleFitsReaderTask.ReaderType reader [INPUT, OPTIONAL, default=SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD]`

The strategy used to parse the contents. If an unrecognized value is typed, the default value is used. HCSS has priority over STANDARD as any FITS file can be read as STANDARD but only some of them are also HCSS FITS files. Possible values:

SimpleFitsReaderTask.ReaderType reader [INPUT, OPTIONAL, default=SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD]
• SimpleFitsReaderTask.ReaderType.HCSS_THEN_STANDARD (default): try to read the FITS Archive as an HCSS FITS file, if it fails then try to read it as an standard FITS file. • SimpleFitsReaderTask.ReaderType.HCSS : read it as an HCSS FITS file. • SimpleFitsReaderTask.ReaderType.STANDARD : read it as an STANDARD FITS file.

Product product [OUTPUT, MANDATORY, default=null]
The Product read from the FITS file.

History

- `2008-07-09 - JCS`: first release
- `2008-08-18 - JDS`: added optional parameter reader (type) to allow transparent failover reading
- `2009-01-14 - JDS`: SPR 5525: cleanup start

3.299. simpleFitsWriter

Full Name:	herschel.ia.toolbox.util.SimpleFitsWriterTask
Alias:	simpleFitsWriter
Type:	Java Task -
Import:	from herschel.ia.toolbox.util import SimpleFitsWriterTask
Category:	task

Description

Saves a product in a FITS file

SimpleFitsWriterTask flow:

- if no filename is given, flow is finished with RuntimeException "File is required."
- if the file already exists and warn is checked: user is asked to confirm overwriting
- if the user does not confirm overwriting (or mode is not interactive), flow is finished with RuntimeException "Execution cancelled."
- Otherwise the product is saved in HCSS' FITS format in the chosen file

Examples

Example 1: Saving a product into a file

```
p = Product()
file = "path_to_file"
simpleFitsWriter(product=p,file=file)
```

Example 2: Saving a product into a file asking before overwriting

```
p = Product()
file = "path_to_file"
simpleFitsWriter(product=p,file=file,warn=True)
```

API Summary

Jython Syntax

```
simpleFitsWriter(<product>, <file>[, <warn>])
```

Properties

```
Product product [INPUT, MANDATORY, default=null]
String file [INPUT, MANDATORY, default=null]
Boolean warn [INPUT, OPTIONAL, default=false]
```

API details

Properties

`Product product [INPUT, MANDATORY, default=null]`

Product to be saved.

String file [INPUT, MANDATORY, default=null]
FITS filename to save the product into.

Boolean warn [INPUT, OPTIONAL, default=false]
If true, asks confirmation before overwriting.

History

- `2008-07-08 - JCS`: first release
- `2008-08-19 - JDS`: Using PopUpDialog, SCR 4573: asking before overwriting
- `2008-11-24 - JDS`: SPR 5525: cleanup start

3.300. SimpleImage

Full Name:	herschel.ia.dataset.image.SimpleImage
Type:	Java Class -
Import:	from herschel.ia.dataset.image import SimpleImage
Category:	Image

Description

A Product describing images.

SimpleImage is a Product describing images.

Example

Example 1: Creating a SimpleImage

```
s = SimpleImage()
s.setImage(myDouble2dImage)
```

API Summary

Constructors

`SimpleImage()`
The standard constructor

`SimpleImage(SimpleImage copy)`
Copy constructor

`SimpleImage(String description)`
Constructor with a description

Methods

`setImage(AbstractOrdered2dData image, Unit unit, String description)`
Sets the image.

`setWcs(Wcs wcs)`
Sets the world coordinates system.

`Wcs getWcs()`
Returns the World Coordinates System.

`setImage(AbstractOrdered2dData image, Unit unit)`
Sets the image.

`getIntensity(double row, double column)`
Returns the intensity

`double getIntensityWorldCoordinates(double x, double y)`
Returns the intensity of the image

`setIntensity(int row, int column, Number value)`
Sets the intensity of the image.

`Number getPixel(int row, int column)`
Returns 1 pixel value.

Methods
`Image` **`getPreview`**`(float res)` Returns a preview of the image
`AbstractOrdered2dData` **`getError`**`()` Returns the error of the image as a Numeric2d.
`AbstractOrdered2dData` **`getExposure`**`()` Returns the exposure of the image as a Numeric2d.
`Flag` **`getFlag`**`()` Returns the flag of the image.
`int` **`getHeight`**`()` Returns the height.
`AbstractOrdered2dData` **`getImage`**`()` returns the image as a Numeric2d.
`Unit` **`getUnit`**`()` Returns the unit.
`int` **`getWidth`**`()` Returns the width
`double` **`getWavelength`**`()` Returns the reference wavelength
`double` **`getWavelength`**`(Length unit)` Returns the reference wavelength
`boolean` **`hasError`**`()` Checks whether the image has an error.
`boolean` **`hasExposure`**`()` Checks whether the image has an exposure.
`boolean` **`hasFlag`**`()` Checks when the image has a flag.
`setUnit``(Unit unit)` Sets the unit.
`setWavelength``(double wavelength)` Sets the reference wavelength.
`setWavelength``(double wavelength, Length unit)` Sets the reference wavelength.
`removeError``()` Removes the error.
`removeExposure``()` Removes the exposure.
`removeFlag``()` Removes the flag.
`setImage``(AbstractOrdered2dData image)` Sets the image.
`setError``(AbstractOrdered2dData error)`

Methods
Sets the error.
setError(AbstractOrdered2dData error, String description) Sets the error.
setFlag(Flag flag, String description) Sets the flag.
setFlag(Flag flag) Sets the flag.
setExposure(AbstractOrdered2dData exposure, String description) Sets the exposure.
setExposure(AbstractOrdered2dData exposure) Sets the exposure.
double **getFrequency**() Returns the reference frequency.
double **getFrequency**(Frequency freq) Returns the reference frequency.
setFrequency(double frequency) Sets the reference frequency.
setFrequency(double frequency, Frequency unit) Sets the reference frequency.
int[] **getDimensions**() Returns the dimension.
Double2d **getImageData**() Returns the Image data.

API details

Constructors

SimpleImage()

A constructor which creates a standard SimpleImage. The standard SimpleImage consists of a Numeric2d for the image, no error and no integration time. The dimension of the standard SimpleImage is 0x0. The reference wavelength is set to 0.0 microns.

Example

Typical example on how to create an SimpleImage.

```
image=SimpleImage(description="ngc 6992", image=im, flag=flag, errors=er,
 unit=unit, wavelength=wavelength, wcs=wcs)
```

SimpleImage(SimpleImage copy)

Constructor which makes a copy from an existent SimpleImage.

Argument

SimpleImage **copy** [INPUT, MANDATORY]

SimpleImage(String description)

Creates a SimpleImage with a given description.

SimpleImage(String description)

Argument

String **description** [INPUT, MANDATORY]

Methods

setImage(AbstractOrdered2dData image, Unit unit, String description)

Sets the image. You can give a description to the image and give a unit. The given unit will automatically also be used for the errors on this image. If the new image has other dimensions than the original image, the errors, exposure and the flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and flag are kept (but the unit of the errors is not).

Arguments

AbstractOrdered2dData **image** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

String **description** [INPUT, MANDATORY]

setWcs(Wcs wcs)

Sets the world coordinates system of the image.

Argument

Wcs **wcs** [INPUT, MANDATORY]

Wcs getWcs()

Returns the World Coordinates System of the image.

Return

Wcs

The World Coordinates System of the image.

setImage(AbstractOrdered2dData image, Unit unit)

Sets the image. You can give a unit. The given unit will automatically also be used for the errors on this image. If the new image has other dimensions than the original image, the errors, the exposure and the flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and the flag are kept (but the unit of the errors is not).

Arguments

AbstractOrdered2dData **image** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

getIntensity(double row, double column)

Returns the intensity of the SimpleImage at a given point. If the pixel is flagged out, NaN is given back.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

double getIntensityWorldCoordinates(double x, double y)

Returns the intensity of the image at a given point in world coordinates. If the pixel is flagged out, NaN is given back.

double getIntensityWorldCoordinates(double x, double y)

Arguments

double **x** [INPUT, MANDATORY]

double **y** [INPUT, MANDATORY]

Return

double

The intensity

setIntensity(int row, int column, Number value)

Sets the intensity of the image at a given point.

Arguments

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Number **value** [INPUT, MANDATORY]

Number getPixel(int row, int column)

Returns 1 pixel value of the image.

Arguments

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

Number

1 pixel value of the image.

Image getPreview(float res)

Returns a new SimpleImage, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.

Argument

float **res** [INPUT, MANDATORY]

Return

Image

The preview of the image

AbstractOrdered2dData getError()

Returns the error of the image as a Numeric2d containing the error of every pixel.

Return

AbstractOrdered2dData

The error

AbstractOrdered2dData getExposure()

Returns the exposure of the image as a Numeric2d containing the exposure of every pixel.

Return

AbstractOrdered2dData getExposure()

AbstractOrdered2dData

The exposure

Flag getFlag()

Returns the flag of the image.

Return

Flag

The flag

int getHeight()

Returns the height of this SimpleImage.

Return

int

The height of this SimpleImage.

AbstractOrdered2dData getImage()

Returns the image as a Numeric2d containing the data of the image.

Return

AbstractOrdered2dData

The image as a Numeric2d

Unit getUnit()

Returns the unit of the image. The unit of the errors of this image is the same as the unit of the image.

Return

Unit

The unit

int getWidth()

Returns the width of this SimpleImage.

Return

int

The width of this SimpleImage.

double getWavelength()

Returns the reference wavelength of the image.

Return

double

The reference wavelength

double getWavelength(Length unit)

Returns the reference wavelength of the image.

Argument

Length **unit** [INPUT, MANDATORY]

Return

double

The reference wavelength

boolean hasError()

Returns true if the image has an error.

Return

boolean

True if the error is set.

boolean hasExposure()

Returns true if the image has an exposure.

Return

boolean

True if the exposure is set.

boolean hasFlag()

Returns true if the image has a flag.

Return

boolean

True if the image has a flag.

setUnit(Unit unit)

Sets the unit of the image. Adapting the unit of the image will also adapt the unit of the errors on the image.

Argument

Unit **unit** [INPUT, MANDATORY]

setWavelength(double wavelength)

Set the reference wavelength of the image in microns.

Argument

double **wavelength** [INPUT, MANDATORY]

setWavelength(double wavelength, Length unit)

Set the reference wavelength of the image.

Arguments

double **wavelength** [INPUT, MANDATORY]

Length **unit** [INPUT, MANDATORY]

removeError()

Removes the error, if an error exists.

removeExposure()

Removes the exposure, if an exposure exists.

removeFlag()

Removes the flag, if a flag exists.

setImage(AbstractOrdered2dData image)

Sets the image. If the new image has other dimensions than the original image, the errors, exposure and flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and flag are kept.

Argument

`AbstractOrdered2dData` **image** [INPUT, MANDATORY]

setError(AbstractOrdered2dData error)

Sets the errors of the image. The Numeric2d containing the errors should have the same dimensions as the image. If this is not the case, the errors will not be adapted.

Argument

`AbstractOrdered2dData` **error** [INPUT, MANDATORY]

setError(AbstractOrdered2dData error, String description)

Sets the errors of the image. The Numeric2d containing the errors should have the same dimensions as the image. If this is not the case, the errors will not be adapted. The errors are also described.

Arguments

`AbstractOrdered2dData` **error** [INPUT, MANDATORY]

`String` **description** [INPUT, MANDATORY]

setFlag(Flag flag, String description)

Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted. The flag is also described.

Arguments

`Flag` **flag** [INPUT, MANDATORY]

`String` **description** [INPUT, MANDATORY]

setFlag(Flag flag)

Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted.

Argument

`Flag` **flag** [INPUT, MANDATORY]

setExposure(AbstractOrdered2dData exposure, String description)

Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.

Arguments

`AbstractOrdered2dData` **exposure** [INPUT, MANDATORY]

setExposure(AbstractOrdered2dData exposure, String description)

`String` **description** [INPUT, MANDATORY]

setExposure(AbstractOrdered2dData exposure)

Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.

Argument

`AbstractOrdered2dData` **exposure** [INPUT, MANDATORY]

double getFrequency()

Returns the reference frequency of the image.

Return

double

The reference frequency of the image.

double getFrequency(Frequency freq)

Returns the reference frequency of the image.

Argument

`Frequency` **freq** [INPUT, MANDATORY]

Return

double

The reference frequency of the image.

setFrequency(double frequency)

Set the reference frequency of the image in gigahertz.

Argument

`double` **frequency** [INPUT, MANDATORY]

setFrequency(double frequency, Frequency unit)

Set the reference frequency of the image.

Arguments

`double` **frequency** [INPUT, MANDATORY]

`Frequency` **unit** [INPUT, MANDATORY]

int[] getDimensions()

Returns the dimension (width, height) of this SimpleImage.

Return

int[]

The dimension (width, height) of this SimpleImage.

Double2d getImageData()

Returns the image data as a Double2d.

Return

Double2d

Double2d getImageData()
The image data as Double2d

3.301. SimpleStack

Full Name:	herschel.ia.slice.image.SimpleStack
Type:	Java Class -
Import:	from herschel.ia.slice.image import SimpleStack
Category:	Image

Description

A SimpleStack is a stack of Images.

The SimpleStack is a way to store and work with a stack of images in the HCSS

Example

Example 1: Typical example on how to create a SimpleStack.

```
stack = SimpleStack(description="Some images")
stack.setImage(new SimpleImage())
```

API Summary

Constructors
SimpleStack()
The standard constructor
SimpleStack(SimpleStack copy)
The copy constructor
SimpleStack(String description)
The constructor with a description

Methods
copy()
Returns a copy.
addImage(Image image)
Adds a new Image to the stack
replaceImage(int layer, Image image)
Replaces an image of the stack.
int **getDepth**()
Returns the number of layers of this stack
SimpleImage **getImage**(int layer)
Returns the chosen image from the Stack.
boolean **hasExposure**(int layer)
Checks whether the given layer has an exposure.
boolean **hasError**(int layer)
Checks whether the given layer has an error.
boolean **hasFlag**(int layer)
Checks whether the given layer has a flag.
AbstractOrdered2dData **getExposure**(int layer)

Methods
Returns the exposure of the given layer as a Numeric2d.
`AbstractOrdered2dData` **`getError`**`(int layer)` Returns the error of the given layer as a Numeric2d.
`Flag` **`getFlag`**`(int layer)` Returns the flag of the given layer.
`setExposure``(int layer, AbstractOrdered2dData exposure)` Sets the exposure of a certain layer.
`setError``(int layer, AbstractOrdered2dData errors)` Sets the error of a certain layer.
`setFlag``(int layer, Flag flag)` Sets the flag of a certain layer.
`removeError``(int layer)` Removes the error of a certain layer.
`removeExposure``(int layer)` Removes the exposure of a certain layer.
`removeFlag``(int layer)` Removes the flag of a certain layer.
`double` **`getIntensity`**`(int layer, double row, double column)` Returns the intensity.
`setIntensity``(int layer, int row, int column, Number value)` Sets the intensity of the stack.
`double` **`getIntensityWorldCoordinates`**`(int layer, double x, double y)` Returns the intensity of the stack
`int[]` **`getDimensions`**`(int layer)` Return the dimensions of the layer.
`int` **`getHeight`**`(int layer)` Returns the height
`int` **`getWidth`**`(int layer)` Returns the width.
`Unit` **`getUnit`**`(int layer)` Returns the unit.
`setUnit``(int layer, Unit unit)` Sets the unit.
`setWavelength``(int layer, double wavelength)` Sets the reference wavelength.
`setWavelength``(int layer, double wavelength, Length unit)` Sets the reference wavelength.
`double` **`getWavelength`**`(int layer)` Returns the reference wavelength
`double` **`getWavelength`**`(int layer, Length unit)` Returns the reference wavelength

Methods
`Wcs getWcs(int layer)` Returns the world coordinates system.
`setWcs(int layer, Wcs wcs)` Sets the world coordinates system.
`Stack getPreview(float resolution)` Returns a preview of the stack
`removeImage(int layer)` Removes a layer.

API details

Constructors

`SimpleStack()`

A constructor which creates a standard SimpleStack.

Example

Typical example on how to create an SimpleStack.

```
image=SimpleStack(description="Some images")
```

`SimpleStack(SimpleStack copy)`

Makes a new SimpleStack from a copy of the given Stack.

Argument

`SimpleStack` **`copy`** `[INPUT, MANDATORY]`

`SimpleStack(String description)`

This constructor also sets the description of the SimpleStack

Argument

`String` **`description`** `[INPUT, MANDATORY]`

Methods

`copy()`

Returns a copy from this SimpleStack.

`addImage(Image image)`

Adds a new image to the stack. The images is added as the last image.

Argument

`Image` **`image`** `[INPUT, MANDATORY]`

`replaceImage(int layer, Image image)`

Replaces an image of the stack.

Arguments

`int` **`layer`** `[INPUT, MANDATORY]`

`Image` **`image`** `[INPUT, MANDATORY]`

`int getDepth()`

Returns the number of layers of this stack.

Return

int

Returns the number of layers of this stack.

`SimpleImage getImage(int layer)`

Returns the chosen layer of the Stack as a SimpleImage.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

SimpleImage

The chosen layer of the Stack.

`boolean hasExposure(int layer)`

Returns true if the layer has an exposure.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

boolean

True if the exposure of the given layer is set.

`boolean hasError(int layer)`

Returns true if the layer has an error.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

boolean

True if the error of the given layer is set.

`boolean hasFlag(int layer)`

Returns true if the layer has a flag.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

boolean

True if the flag of the given layer is set.

`AbstractOrdered2dData getExposure(int layer)`

Returns the exposure of the given layer as a Numeric2d containing the exposure of every pixel.

Argument

AbstractOrdered2dData getExposure(int layer)

int **layer** [INPUT, MANDATORY]

Return

AbstractOrdered2dData

The exposure

AbstractOrdered2dData getError(int layer)

Returns the error of the given layer as a Numeric2d containing the error of every pixel.

Argument

int **layer** [INPUT, MANDATORY]

Return

AbstractOrdered2dData

The error

Flag getFlag(int layer)

Returns the flag of the given layer.

Argument

int **layer** [INPUT, MANDATORY]

Return

Flag

The flag

setExposure(int layer, AbstractOrdered2dData exposure)

Set the exposure of the given layer.

Arguments

int **layer** [INPUT, MANDATORY]

AbstractOrdered2dData **exposure** [INPUT, MANDATORY]

setError(int layer, AbstractOrdered2dData errors)

Set the error of the given layer.

Arguments

int **layer** [INPUT, MANDATORY]

AbstractOrdered2dData **errors** [INPUT, MANDATORY]

setFlag(int layer, Flag flag)

Set the flag of the given layer.

Arguments

int **layer** [INPUT, MANDATORY]

Flag **flag** [INPUT, MANDATORY]

removeError(int layer)

Removes the error of the given layer.

Argument

removeError(int layer)

int **layer** [INPUT, MANDATORY]

removeExposure(int layer)

Remove the exposure of the given layer.

Argument

int **layer** [INPUT, MANDATORY]

removeFlag(int layer)

Removes the flag of the given layer.

Argument

int **layer** [INPUT, MANDATORY]

double getIntensity(int layer, double row, double column)

Returns the intensity of the stack at a given point. If the pixel is masked out, NaN is given back.

Arguments

int **layer** [INPUT, MANDATORY]

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Return

double

The intensity

setIntensity(int layer, int row, int column, Number value)

Sets the intensity of the stack at a given point.

Arguments

int **layer** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Number **value** [INPUT, MANDATORY]

double getIntensityWorldCoordinates(int layer, double x, double y)

Returns the intensity of the stack at a given point in world coordinates. If the pixel is masked out, NaN is given back.

Arguments

int **layer** [INPUT, MANDATORY]

double **x** [INPUT, MANDATORY]

double **y** [INPUT, MANDATORY]

Return

double

The intensity

int[] getDimensions(int layer)

Returns the dimension (width * height) of the given layer of this SimpleStack.

int[] getDimensions(int layer)

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

int[]

Returns the dimension (width * height) of the given layer this SimpleStack.

int getHeight(int layer)

Returns the height of the given layer of this SimpleStack.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

int

The height of the given layer of this SimpleStack.

int getWidth(int layer)

Returns the width of the given layer of this SimpleStack.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

int

The width of the given layer of this SimpleStack.

Unit getUnit(int layer)

Returns the unit of the given layer of the stack. The unit of the errors on this layer of the stack is the same as the unit of the image.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

Return

Unit

The unit

setUnit(int layer, Unit unit)

Sets the unit of the given layer of the stack. Adapting the unit of the image will also adapt the unit of the errors on the given layer of the stack.

Arguments

`int` **`layer`** `[INPUT, MANDATORY]`

`Unit` **`unit`** `[INPUT, MANDATORY]`

setWavelength(int layer, double wavelength)

Set the reference wavelength of the given layer of the stack in microns.

Arguments

`int` **`layer`** `[INPUT, MANDATORY]`

setWavelength(int layer, double wavelength)

double **wavelength** [INPUT, MANDATORY]

setWavelength(int layer, double wavelength, Length unit)

Set the reference wavelength of the given layer of the stack.

Arguments

int **layer** [INPUT, MANDATORY]

double **wavelength** [INPUT, MANDATORY]

Length **unit** [INPUT, MANDATORY]

double getWavelength(int layer)

Returns the reference wavelength of the given layer of the stack.

Argument

int **layer** [INPUT, MANDATORY]

Return

double

The reference wavelength

double getWavelength(int layer, Length unit)

Returns the reference wavelength of the given layer of the stack.

Arguments

int **layer** [INPUT, MANDATORY]

Length **unit** [INPUT, MANDATORY]

Return

double

The reference wavelength

Wcs getWcs(int layer)

Returns the World Coordinates System of the given layer of the stack.

Argument

int **layer** [INPUT, MANDATORY]

Return

Wcs

The world coordinates system

setWcs(int layer, Wcs wcs)

Sets the World Coordinates System of the given layer of the stack.

Arguments

int **layer** [INPUT, MANDATORY]

Wcs **wcs** [INPUT, MANDATORY]

Stack getPreview(float resolution)

Returns a preview of a stack with the same number of layers, but with a decrease in resolution in width and height.

`Stack getPreview(float resolution)`

Argument

`float` **`resolution`** `[INPUT, MANDATORY]`

Return

Stack

The preview

`removeImage(int layer)`

Removes a layer from the stack.

Argument

`int` **`layer`** `[INPUT, MANDATORY]`

3.302. SIN

Full Name:	herschel.ia.numeric.toolbox.basic.Sin
Alias:	SIN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Sin

Description

Computes the trigonometric sine of an number or array

Gives the sine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

Example

Example 1: Apply SIN on a Float1d

```
x=Float1d([0,0.5])
print  SIN(x) # [0.0,0.47942555]
```

API Summary

Jython Syntax

```
<y>=SIN(<x>)
```

Properties

```
any type x [INPUT, MANDATORY, default=no default value]
any type y [OUTPUT, NOT_MANDATORY, default=no default value]
```

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]

The input is in radians, and may be of any type.

any type y [OUTPUT, NOT_MANDATORY, default=no default value]

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- ARCSIN
- COS
- SINH
- TAN

3.303. SINH

Full Name:	herschel.ia.numeric.toolbox.basic.SinH
Alias:	SINH
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import SinH

Description

Computes the trigonometric hyperbolic sine of an number or array

Gives the hyperbolic sine of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

API Summary

Jython Syntax
`<y>=SINH(<x>)`

Properties
`any type x [INPUT, MANDATORY, default=no default value]`
`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

Miscellaneous

Does not work for complex values.

API details

Properties

`any type x [INPUT, MANDATORY, default=no default value]`

The input is in radians, and may be of any type.

`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- SIN
- ARCSIN

3.304. SKEWNESS

Full Name:	herschel.ia.numeric.toolbox.basic.Skewness
Alias:	SKEWNESS
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Skewness

Description

Yields the skewness of the elements in the input array.

The skewness is defined as the third moment divided by the standard deviation raised to the thrid power.

Example

Example 1: Apply SKEWNEES on a Float1d

```
x=Float1d([1,3,2,3,4])
print SKEWNESS(x) # -0.19430208
```

API Summary

Jython Syntax

```
<y>=SKEWNESS(<x>)
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
double y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double y [OUTPUT, MANDATORY, default=no default value]

Returns a double

See also

- KURTOSIS
- MEAN
- MEDIAN
- STDDEV

- VARIANCE

3.305. SkyAperturePhotometryExplorer

Full Name:	herschel.ia.gui.image.SkyAperturePhotometryExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import SkyAperturePhotometryExplorer

Description

An explorer for SkyAperturePhotometryProducts.

API Summary

Constructors
SkyAperturePhotometryExplorer() The constructor of a new SkyAperturePhotometryExplorer.
SkyAperturePhotometryExplorer(Object object) The constructor of a new SkyAperturePhotometryExplorer associated

Methods
Class **getVariableType()** Returns the expected variable type for this SkyAperturePhotometryExplorer.
JTable **getResultsTable()** Constructs and returns the results table for this SkyAperturePhotometryExplorer.

API details

Constructors

SkyAperturePhotometryExplorer()

SkyAperturePhotometryExplorer(Object object)

with the given object.

Argument

Object **object** [INPUT, MANDATORY]

Methods

Class getVariableType()

Return

Class

Returns the expected variable type for this SkyAperturePhotometryExplorer.

JTable getResultsTable()

Return

JTable

JTable getResultsTable()
Returns the results table for this SkyAperturePhotometryExplorer.

3.306. SkyAperturePhotometryProduct

Full Name:	herschel.ia.dataset.image.SkyAperturePhotometryProduct
Type:	Java Class -
Import:	from herschel.ia.dataset.image import SkyAperturePhotometryProduct

Description

A class to deal with the results of aperture photometry with a

circular target aperture and an annular/rectangular sky aperture.

API Summary

Constructor
SkyAperturePhotometryProduct() The constructor of a new SkyAperturePhotometryProduct.

Methods
setAlgorithm(int index) Sets the sky estimation algorithm for this SkyAperturePhotometryProduct to the
setResultsTable(Double2d resultsTable) Sets the results table for this SkyAperturePhotometryProduct to the
String **getAlgorithm**() Returns the String representation of the sky estimation algorithm for this
double **getSkyTotal**() Returns the total flux for this SkyAperturePhotometryProduct.
double **getNbOfSkyPixels**() Returns the number of pixels for the sky for this SkyAperturePhotometryProduct.
double **getSkyError**() Returns the error for the sky for this SkyAperturePhotometryProduct.
double **getTargetTotal**() Returns the total flux for the target (sky subtracted) for this
double **getNbOfTargetPixels**() Returns the number of pixels for the target (sky subtracted) for this
double **getIntensityPerTargetPixel**() Returns the intensity per pixel for the target (sky subtracted) for this
double **getTargetError**() Returns the error for the target (sky subtracted) for this

Miscellaneous

Aperture specific information is stored in meta data, all other data (results of calculations) are stored in TableDatasets.

API details

Constructor

SkyAperturePhotometryProduct()

Methods

setAlgorithm(int index)

algorithm with the given index.

Argument

`int` **index** [INPUT, MANDATORY]

setResultsTable(Double2d resultsTable)

given table.

Argument

`Double2d` **resultsTable** [INPUT, MANDATORY]

String getAlgorithm()

SkyAperturePhotometryProduct.

Return

String

Returns the String representation of the sky estimation algorithm for this SkyAperturePhotometryProduct.

double getSkyTotal()

Return

double

Returns the total flux for this SkyAperturePhotometryProduct.

double getNbOfSkyPixels()

Return

double

Returns the number of pixels for the sky for this SkyAperturePhotometryProduct.

double getSkyError()

Return

double

Returns the error for the sky for this SkyAperturePhotometryProduct.

double getTargetTotal()

SkyAperturePhotometryProduct.

Return

`double getTargetTotal()`

double

Returns the total flux for the target (sky subtracted) for this SkyAperturePhotometryProduct.

`double getNbOfTargetPixels()`

SkyAperturePhotometryProduct.

Return

double

Returns the number of pixels for the target (sky subtracted) for this SkyAperturePhotometryProduct.

`double getIntensityPerTargetPixel()`

SkyAperturePhotometryProduct.

Return

double

Returns the intensity per pixel for the target (sky subtracted) for this SkyAperturePhotometryProduct.

`double getTargetError()`

SkyAperturePhotometryProduct.

Return

double

Returns the error for the target (sky subtracted) for this SkyAperturePhotometryProduct.

3.307. SkyAperturePhotometryTask

Full Name:	herschel.ia.toolbox.image.SkyAperturePhotometryTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import SkyAperturePhotometryTask

Description

image, INPUT, Image, MANDATORY, No default value

The image.

API Summary

Properties
`Double` **`centerX`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Double` **`centerY`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`String` **`centerRA`** `[INPUT, OPTIONAL, default=Default value : "centerRA"]`
`String` **`centerDec`** `[INPUT, OPTIONAL, default=Default value : "centerDec"]`
`Double` **`radiusPixels`** `[INPUT, OPTIONAL, default=Default value : NaN]`
`Integer` **`radiusArcsec`** `[INPUT, OPTIONAL, default=Default value : 4]`
`String` **`algorithm`** `[INPUT, MANDATORY, default=Default value : NaN]`
`boolean` **`fractional`** `[INPUT, OPTIONAL, default=Default value : true]`
`ApperturePhotometryProduct` **`result`** `[OUTPUT, MANDATORY, default=No default value]`

Miscellaneous

The parameters concerning the (annular/rectangular) sky aperture are specified in the subclasses/subtasks (AnnularSkyAperturePhotometryTask and RectangularSkyAperturePhotometryTask).

API details

Properties

`Double centerX [INPUT, OPTIONAL, default=Default value : NaN]`

The x-pixel-coordinate of the target center.

`Double centerY [INPUT, OPTIONAL, default=Default value : NaN]`

The y-pixel-coordinate of the target center.

`String centerRA [INPUT, OPTIONAL, default=Default value : "centerRA"]`

The right ascension of the target center.

String centerDec [INPUT, OPTIONAL, default=Default value : "centerDec"]

The declination of the target center.

Double radiusPixels [INPUT, OPTIONAL, default=Default value : NaN]

The radius of the circular target aperture in pixels.

Integer radiusArcsec [INPUT, OPTIONAL, default=Default value : 4]

The radius of the circular target aperture in arcsec.

String algorithm [INPUT, MANDATORY, default=Default value : NaN]

The sky estimation algorithm.

boolean fractional [INPUT, OPTIONAL, default=Default value : true]

The type of pixels.

AperturePhotometryProduct result [OUTPUT, MANDATORY, default=No default value]

The result.

3.308. SlicedCube

Full Name:	herschel.ia.slice.image.SlicedCube
Type:	Java Class -
Import:	from herschel.ia.slice.image import SlicedCube
Category:	Image

Description

A SlicedCube is a large cube which is stored in slices to make the memory usage lower.

A SlicedCube is a large cube which is stored in slices to make the memory usage lower.

Example

Example 1: Typical example on how to create a SlicedCube.

```
cube = SlicedCube(description="My large cube")
cube.setImage(myData)
```

API Summary

Constructors
SlicedCube()
A constructor which creates a standard SlicedCube. The cube is sliced
SlicedCube(SlicedCube copy)
The copy constructor
SlicedCube(String description)
The constructor with a description

Methods
copy()
Returns a copy.
setSliceSize(int depth, int row, int column)
Sets the size of the slices of the cube.
int[] **getSliceSize**()
Returns the size of the slices of the cube.
SimpleCube **getSlice**(int sliceNumber)
Returns the chosen slice.
setImage(AbstractOrdered3dData cube, Unit unit)
Sets the cube.
Wcs **getWcs**()
Returns the World Coordinates System.
setWcs(Wcs wcs)
Sets the world coordinates system.
int[] **getDimensions**()
Returns the dimensions.
setImage(AbstractOrdered3dData cube)

Methods
Sets the cube.
`int` **`getHeight`**`()` Returns the height.
`int` **`getWidth`**`()` Returns the width
`int` **`getDepth`**`()` Returns the depth
`Unit` **`getUnit`**`()` Returns the unit.
`setUnit``(Unit unit)` Sets the unit.
`getIntensity``(int depth, double row, double column)` Returns the intensity
`Double` **`getIntensityWorldCoordinates`**`(int i, double x, double y)` Returns the intensity of the cube
`Number` **`getPixel`**`(int depth, int row, int column)` Returns 1 pixel value.
`setIntensity``(int depth, int row, int column, Number value)` Sets the intensity of the cube.
`AbstractOrdered3dData` **`getImage`**`(int sliceNumber)` returns the image data of the slice as a Numeric3d.
`AbstractOrdered3dData` **`getImage`**`()` returns the image as a Numeric2d.
`boolean` **`hasError`**`()` Checks whether the cube has an error.
`setError``(AbstractOrdered3dData error)` Sets the error.
`removeError``()` Removes the error.
`Number` **`getError`**`(int depth, int row, int column)` Returns the error of 1 pixel.
`AbstractOrdered3dData` **`getError`**`(int sliceNumber)` returns the error of the slice as a Numeric3d.
`AbstractOrdered3dData` **`getError`**`()` Returns the error of the cube as a Numeric3d.
`boolean` **`hasExposure`**`()` Checks whether the cube has an exposure.
`setExposure``(AbstractOrdered3dData exposure)` Sets the exposure.
`setExposure``(AbstractOrdered3dData exposure, String description)` Sets the exposure.

Methods
`removeExposure()` Removes the exposure.
`Number getExposure(int depth, int row, int column)` Returns the exposure of 1 pixel.
`AbstractOrdered3dData getExposure(int sliceNumber)` returns the exposure of the slice as a Numeric3d.
`AbstractOrdered3dData getExposure()` Returns the exposure of the cube as a Numeric3d.
`boolean hasFlag()` Checks when the image has a flag.
`setFlag(Flag flag)` Sets the flag.
`setFlag(Flag flag, String description)` Sets the flag.
`removeFlag()` Removes the flag.
`boolean getFlag(int depth, int row, int column)` Returns the flag of 1 pixel.
`Flag getFlag(int sliceNumber)` returns the flag of the slice.
`Flag getFlag()` Returns the flag of the cube.
`Cube getPreview(float res)` Returns a preview of the cube
`Double3d getImageData()` Returns the Image data.

API details

Constructors

`SlicedCube()`
in pieces of 50x50x50 pixels. Every slice is kept in a ListContext as a SimpleCube. The usage of a SlicedCube should be transparent to the user. Exactly the same methods are available on a SlicedCube as on a SimpleCube. **Example** Typical example on how to create a SlicedCube. `cube=SlicedCube(description="ngc 6992", image=im, flag=flag, errors=er, quantity=quant, wavelength=wavelength, wcs=wcs)`

`SlicedCube(SlicedCube copy)`
Makes a new SlicedCube from a copy of the given Cube. **Argument**

SlicedCube(SlicedCube copy)

SlicedCube **copy** [INPUT, MANDATORY]

SlicedCube(String description)

This constructor also sets the description of the SlicedCube

Argument

String **description** [INPUT, MANDATORY]

Methods

copy()

Returns a copy from this SlicedCube.

setSliceSize(int depth, int row, int column)

Sets the size of the slices of the cube. The standard size is 50x50x50. At this moment, the size should be set before adding data to the SlicedImage!

Arguments

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

int[] getSliceSize()

Returns the size of the slices of the cube. The standard size is 50x50x50.

Return

int[]

The size of the slices (in depth, rows and columns)

SimpleCube getSlice(int sliceNumber)

Returns the chosen slice of the cube as a SimpleCube.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

SimpleCube

The chosen slice of the Cube.

setImage(AbstractOrdered3dData cube, Unit unit)

Sets the cube. You can give a unit. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, exposure and flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposures and flag are kept (but the unit of the errors is not). The cube data will be automatically sliced.

Arguments

AbstractOrdered3dData **cube** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

Wcs getWcs()

Returns the World Coordinates System of the cube.

Wcs getWcs()

Return

Wcs

The World Coordinates System of the cube.

setWcs(Wcs wcs)

Sets the world coordinates system of the cube.

Argument

`Wcs` **wcs** `[INPUT, MANDATORY]`

int[] getDimensions()

Returns the dimensions (depth, width, height) of this SlicedCube.

Return

int[]

Returns the dimensions (depth, width, height) of this SlicedCube.

setImage(AbstractOrdered3dData cube)

Sets the cube. The description is SlicedCube and the unit is Scalar.ONE. The given unit will automatically also be used for the errors on this cube. If the new cube has other dimensions than the original cube, the errors, exposure and flag are removed. If the new cube has the same dimensions, then the old values for the errors, exposures and flag are kept (but the unit of the errors is not). The cube data will be automatically sliced.

Argument

`AbstractOrdered3dData` **cube** `[INPUT, MANDATORY]`

int getHeight()

Returns the height of this SimpleCube.

Return

int

The height of this SimpleCube.

int getWidth()

Returns the width of this SlicedCube.

Return

int

The width of this SlicedCube.

int getDepth()

Returns the depth of this SlicedCube.

Return

int

The depth of this SlicedCube.

Unit getUnit()

Returns the unit of the cube. The unit of the errors of this cube is the same as the unit of the cube.

Return

Unit

The unit

setUnit(Unit unit)

Sets the unit of the cube. Adapting the unit of the cube will also adapt the unit of the errors on the cube.

Argument

Unit **unit** [INPUT, MANDATORY]

getIntensity(int depth, double row, double column)

Returns the intensity of the SimpleCube at a given point. If the pixel is flagged out, NaN is given back.

Arguments

int **depth** [INPUT, MANDATORY]

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

Double getIntensityWorldCoordinates(int i, double x, double y)

Returns the intensity of the cube at a given point in world coordinates. If the pixel is flagged out, NaN is given back.

Arguments

int **i** [INPUT, MANDATORY]

double **x** [INPUT, MANDATORY]

double **y** [INPUT, MANDATORY]

Return

Double

The intensity

Number getPixel(int depth, int row, int column)

Returns 1 pixel value of the cube.

Arguments

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

Number

1 pixel value of the cube.

setIntensity(int depth, int row, int column, Number value)

Sets the intensity of the cube at a given point.

setIntensity(int depth, int row, int column, Number value)

Arguments

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Number **value** [INPUT, MANDATORY]

AbstractOrdered3dData getImage(int sliceNumber)

Returns the image data as a Numeric3d containing the data of the slice of the cube.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered3dData

The image data of the slice as a Numeric3d

AbstractOrdered3dData getImage()

Returns the image as a Numeric2d containing the data of the image.

Return

AbstractOrdered3dData

The image as a Numeric2d

boolean hasError()

Returns true if the cube has an error.

Return

boolean

True if the error is set.

setError(AbstractOrdered3dData error)

Sets the errors of the cube. The Numeric3d containing the errors should have the same dimensions as the cube. If this is not the case, the errors will not be adapted.

Argument

AbstractOrdered3dData **error** [INPUT, MANDATORY]

removeError()

Removes the error, if an error exists.

Number getError(int depth, int row, int column)

Returns the error of 1 pixel.

Arguments

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

Number getError(int depth, int row, int column)

Number

The error of 1 pixel.

AbstractOrdered3dData getError(int sliceNumber)

Returns the error as a Numeric3d.

Argument

`int` **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered3dData

The error of the slice as a Numeric3d

AbstractOrdered3dData getError()

Returns the error of the cube as a Numeric3d containing the error of every pixel.

Return

AbstractOrdered3dData

The error

boolean hasExposure()

Returns true if the cube has an exposure.

Return

boolean

True if the exposure is set.

setExposure(AbstractOrdered3dData exposure)

Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.

Argument

`AbstractOrdered3dData` **exposure** [INPUT, MANDATORY]

setExposure(AbstractOrdered3dData exposure, String description)

Sets the exposure of every pixel of the cube. The Numeric3d containing the exposure should have the same dimensions as the cube. If this is not the case, the exposure will not be adapted.

Arguments

`AbstractOrdered3dData` **exposure** [INPUT, MANDATORY]

`String` **description** [INPUT, MANDATORY]

removeExposure()

Removes the exposure, if an exposure exists.

Number getExposure(int depth, int row, int column)

Returns the exposure of 1 pixel.

Arguments

Number getExposure(int depth, int row, int column)

int **depth** [INPUT, MANDATORY]

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

Number

The exposure of 1 pixel.

AbstractOrdered3dData getExposure(int sliceNumber)

Returns the exposure as a Numeric3d.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered3dData

The exposure of the slice as a Numeric3d

AbstractOrdered3dData getExposure()

Returns the exposure of the cube as a Numeric3d containing the exposure of every pixel.

Return

AbstractOrdered3dData

The exposure

boolean hasFlag()

Returns true if the image has a flag.

Return

boolean

True if the image has a flag.

setFlag(Flag flag)

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted.

Argument

Flag **flag** [INPUT, MANDATORY]

setFlag(Flag flag, String description)

Sets the flag of every pixel of the cube. The Flag should have the same dimensions as the cube. If this is not the case, the Flag will not be adapted. The flag is also described.

Arguments

Flag **flag** [INPUT, MANDATORY]

String **description** [INPUT, MANDATORY]

removeFlag()

Removes the flag, if a flag exists.

boolean getFlag(int depth, int row, int column)

Returns the flag of 1 pixel.

Arguments

`int` **`depth`** `[INPUT, MANDATORY]`

`int` **`row`** `[INPUT, MANDATORY]`

`int` **`column`** `[INPUT, MANDATORY]`

Return

boolean

The flag of 1 pixel. True if flagged out.

Flag getFlag(int sliceNumber)

Returns the flag.

Argument

`int` **`sliceNumber`** `[INPUT, MANDATORY]`

Return

Flag

The flag of the slice

Flag getFlag()

Returns the flag of the cube.

Return

Flag

The flag

Cube getPreview(float res)

Returns a new SlicedImage, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.

Argument

`float` **`res`** `[INPUT, MANDATORY]`

Return

Cube

The preview of the cube

Double3d getImageData()

Returns the cube data as a Double3d.

Return

Double3d

The cube data as Double3d

3.309. SlicedImage

Full Name:	herschel.ia.slice.image.SlicedImage
Type:	Java Class -
Import:	from herschel.ia.slice.image import SlicedImage
Category:	Image

Description

A SlicedImage is a large image which is stored in slices to make the memory usage lower.

A SlicedImage is a large image which is stored in slices to make the memory usage lower.

Example

Example 1: Typical example on how to create a SlicedImage.

```
image = SlicedImage(description="My large image")
image.setImage(myData)
```

API Summary

Constructors
SlicedImage() A constructor which creates a standard SlicedImage. The image is sliced
SlicedImage(SlicedImage copy) The copy constructor
SlicedImage(String description) The constructor with a description

Methods
copy() Returns a copy.
setSliceSize(int row, int column) Sets the size of the slices of the image.
int[] **getSliceSize**() Returns the size of the slices of the image.
SimpleImage **getSlice**(int sliceNumber) Returns the chosen slice.
setImage(AbstractOrdered2dData image, Unit unit) Sets the image.
setWavelength(double wavelength) Sets the reference wavelength.
setWavelength(double wavelength, Length unit) Sets the reference wavelength.
double **getWavelength**() Returns the reference wavelength
double **getWavelength**(Length unit)

Methods
Returns the reference wavelength
`Wcs` **`getWcs`**`()` Returns the World Coordinates System.
`setWcs``(Wcs wcs)` Sets the world coordinates system.
`int[]` **`getDimensions`**`()` Returns the dimension.
`setImage``(AbstractOrdered2dData image)` Sets the image.
`int` **`getHeight`**`()` Returns the height.
`int` **`getWidth`**`()` Returns the width
`Unit` **`getUnit`**`()` Returns the unit.
`setUnit``(Unit unit)` Sets the unit.
`getIntensity``(double row, double column)` Returns the intensity
`double` **`getIntensityWorldCoordinates`**`(double x, double y)` Returns the intensity of the image
`Number` **`getPixel`**`(int row, int column)` Returns 1 pixel value.
`setIntensity``(int row, int column, Number value)` Sets the intensity of the image.
`AbstractOrdered2dData` **`getImage`**`(int sliceNumber)` returns the image data of the slice as a Numeric2d.
`AbstractOrdered2dData` **`getImage`**`()` returns the image as a Numeric2d.
`boolean` **`hasError`**`()` Checks whether the image has an error.
`setError``(AbstractOrdered2dData error)` Sets the error.
`removeError``()` Removes the error.
`Number` **`getError`**`(int row, int column)` Returns the error of 1 pixel.
`AbstractOrdered2dData` **`getError`**`(int sliceNumber)` returns the error of the slice as a Numeric2d.
`AbstractOrdered2dData` **`getError`**`()` Returns the error of the image as a Numeric2d.

Methods
`boolean` **`hasExposure`**`()` Checks whether the image has an exposure.
`setExposure``(AbstractOrdered2dData exposure)` Sets the exposure.
`setExposure``(AbstractOrdered2dData exposure, String description)` Sets the exposure.
`removeExposure``()` Removes the exposure.
`Number` **`getExposure`**`(int row, int column)` Returns the exposure of 1 pixel.
`AbstractOrdered2dData` **`getExposure`**`(int sliceNumber)` returns the exposure of the slice as a Numeric2d.
`AbstractOrdered2dData` **`getExposure`**`()` Returns the exposure of the image as a Numeric2d.
`boolean` **`hasFlag`**`()` Checks when the image has a flag.
`setFlag``(Flag flag)` Sets the flag.
`setFlag``(Flag flag, String description)` Sets the flag.
`removeFlag``()` Removes the flag.
`boolean` **`getFlag`**`(int row, int column)` Returns the flag of 1 pixel.
`Flag` **`getFlag`**`(int sliceNumber)` returns the flag of the slice.
`Flag` **`getFlag`**`()` Returns the flag of the image.
`Image` **`getPreview`**`(float res)` Returns a preview of the image
`Double2d` **`getImageData`**`()` Returns the Image data.
`double` **`getFrequency`**`()` Returns the reference frequency.
`double` **`getFrequency`**`(Frequency freq)` Returns the reference frequency.
`setFrequency``(double frequency)` Sets the reference frequency.
`setFrequency``(double frequency, Frequency unit)` Sets the reference frequency.

API details

Constructors

SlicedImage()

in pieces of 500x500 pixels. Every slice is kept in a ListContext as a SimpleImage. The usage of a SlicedImage should be transparent to the user. Exactly the same methods are available on a SlicedImage as on a SimpleImage.

Example

Typical example on how to create a SlicedImage.

```
image=SlicedImage(description="ngc 6992", image=im, flag=flag, errors=er,
 quantity=quant, wavelength=wavelength, wcs=wcs)
```

SlicedImage(SlicedImage copy)

Makes a new SlicedImage from a copy of the given Image.

Argument

SlicedImage **copy** [INPUT, MANDATORY]

SlicedImage(String description)

This constructor also sets the description of the SlicedImage

Argument

String **description** [INPUT, MANDATORY]

Methods

copy()

Returns a copy from this SlicedImage.

setSliceSize(int row, int column)

Sets the size of the slices of the image. The standard size is 500 rows and 500 columns. At this moment, the size should be set before adding data to the SlicedImage!

Arguments

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

int[] getSliceSize()

Returns the size of the slices of the image. The standard size is 500 rows and 500 columns.

Return

int[]

The size of the slices (in rows and columns)

SimpleImage getSlice(int sliceNumber)

Returns the chosen slice of the image as a SimpleImage.

Argument

int **sliceNumber** [INPUT, MANDATORY]

SimpleImage getSlice(int sliceNumber)

Return

SimpleImage

The chosen slice of the Image.

setImage(AbstractOrdered2dData image, Unit unit)

Sets the image. You can give a unit. The given unit will automatically also be used for the errors on this image. If the new image has other dimensions than the original image, the errors, exposure and flag are removed. If the new image has the same dimensions, then the old values for the errors, exposures and flag are kept (but the unit of the errors is not). The image data will be automatically sliced.

Arguments

AbstractOrdered2dData **image** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

setWavelength(double wavelength)

Set the reference wavelength of the image in microns.

Argument

double **wavelength** [INPUT, MANDATORY]

setWavelength(double wavelength, Length unit)

Set the reference wavelength of the image.

Arguments

double **wavelength** [INPUT, MANDATORY]

Length **unit** [INPUT, MANDATORY]

double getWavelength()

Returns the reference wavelength of the image.

Return

double

The reference wavelength

double getWavelength(Length unit)

Returns the reference wavelength of the image.

Argument

Length **unit** [INPUT, MANDATORY]

Return

double

The reference wavelength

Wcs getWcs()

Returns the World Coordinates System of the image.

Return

Wcs

Wcs getWcs()

The World Coordinates System of the image.

setWcs(Wcs wcs)

Sets the world coordinates system of the image.

Argument

Wcs **wcs** [INPUT, MANDATORY]

int[] getDimensions()

Returns the dimension (width, height) of this SlicedImage.

Return

int[]

The dimension (width, height) of this SlicedImage.

setImage(AbstractOrdered2dData image)

Sets the image. If the new image has other dimensions than the original image, the errors, exposure and flag are removed. If the new image has the same dimensions, then the old values for the errors, exposure and flag are kept.

Argument

AbstractOrdered2dData **image** [INPUT, MANDATORY]

int getHeight()

Returns the height of this SimpleImage.

Return

int

The height of this SimpleImage.

int getWidth()

Returns the width of this SlicedImage.

Return

int

The width of this SlicedImage.

Unit getUnit()

Returns the unit of the image. The unit of the errors of this image is the same as the unit of the image.

Return

Unit

The unit

setUnit(Unit unit)

Sets the unit of the image. Adapting the unit of the image will also adapt the unit of the errors on the image.

setUnit(Unit unit)

Argument

Unit **unit** [INPUT, MANDATORY]

getIntensity(double row, double column)

Returns the intensity of the SlicedImage at a given point. If the pixel is flagged out, NaN is given back.

Arguments

double **row** [INPUT, MANDATORY]

double **column** [INPUT, MANDATORY]

double getIntensityWorldCoordinates(double x, double y)

Returns the intensity of the image at a given point in world coordinates. If the pixel is flagged out, NaN is given back.

Arguments

double **x** [INPUT, MANDATORY]

double **y** [INPUT, MANDATORY]

Return

double

The intensity

Number getPixel(int row, int column)

Returns 1 pixel value of the image.

Arguments

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Return

Number

1 pixel value of the image.

setIntensity(int row, int column, Number value)

Sets the intensity of the image at a given point.

Arguments

int **row** [INPUT, MANDATORY]

int **column** [INPUT, MANDATORY]

Number **value** [INPUT, MANDATORY]

AbstractOrdered2dData getImage(int sliceNumber)

Returns the image data as a Numeric2d containing the data of the slice of the image.

Argument

int **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered2dData

AbstractOrdered2dData getImage(int sliceNumber)

The image data of the slice as a Numeric2d

AbstractOrdered2dData getImage()

Returns the image as a Numeric2d containing the data of the image.

Return

AbstractOrdered2dData

The image as a Numeric2d

boolean hasError()

Returns true if the image has an error.

Return

boolean

True if the error is set.

setError(AbstractOrdered2dData error)

Sets the errors of the image. The Numeric2d containing the errors should have the same dimensions as the image. If this is not the case, the errors will not be adapted.

Argument

`AbstractOrdered2dData` **`error`** `[INPUT, MANDATORY]`

removeError()

Removes the error, if an error exists.

Number getError(int row, int column)

Returns the error of 1 pixel.

Arguments

`int` **`row`** `[INPUT, MANDATORY]`

`int` **`column`** `[INPUT, MANDATORY]`

Return

Number

The error of 1 pixel.

AbstractOrdered2dData getError(int sliceNumber)

Returns the error as a Numeric2d.

Argument

`int` **`sliceNumber`** `[INPUT, MANDATORY]`

Return

AbstractOrdered2dData

The error of the slice as a Numeric2d

AbstractOrdered2dData getError()

Returns the error of the image as a Numeric2d containing the error of every pixel.

AbstractOrdered2dData getError()

Return

AbstractOrdered2dData

The error

boolean hasExposure()

Returns true if the image has an exposure.

Return

boolean

True if the exposure is set.

setExposure(AbstractOrdered2dData exposure)

Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.

Argument

`AbstractOrdered2dData` **exposure** [INPUT, MANDATORY]

setExposure(AbstractOrdered2dData exposure, String description)

Sets the exposure of every pixel of the image. The Numeric2d containing the exposure should have the same dimensions as the image. If this is not the case, the exposure will not be adapted.

Arguments

`AbstractOrdered2dData` **exposure** [INPUT, MANDATORY]

`String` **description** [INPUT, MANDATORY]

removeExposure()

Removes the exposure, if an exposure exists.

Number getExposure(int row, int column)

Returns the exposure of 1 pixel.

Arguments

`int` **row** [INPUT, MANDATORY]

`int` **column** [INPUT, MANDATORY]

Return

Number

The exposure of 1 pixel.

AbstractOrdered2dData getExposure(int sliceNumber)

Returns the exposure as a Numeric2d.

Argument

`int` **sliceNumber** [INPUT, MANDATORY]

Return

AbstractOrdered2dData

The exposure of the slice as a Numeric2d

AbstractOrdered2dData getExposure()

Returns the exposure of the image as a Numeric2d containing the exposure of every pixel.

Return

AbstractOrdered2dData

The exposure

boolean hasFlag()

Returns true if the image has a flag.

Return

boolean

True if the image has a flag.

setFlag(Flag flag)

Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted.

Argument

`Flag` **flag** [INPUT, MANDATORY]

setFlag(Flag flag, String description)

Sets the flag of every pixel of the image. The Flag should have the same dimensions as the image. If this is not the case, the Flag will not be adapted. The flag is also described.

Arguments

`Flag` **flag** [INPUT, MANDATORY]

`String` **description** [INPUT, MANDATORY]

removeFlag()

Removes the flag, if a flag exists.

boolean getFlag(int row, int column)

Returns the flag of 1 pixel.

Arguments

`int` **row** [INPUT, MANDATORY]

`int` **column** [INPUT, MANDATORY]

Return

boolean

The flag of 1 pixel. True if flagged out.

Flag getFlag(int sliceNumber)

Returns the flag.

Argument

`int` **sliceNumber** [INPUT, MANDATORY]

Return

Flag

Flag getFlag(int sliceNumber)

The flag of the slice

Flag getFlag()

Returns the flag of the image.

Return

Flag

The flag

Image getPreview(float res)

Returns a new SlicedImage, but with a lower resolution if res is smaller than 1. A higher resolution is returned, if res is larger than 1. The spatial regridding is based on linear interpolation which will, in general, not be suitable for scientific purposes.

Argument

`float res [INPUT, MANDATORY]`

Return

Image

The preview of the image

Double2d getImageData()

Returns the image data as a Double2d.

Return

Double2d

The image data as Double2d

double getFrequency()

Returns the reference frequency of the image.

Return

double

The reference frequency of the image.

double getFrequency(Frequency freq)

Returns the reference frequency of the image.

Argument

`Frequency freq [INPUT, MANDATORY]`

Return

double

The reference frequency of the image.

setFrequency(double frequency)

Set the reference frequency of the image in gigahertz.

Argument

setFrequency(double frequency)

```
double frequency [INPUT, MANDATORY]
```

setFrequency(double frequency, Frequency unit)

Set the reference frequency of the image.

Arguments

```
double frequency [INPUT, MANDATORY]
Frequency unit [INPUT, MANDATORY]
```

3.310. SmoothingTask

Full Name:	herschel.ia.toolbox.image.SmoothingTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import SmoothingTask

Description

An abstract Task for smoothing.

An abstract Task for smoothing images.

3.311. SmoothSpectrum

Full Name:	herschel.ia.toolbox.spectrum.SmoothSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import SmoothSpectrum

Description

Task for smoothing the segments included in a spectrum container according to a selected filter.

The other attributes found in the spectra are not processed but just copied to the result container.

Flags and weights can be included in the processing by setting the 'variant'-parameter accordingly.

API Summary

Properties
`SpectrumContainer` **`ds`** `[INPUT, MANDATORY, default=no default value.]`
`SpectrumContainer` **`result`** `[OUTPUT, MANDATORY, default=no default value.]`
`String` **`filter`** `[INPUT, OPTIONAL, default=no default value.]`
`Double` **`width`** `[INPUT, OPTIONAL, default=no default value.]`
`String` **`variant`** `[INPUT, OPTIONAL, default=no default value.]`
`String` **`edge`** `[INPUT, OPTIONAL, default="REPEAT".]`
`Boolean` **`center`** `[INPUT, OPTIONAL, default=True.]`
`Boolean` **`overwrite`** `[INPUT, OPTIONAL, default=False.]`

API details

Properties

`SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]`

Input container with the spectra to be smoothed.

`SpectrumContainer result [OUTPUT, MANDATORY, default=no default value.]`

Output container with the smoothed spectra.

`String filter [INPUT, OPTIONAL, default=no default value.]`

Filter (convolution) to be applied. Available are: "Box" and "Gaussian".

`Double width [INPUT, OPTIONAL, default=no default value.]`

Smoothing width parameter to configure the filter/convolution kernel. It is specified as a number of channels.

`String variant [INPUT, OPTIONAL, default=no default value.]`

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-weight" / "flux-flag-weight"

String edge [INPUT, OPTIONAL, default="REPEAT".]

Parameter to configure the behavior at the edges: "ZEROES", "CIRCULAR", "REPEAT".

Boolean center [INPUT, OPTIONAL, default=True.]

Parameter to configure the behavior of the convolution: If set to true the smoothed value is assigned to the center of the smoothing interval. Otherwise, it is assigned at the left edge of the smoothing interval.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein are overwritten.

History

- `2008-03-19 - meli`: initial.

3.312. SOLVE

Full Name:	herschel.ia.numeric.toolbox.matrix.doc.SOLVE.entry.urm.xml
Alias:	SOLVE
Type:	Unknown (XML-based documentation) -
Import:	from herschel.ia.numeric.toolbox.matrix.doc import SOLVE.entry.urm.xml

Description

Solves systems of linear equations.

This wrapper is deprecated. Please, use the MatrixSolve Java class.

Limitations

This is a Jython wrapper function for the numeric MatrixSolve(b)(A).

See also

- MatrixSolve

3.313. SORT

Full Name:	herschel.ia.numeric.toolbox.basic.Sort
Alias:	SORT
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Sort

Description

Sorts the specified array into ascending natural order.

Index sorting is available as a special case of SORT i.e. SORT.BY_INDEX which returns an index array. The SORT.IS_SORTED function tests if the given array is sorted.

Examples

Example 1: Apply SORT on a Double1d

```
x=Double1d([-1,4,2,7,2,-6,-8,3,2])
print SORT(x)              # [-8.0,-6.0,-1.0,2.0,2.0,2.0,3.0,4.0,7.0]
```

Example 2: Apply SORT.BY_INDEX

```
x=Double1d([-1,4,2,7,2,-6,-8,3,2])
print SORT.BY_INDEX(x)     # [6,5,0,2,4,8,7,1,3]
```

Example 3: Apply SORT.IS_SORTED

```
x=Double1d([-1,4,2,7,2,-6,-8,3,2])
print SORT.IS_SORTED(x)    # 0 (false)
```

API Summary

Jython Syntax

```
<y>=SORT(<x>)
```

Properties

```
any array x [INPUT, MANDATORY, default=no default value]
an array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any array x [INPUT, MANDATORY, default=no default value]

The input array can be an array of rank 1

an array y [OUTPUT, MANDATORY, default=no default value]

Returns an array with the same number of elements as the input array and sorted in the ascending natural order.

3.314. SourceExtractorTask

Full Name:	herschel.ia.toolbox.srcext.SourceExtractorTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.srcext import SourceExtractorTask
Category:	task

Description

This task extracts point sources from an HCSS SimpleImage image.

This task extracts point sources from an HCSS SimpleImage image.

Example

Example 1: Extract sources from a SimpleImage

```
# myImage is a SimpleImage
disp = Display(myImage)
# Extract a list of sources from the image
sourceList = sourceExtractor(
    image = myImage,         # Image name
    detThreshold = 0.2,      # Threshold in DAOPHOT H units
    fwhm = 17.0,             # FWHM of PRF (arcsec)
    #prf = "",               # FITS file containing PRF
    #corner1Ra = 0.3,        # Minimum RA to consider (degrees)
    #corner1Dec = 0.3,       # Minimum dec to consider (degrees)
    #corner2Ra = 0.5,        # Maximum RA to consider (degrees)
    #corner2Dec = 0.5,       # Maximum dec to consider (degrees)
    #algorithm = "daophot", # DAOPHOT only at present
    pixelRegion = 2.5,       # Source search radius in pixels
    )
# How many sources found?
print "Found", sourceList.getSize(), "sources."
# Display each source on the image
wcs = myImage.getWcs()
for source in sourceList.iterator():
    # Find the image position in pixels
    x, y = wcs.getPixelCoordinates(source.getRa(), source.getDec())
    # Display a circle around the source location
    disp.addCircle(x, y, 5, 2, java.awt.Color.YELLOW)
```

API Summary

Properties
SimpleImage **image** [INPUT, MANDATORY, default=no default value]
Double **detThreshold** [INPUT, MANDATORY, default=no default value]
Double **fwhm** [INPUT, MANDATORY, default=no default value]
SimpleImage **prf** [INPUT, OPTIONAL, default=no default value]
Double **pixelRegion** [INPUT, MANDATORY, default=no default value]
String **algorithm** [INPUT, OPTIONAL, default=default value: daophot]
Double **corner1Ra** [INPUT, OPTIONAL, default=default value: Ra of image pixel (0]
Double **corner1Dec** [INPUT, OPTIONAL, default=defaut value: Dec of image pixel (0]

Properties
Double **corner2Ra** [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1]
Double **corner2Dec** [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1]
SourceListProduct **inputSourceList** [INPUT, OPTIONAL, default=no default value]
Double **fluxPriorsLambda** [INPUT, OPTIONAL, default=no default value]
Boolean **fitBackground** [INPUT, OPTIONAL, default=no default value]
Boolean **subtractMedianBackground** [INPUT, OPTIONAL, default=no default value]
Boolean **returnPixelCoordinates** [INPUT, OPTIONAL, default=default value: False]

API details

Properties

SimpleImage image [INPUT, MANDATORY, default=no default value]

Image map to search for point sources

Double detThreshold [INPUT, MANDATORY, default=no default value]

Threshold at which a local maximum is detected

Double fwhm [INPUT, MANDATORY, default=no default value]

Width in arcsecs of a gaussian default beam profile. For SPIRE, the values should be:

- PSW : 18.6
- PMW : 24.0
- PLW : 35.2

SimpleImage prf [INPUT, OPTIONAL, default=no default value]

SimpleImage of a custom point response function. If defined, this will be used in preference to the default gaussian.

Double pixelRegion [INPUT, MANDATORY, default=no default value]

Radius around each pixel to consider when searching for local maxima.

String algorithm [INPUT, OPTIONAL, default=default value: daophot]

Algorithm to use to extract sources. Currently only DAOPHOT is implemented; in future releases other algorithms (notably the Bayesian SussExtractor) will be available.

Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0]

These "corner..." parameters locate two points on the image in world coordinates, defining opposite corners of a rectangle. Only sources within this rectangle will be returned in the output

Double corner1Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (0]

SourceListProduct. If these parameters are not used then the entire map will be searched. Otherwise, if at least one of these parameters are defined, then all four must be.

Double corner1Dec [INPUT, OPTIONAL, default=defaut value: Dec of image pixel (0]

See description for corner1Ra.

Double corner2Ra [INPUT, OPTIONAL, default=default value: Ra of image pixel (width-1]

See description for corner1Ra.

Double corner2Dec [INPUT, OPTIONAL, default=default value: Dec of image pixel (width-1]

See description for corner1Ra.

SourceListProduct inputSourceList [INPUT, OPTIONAL, default=no default value]

SourceListProduct containing a list of "known" source positions, at which the fluxes will be extracted.

Double fluxPriorsLambda [INPUT, OPTIONAL, default=no default value]

Lambda value for SUSSEXtractor flux priors: 0 = tophat, 1 = Jeffreys, >1 = deboost

Boolean fitBackground [INPUT, OPTIONAL, default=no default value]

SUSSEXtractor flag to fit background as a free parameter

Boolean subtractMedianBackground [INPUT, OPTIONAL, default=no default value]

SUSSEXtractor flag to subtract median background from image

Boolean returnPixelCoordinates [INPUT, OPTIONAL, default=default value: False]

Return x, y coordinates of sources in ra, dec columns. If image has a valid WCS, FWHM is assumed to be in arcsec; otherwise FWHM is assumed to be in pixels

3.315. Spectrum1d

Full Name:	herschel.ia.dataset.spectrum.Spectrum1d
Type:	Java Class -
Import:	from herschel.ia.dataset.spectrum import Spectrum1d
Category:	Datasets

Description

Spectrum1d is an extension of AbstractSpectrumDataset which in turn

is an extension of (Strict)TableDataset. It is a 1-dimensional incarnation of a SpectrumDataset.

Spectrum1d implements an iterator over spectral segmants when these are defined.

Within a Spectrum1d there are provisions for 4 (predefined) columns.

1. flux, a Double1d array. This flux column is more or less obligatory, otherwise there can't be much talk of a spectrum.

2. weight. A Double1d array of the same dimensionality as flux.

   weight = 0, means that the corresponding samples are irrelevant.

   weight = 1 / stdev^2, if such a value is available for the fluxes.

3. flag, an Int1d array of the same size as flux. The definition of flags is of course problem dependent. It is suggested to store the meaning of the flags in MetaData.

4. segment, an Int1d array of the same size as flux. The values within this array indicate to which segment the corresponding flux/weight/flag/wave belong. If this column is not present it is assumed the Spectrum1d contains only one {@link SpectralSegment spectral segment}.

The other part that a spectrum needs is a frequency or wavelength scale. See {@link herschel.ia.dataset.spectrum.AbstractSpectrumDataset}

Example

Example 1: In Jide:

```
#flux is a Double1d
#segment is a Int1d of the same length containing [1,1,...1,2,2,...2,3,3...]
s1 = Spectrum1d()
s1.setFlux( flux )
s1.setSegment( segment )
it = s1.iterator()
seg = s1.getSpectralSegment()
while it.hasNext() :
  print F.p( seg.getFlux() )        # gets the 1's, the 2's etc.
  seg = it.next()
```

Limitations

Spectrum1d still **is** a TableDataset and can be addressed as such. If you do so, the special methods of Spectrum1d (and its predecessors, AbstractSpectrumDataset and StrictTableDataset) are **not** guarandeed to work.

History

- 06-03-2006 DK.

3.316. Spectrum2d

Full Name:	herschel.ia.dataset.spectrum.Spectrum2d
Type:	Java Class -
Import:	from herschel.ia.dataset.spectrum import Spectrum2d
Category:	Datasets

Description

Spectrum2d is an extension of AbstractSpectrumDataset which in turn

is an extension of (Strict)TableDataset. It is a 2-dimensional incarnation of a SpectrumDataset.

Within a Spectrum2d there are provisions for 3 (predefined) columns.

1. flux, a Double2d array, where the first axis runs over the spectral dimension and the second axis runs over e.g. time. In the following this time index is called sequential index or sequindex.

   There is no requirement that either axis projects monotonically or equidistantly on the frequency/sequindex scale. It is not even necessary that they are independent, ie. frequency can change over sequindex. This flux column is more or less obligatory, otherwise there can't be much talk of a spectrum.

2. weight. An Double2d array of the same dimensionality as flux.

   weight = 0, means that the corresponding samples are irrelevant.

   weight = 1 / stdev^2, if such a value is available for the fluxes.

3. flag, an Int1d array of the same size as flux. The definition of flags is of course problem dependent. It is suggested to store the meaning of the flags in MetaData.

The other part that a spectrum needs is a frequency or wavelength scale. See {@link herschel.ia.dataset.spectrum.AbstractSpectrumDataset}

As a further extension of the Spectrum2d we introduce the concept of "subbands". Subbands are vertical splits in the flux (and weight, flag etc.) columns, equivalent to (functionality of) the segment column in {@link Spectrum1d}. The flux etc. columns are replaced with flux_1, flux_2, ... columns, depending on how many subbands were defined. The definition of the subbands is stored in small array MetaData subbandstart and subbandlength. {@link herschel.ia.dataset.spectrum.StrictTableDataset#setMeta(String name, Double1d data)} Operations of split and join are exact inverses of each other as long as the metadata of subbandstart and subbandlength is kept consistent and provided that the subband ranges cover the complete width of flux. Otherwise what is lost will stay so.

The {@link SpectralSegment SpectralSegment} interface is fully implemented on Spectrum2d, whether it has subbands or not. When there are no subbands, each time a next segment is requested, the next row is returned. When subbands are present, first a row from the next subband at the same sequential index is returned. When there are no more subbands, the row at the next sequindex in the first subband is returned. This behaviour can be overruled by {@link #setColumnFirst(boolean cf)}.

Example

Example 1: In Jide:

```
flux = Double2d()
for i in range(5) : flux.appendRow( Double1d( range(1000 ) ) + i )
```

Example 1: In Jide:

```
flag = Int2d( 5, 1000 )
s2 = Spectrum2d( flux, None, flag )
F = DataFormatter()
print F.p( s2.getFlux( ), 6 )
s2.setMeta( "subbandstart", Int1d( [0,100,500] ) )
s2.setMeta( "subbandlength", Int1d( [100,200,400] ) )
s2.splitInSegments( ["flux","flag"] );
print F.p( s2.getFlux( 2 ) )
print s2.getSegmentCount()            # 15 = 5 * 3
it = s2.iterator()
seg = s2.getSpectralSegment()
while it.hasNext() :
  seg = it.next()
  print F.p( seg.getFlux() )
```

Limitations

Spectrum2d still **is** a TableDataset and can be addressed as such. If you do so, the special methods of Spectrum2d (and its predecessors, AbstractSpectrumDataset and StrictTableDataset) are **not** guarandeed to work.

History

- 07-03-2006 DK.

3.317. SpectrumPlotting

Full Name:	herschel.ia.gui.cube.SpectrumPlotting
Type:	Java Class -
Import:	from herschel.ia.gui.cube import SpectrumPlotting

Description

SpectrumPlotting

A class to deal with CubeSpectrumAnalysis, the real creactomputation of the spectrum is made in the task ExtractSpectrum()

API Summary

Constructor
SpectrumPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox) Constructor for SpectrumPlotting. When the new SpectrumPlotting is

Methods
setClicked(Double center) Sets the center of the rectancle, associated with this
PlotXY **getPlot**() Returns the PlotXY, shown in this SpectrumPlotting.
ImageFigure **getRect**() Returns the straight line (ImageFigure), of which we wish to plot
Double **getClicked**() Returns the clicked position in PixelCoordinates.
int **getClicks**() Returns the number of valid clicks (i.e. in the image) you made.
void **resetClicks**() reste the number of valid clicks
String **getSkyCoordinates**() Returns the String version of the sky coordinates
String **getPixelCoordinates**() Returns the String version of the PixelCoordinates center of
setPlot() initiate the PlotXY, shown in this SpectrumPlotting.
boolean **checkPointclicked**() Returns a Flag
ArrayList **getImageFigures**() Returns all ImageFigures used for this SpectrumPlotting (the
Double1d **getSpectrum**() Returns a Double1d at this date containing the spectrum values

Methods
`void saveData()` Returns all ImageFigures used for this SpectrumPlotting (the
`void saveScript()` Returns all ImageFigures used for this SpectrumPlotting (the

API details

Constructor

`SpectrumPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)`

component-based, a window for plotting the intensity is opened; otherwise nothing will be shown.

Arguments

`boolean` **`useAsComponent`** `[INPUT, MANDATORY]`

`CubeSpectrumAnalysisToolbox` **`toolbox`** `[INPUT, MANDATORY]`

Methods

`setClicked(Double center)`

SpectrumPlotting to the given point (in UserCoordinates).

Argument

`Double` **`center`** `[INPUT, MANDATORY]`

`PlotXY getPlot()`

Return

PlotXY

The PlotXY, shown in this SpectrumPlotting.

`ImageFigure getRect()`

the intensity in this SpectrumPlotting.

Return

ImageFigure

The straight line (ImageFigure), of which we wish to plot the intensity in this SpectrumPlotting.

`Double getClicked()`

Return

Double

Returns the clicked position in PixelCoordinates.

`int getClicks()`

Return

int

int getClicks()

Returns the number of valid clicks (i.e. in the image) you made.

void resetClicks()

Return

void

reset the number of valid clicks

String getSkyCoordinates()

(WorldCoordinates) of the point Clicked.

Return

String

Returns the String version of the sky coordinates (WorldCoordinates) of the point Clicked.

String getPixelCoordinates()

the rectanlgle which is also the point analysed.

Return

String

Returns the String version of the PixelCoordinates.

setPlot()

boolean checkPointclicked()

Return

boolean

returns a Boolean Flag which allow the extraction of the spectrum at the given position.

ArrayList getImageFigures()

rectangle on the image).

Return

ArrayList

Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image).

Double1d getSpectrum()

Return

Double1d

Returns a Double1d at this date containing the spectrum values

void saveData()

rectangle on the image).

`void saveData()`

Return

void

Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image).

`void saveScript()`

rectangle on the image).

Return

void

Returns all ImageFigures used for this SpectrumPlotting (the rectangle on the image).

3.318. SpectrumStatistics

Full Name:	herschel.ia.toolbox.spectrum.SpectrumStatistics
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import SpectrumStatistics

Description

Task to compute statistical characteristics for the spectrum data included in one or several containers.

Mean, RMS and median are always reported - percentiles can be given on demand. Two modes are available: Compute the statistics over a set of PointSpectra on a channel by channel basis or compute the statistics for a range within a single point spectrum.

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import SpectrumStatistics
# ds: some spectrum container
statistics = SpectrumStatistics()
stats = statistics(ds=ds, mode="all") # stats is a product
stats = statistics(ds=ds, mode="perChannel") # stats is a product but without
 "summary" TableDataset
stats = statistics(ds=ds, mode="acrossChannels") # stats is a TableDataset
stats = statistics(ds=ds, mode = "all", percentiles=[0.2,0.8], ranges =
 Range(500,1500))
```

API Summary

Properties
`SpectrumContainer` **`ds`** `[INPUT, MANDATORY, default=no default value.]`
`String` **`mode`** `[INPUT, OPTIONAL, default=default value 'all'.]`
`Object` **`stats`** `[OUTPUT, OPTIONAL, default=no default value.]`
`Object` **`ranges`** `[INPUT, OPTIONAL, default=no default value.]`
`double[]` **`percentiles`** `[INPUT, OPTIONAL, default=no default value.]`

API details

Properties

`SpectrumContainer ds [INPUT, MANDATORY, default=no default value.]`

The input container(s) to be considered.

`String mode [INPUT, OPTIONAL, default=default value 'all'.]`

Contains the output with the statistics when the task is used in the normal mode. Possible values are 'acrossChannel', 'perChannel', 'all'. 'acrossChannel'-mode: Here, the statistics are computed separately for each spectrum and each segments. If a range has been set, this range is taken accordingly. 'perChannel'-mode: Here, the statistics are computed for all the spectra included in the container on a per channel basis. 'all': Both, the per channel and the across channel statistics are computed.

Object stats [OUTPUT, OPTIONAL, default=no default value.]

If the task is used in the per channel mode, a product is created which contains for each statistical characteristics and each sub-segment one Spectrum1d. In case 'mode' is set to 'all', the product also contains the across channel statistics as a "summary". In case the task is used in the across channel mode, the statistics are written in a TableDataset (each point spectrum included in the input spectrum container corresponds a line in the table).

Object ranges [INPUT, OPTIONAL, default=no default value.]

Specify the ranges that should be considered within the spectra. You have various alternatives to do that:

- Specify a 'Range'-object: Then this range object is considered for all the sub-segments as the range of pixel numbers to select.
- Specify an array of 'Range'-objects: Here, each Range object in the array is associated with the Range of pixel number to select for each sub-segments. The number of ranges specified in the array must correspond to the number of segments.
-
-

double[] percentiles [INPUT, OPTIONAL, default=no default value.]

Specify what percentiles should be given in the output (a list of probabilities).

History

- `2008-06-05 - meli`: initial.

3.319. SpectrumTask

Full Name:	herschel.ia.toolbox.spectrum.SpectrumTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import SpectrumTask

Description

Abstract base class for tasks processing {@link SpectrumContainer} data structures.

3.320. SpireDataFrameFactoryImpl

Full Name:	herschel.vanilla.ccm.spire.SpireDataFrameFactoryImpl
Type:	Java Class -
Import:	from herschel.vanilla.ccm.spire import SpireDataFrameFactoryImpl

History

- March 2004 : Original
- January 2005 : Use Number[] to set the data
- March 2005 : Implements dataframes with sequenceCount and packetTime
- December 2005: DataFrameDetails now adds package name to "classname"
- 8 May 2006 : Remove usage of deprecated DataFrame methods. Use generics
- 4 Oct 2006 : Add constructors using TmSource
- 20 Mar 2007 : Use getSourceName() instead of getModelName().
- 21 Jul 2008 : Remove useless methods. Fix method to copy spire data frames

3.321. SpireDataFrameFactoryImpl

Full Name:	herschel.versant.ccm.spire.SpireDataFrameFactoryImpl
Type:	Java Class -
Import:	from herschel.versant.ccm.spire import SpireDataFrameFactoryImpl

History

- March 2004 : Original
- January 2005 : Use Number[] to set the data
- March 2005 : Implements dataframes with sequenceCount and packetTime
- December 2005: DataFrameDetails now adds package name to "classname"
- 8 May 2006 : Remove usage of deprecated DataFrame methods. Use generics
- 4 Oct 2006 : Add constructors using TmSource
- 20 Mar 2007 : Use getSourceName() instead of getModelName().
- 21 Jul 2008 : Remove useless methods. Fix method to copy spire data frames

3.322. SpireFlags

Full Name:	herschel.ia.obs.quality.SpireFlags
Type:	Java Class -
Import:	from herschel.ia.obs.quality import SpireFlags

History

- 03 Jun 09: Renamed several "get" methods to address SPR 6995.

3.323. SQRT

Full Name:	herschel.ia.numeric.toolbox.basic.Sqrt
Alias:	SQRT
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Sqrt

Description

Computes x=#x, where x is a number or a numeric array.

Example

Example 1: Apply SQRT on a Int1d

```
x=Int1d( [0,4,9] )
print SQRT(x) # [0,2.,3.]
```

API Summary

Jython Syntax

```
<y>=SQRT(<x>)
```

Properties

```
any number or array x [INPUT, MANDATORY, default=no default value]
boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any number or array x [INPUT, MANDATORY, default=no default value]

The input can be a number or an array

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array (or a number if the input is not an array) where each element is the the # of the corresponding element of the input array

3.324. SQUARE

Full Name:	herschel.ia.numeric.toolbox.basic.Square
Alias:	SQUARE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Square

Description

Computes $x=x^2$, where x is a number or a numeric array.

Example

Example 1: Apply SQUARE on a Int1d

```
x=Int1d( [-1,0,2] )
print SQUARE(x) # [1,0,4]
```

API Summary

Jython Syntax
<y>=SQUARE(<x>)

Properties
any number or array **x** [INPUT, MANDATORY, default=no default value]
boolean array or a boolean **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any number or array x [INPUT, MANDATORY, default=no default value]

The input can be a number or an array

boolean array or a boolean y [OUTPUT, MANDATORY, default=no default value]

Returns an array (or a number if the input is not an array) where each element is the the square of the corresponding element of the input array

3.325. STDDEV

Full Name:	herschel.ia.numeric.toolbox.basic.StdDev
Alias:	STDDEV
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import StdDev

Description

Yields the standard deviation of the elements in the input array.

The standard deviation is equivalent to SQRT(VARIANCE(x)).

Example

Example 1: Apply STDDEV on a Float1d

```
x=Float1d([1,3,2,3,4])
print STDDEV(x) # 1.3
```

API Summary

Jython Syntax

```
<y>=STDDEV(<x>)
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
double y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double y [OUTPUT, MANDATORY, default=no default value]

Returns a double

See also

- MEAN
- MEDIAN
- VARIANCE

3.326. String1d

Full Name:	herschel.ia.numeric.String1d
Type:	Java Class -
Import:	from herschel.ia.numeric import String1d

Description

A rectangular numeric String array of rank 1.

Numeric arrays are part of the Numeric library. An explanation on how to use them, you can find in chapter **Numeric -> Arrays**. (Yes, it should be a link)

3.327. SubtractSpectrum

Full Name:	herschel.ia.toolbox.spectrum.SubtractSpectrum
Type:	Java Task -
Import:	from herschel.ia.toolbox.spectrum import SubtractSpectrum

Description

Task for subtracting a scalar from the flux data included in a spectrum container or for subtracting two spectrum containers from each other on a spectrum-by-spectrum basis.

For the scalar mode, 'ds' and 'param' need to specified - for the pair-wise mode, 'ds1' and 'ds2' need to be set. In the pairwise mode, the task iterates through the two containers and computes the pair-operation. In case the size of the two containers is different, the iteration stops at the minimum size. Hence, the remaining spectra in the larger container are ignored. In case one of the datasets only contains a single spectrum, the task always refers to this single spectrum while iterating over all the spectra in the other dataset.

Different selection schemes are available for selecting the point spectra or the segments to be processed. The most simple scheme is to specify lists of point spectrum indices (`selection=[1,3,4,2]`). In this case, the operation (scalar- or pair-operation) is just taken over the point spectra with corresponding indices. Instead of specifying a list of indices (which requires the knowledge of these) other recipes to specify a selection are available. Typical recipes test attribute values to match certain values or to be included in a given interval. These recipes we refer to as selection models. The different options to specify selections can be combined. Here, an AND logic is adopted.

Similarly, you can specify what segments to consider. In the most simple case, you can just specify the indices of segments to be considered (for the scalar operation `segments=[1,3,4]` and for the pair operation `segments1=[1,3,4], segments1=[3,6,5]`). In more advanced situations you can use a `SegmentSelection` object. Note that the pairs of segments to be processed need to be consistent - otherwise the processing will fail.

You can specify whether the orginal datasets should be overwritten with the 'overwrite' flag. By default no data is overwritten. In the pair-wise mode, it is not always possible to overwrite the data - in particular if non-trivial segment selections are specified.

For the pair-wise mode, there are further options that can be selected by the 'variant' (e.g. whether to use flags and weights).

Example

Example 1: from Jide:

```
from herschel.ia.toolbox.spectrum import SubtractSpectrum
subtract = SubtractSpectrum()
subtractConstant = subtract(ds=spectra, param=2.1)
subtractedFactor = subtract(ds=spectra, selection={"bbtype":[6031]}, param=2.1)
subtractedFactor = subtract(ds=spectra, selection=[0,1,2,3], param=2.1)
subtractedFactor = subtract(ds=spectra, selection={"Chopper":([-4.4,5.9],0.2),
 "bbtype":[6031]}, param=2.1)
subtractedFactor = subtract(ds=spectra, selection={"LoFrequency":
(4000.0,5000.0)}, param=2.1)
subtractedPairWise = subtract(ds1=spectra1, ds2=spectra2)
subtractedPairWise = subtract(ds1=spectra1, ds2=spectra2, selection={"bbtype":
[6031]})
subtractedPairWise = subtract(ds1=spectra1, ds2=spectra2, selection=[0,1,2,3])
subtractedPairWise = subtract(ds1=spectra1, ds2=spectra2, selection={"Chopper":
([-4.4,5.9],0.2), "bbtype":[6031]})
```

Example 1: from Jide:

```
subtractedPairWise = subtract(ds1=spectra1, ds2=spectra2,
 selection={"LoFrequency":(4000.0,5000.0)})
```

API Summary

Properties
SpectrumContainer **ds** [INPUT, OPTIONAL, default=no default value.]
Double **param** [INPUT, OPTIONAL, default=no default value.]
Object **selection** [INPUT, OPTIONAL, default=None.]
PyDictionary|Map> **selection_lookup** [INPUT, OPTIONAL, default=no default value.]
PyList **selection_index** [INPUT, OPTIONAL, default=No default value.]
Boolean **overwrite** [INPUT, OPTIONAL, default=False.]
SpectrumContainer **ds1** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **ds2** [INPUT, OPTIONAL, default=no default value.]
Object **segments** [INPUT, OPTIONAL, default=no default value.]
Object **segments1** [INPUT, OPTIONAL, default=no default value.]
Object **segments2** [INPUT, OPTIONAL, default=no default value.]
String **variant** [INPUT, OPTIONAL, default=no default value.]
SpectrumContainer **result** [OUTPUT, OPTIONAL, default=no default value]

API details

Properties

SpectrumContainer ds [INPUT, OPTIONAL, default=no default value.]

Input container to be processed by the task in case the task is used as a scalar operation.

Double param [INPUT, OPTIONAL, default=no default value.]

The scalar parameter to be considered when the task is used as scalar operation.

Object selection [INPUT, OPTIONAL, default=None.]

Specify what point spectra the operation should be applied to. Different ways to specify these selections are possible:

- Specify a list of indices (in jython) of the point spectra for which the add should be applied.
- Use a dictionary (in jython) to specify a selection model (lookup for attributes, filter attributes to be included in ranges or intervals).
- Pass a string representation of a jython list or a jython dictionary. The string will be parsed and the prcessing delegated to the two cases above.
- Pass any java instance that implements the SelectionModel interface (for the advanced user).

PyDictionary|Map> selection_lookup [INPUT, OPTIONAL, default=no default value.]

Specify a PyDictionary with keys given by the attribute name to look up and a list of admissible values. Behind the scenes, a DiscreteValueSelectionModel is instantiated.

PyList selection_index [INPUT, OPTIONAL, default=No default value.]

Specify a PyList with the indices of the point spectra to be considered.

Boolean overwrite [INPUT, OPTIONAL, default=False.]

Specify whether the input data container can be reused - the values found therein is overwritten.

SpectrumContainer ds1 [INPUT, OPTIONAL, default=no default value.]

First input container for pair-wise operations.

SpectrumContainer ds2 [INPUT, OPTIONAL, default=no default value.]

Second input container for pair-wise operations.

Object segments [INPUT, OPTIONAL, default=no default value.]

Specify what segments the operation should be applied to. There are two options available:

- Either pass an instance of a SegmentSelection which gives the information on what segments for each point spectrum included in the container.
- Specify a PyList of segment indices.

Object segments1 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds1'.

Object segments2 [INPUT, OPTIONAL, default=no default value.]

Specify the segment selection to be associated with 'ds2'.

String variant [INPUT, OPTIONAL, default=no default value.]

Specify the variant of processing mode you would like to run (including flags / weights / ...). HCSS defaults include: "flux" / "flux-wave" / "flux-wave-weight" / "flux-flag-wave" / "flux-weight-flag-wave"

SpectrumContainer result [OUTPUT, OPTIONAL, default=no default value]

Result object containing the results of the operation applied.

See also

- SpectrumTask

History

- `2007-08-17 - meli`: initial.

- `2008-03-16 - meli`: major refactoring of the toolbox.
- `2008-10-03 - meli`: segments, javadoc

3.328. SUM

Full Name:	herschel.ia.numeric.toolbox.basic.Sum
Alias:	SUM
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Sum

Description

Yields the sum of all elements in the input array.

Returns a scalar of the same type as the type of the input array.

Example

Example 1: Apply SUM on a Int2d

```
x=Int2d( [ [1,2], [-1,3] ])
print SUM(x)    # 5
print SUM(x, 0) # [0,5]
print SUM(x, 1) # [3,2]
```

API Summary

Jython Syntax

```
<y>=SUM(<x>, [,<dim>])
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
integer dim [INPUT, MANDATORY, default=no default value]
float or double array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array can be any scalar array.

integer dim [INPUT, MANDATORY, default=no default value]

The dimension to compute the calculation along

float or double array y [OUTPUT, MANDATORY, default=no default value]

Returns a scalar of the same type as the type of the input array.

See also

- PRODUCT

3.329. TablePlotter

Full Name:	herschel.ia.gui.explorer.table.TablePlotter
Type:	Java Class -
Import:	from herschel.ia.gui.explorer.table import TablePlotter

Description

TablePlotter is a GUI tool to view and analyze TableDataset. It can be invoked in JIDE and HIPE by right clicking

on a TableDataset. It can also be invoked by a Jython command line in JIDE or Hipe through jython script. By default, the second column is plotted as y data and first column as x data unless x and y index are specified. All the default can be changed vs a group of buttons and selectors.

Example

Example 1: Invoke TablePlotter from command line

```
TablePlotter is a pluggable component which can be plugged in any GUI
 containers.
Example:  This example shows how to use TablePlotter in command line
<pre>
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.gui.explorer.table import TablePlotter
from herschel.ia.gui.explorer.table import OverPlotter
from herschel.ia.gui.explorer.table import *
from herschel.share.component import *
################################################
# Load your data
################################################
fits=FitsArchive();
path="/home/zhang/Development/data/"
filename = "loadCurve.fits"
try:
     fits.reader = FitsArchive.HCSS_READER
     p = fits.load(path+filename)
 except java.io.IOException:
    fits.reader = FitsArchive.STANDARD_READER
     p = fits.load(path+filename)
 ################################################
 # Load TablePlotter with a TableDataset only
 ################################################
 tbs =p.default
 tpl=TablePlotter(tbs)
 wm=WindowManager.getDefault()
 #Load TablePlotter
 wm.addWindow('test', tpl.component, 1)
 #get extracted table
 extractedTable = tpl.activeLayerStruct.extractedTableDataset
 #get flags
 flags = tpl.activeLayerStruct.flags
 print flags.size
 print flags.dimensions
 ################################################
 # Load a TablePlotter with a 2d flags
 ################################################
 tpl1=TablePlotter(tbs, flags)
 wm=WindowManager.getDefault()
 print tbs.rowCount
 #Load TablePlotter
```

Example 1: Invoke TablePlotter from command line

```
wm.addWindow('test 2d fkags', tpl1.component, 1)
################################################
# Load TablePlotter with a 1d flag
################################################
flag=flags.get(Range(0, tbs.rowCount), 1)
tpl2=TablePlotter(tbs, flag)
wm=WindowManager.getDefault()
print tbs.rowCount
#Load TablePlotter
wm.addWindow('test 1d flags', tpl2.component, 1)
flags1d = tpl2.activeLayerStruct.flags
print flags1d.size
</pre>
```

API Summary

Constructors
TablePlotter() A default constructor
TablePlotter(TableDataset tds) A constructor
TablePlotter(TableDataset tds, Bool2d flags) A constructor
TablePlotter(TableDataset tds, integer xIndex, integer yIndex) A constructor
TablePlotter(TableDataset tds, Bool1d flags, integer xIndex, int yIndex) A constructor
TablePlotter(TablePlotter tds, Bool1d flags) This constructor will initiate an instance of TablePlotter with pre selected and de-selected columns.

Methods
JComponent **getComponent**() Inherit from explorer
String **getDescription**()
String **getName**()
setObject(Object data)
LayerStruct **getActiveLayerStruct**() This is a command line API. It is a getter to get the active LayerStruct object.

Limitations

The TableDataset has to be one dimensional numeric array such as Double1d, Float1d, Long1d, Short1d and Complex1d.

Miscellaneous

All the examples here are given in Jython script

API details

Constructors

TablePlotter()

This is a default constructor which will be called in DatasetInspector It initializes the DataObjectLinsterSupport for data extraction.

TablePlotter(TableDataset tds)

This constroctor will initialize and create an instance of TablePlotter for the input dataset.

Argument

`TableDataset` **`tds`** `[INPUT, MANDATORY, default=no default value]`

A TableDataset to be plotted. It contains one or more columns.

Example

Create a TablePlotter instance with a tds dataset

```
from hersechl.ia.gui.explorer.table import TablePlotter  #import
 TablePlotter
tpl=TablePlotter(tds)  #tds is a dataset defined somewhere else
```

TablePlotter(TableDataset tds, Bool2d flags)

This constructor will initiate an instance of TablePlotter with two inputs, a TableDataset and its flags. The flags tell which data points are selected and de-selected. TablePlotter will display two layer plots and one of which is for selected data (blue color) and the other is for de-selected data (red color).

Arguments

`TableDataset` **`tds`** `[INPUT, MANDATORY, default=no default value]`

A TableDataset to be plotted and viewed. It contains one or more columns.

`Bool2d` **`flags`** `[INPUT, MANDATORY, default=no default value]`

A flags to indicate which data points are selected and de-selected.

Example

create a TablePlotter instance with a tds dataset and flags

```
<pre>
 from hersechl.ia.gui.explorer.table import TablePlotter #import
 TablePlotter
 tpl=TablePlotter(tds, flags) #tds and flags are defined somewhere else
</pre>
```

TablePlotter(TableDataset tds, integer xIndex, integer yIndex)

This constructor will initiate an instance of an TablePlotter with three input, a table, the xIndex and the yIndex.

Arguments

`TableDataset` **`tds`** `[INPUT, MANDATORY, default=no default value]`

A TableDataset to be plotted and viewed. It contains one or more columns.

`integer` **`xIndex`** `[INPUT, MANDATORY, default=no default value]`

The xIndex is an integer which indicates the x column data

`integer` **`yIndex`** `[INPUT, MANDATORY, default=no default value]`

The yIndex is an integer which indicates the y column data

TablePlotter(TableDataset tds, integer xIndex, integer yIndex)

Example

create an instance of TablePlotter where column 3 is x and column 10 is y

```
from hersechl.ia.gui.explorer.table import TablePlotter #import the
 TablePlotter
#plot the 10th column vs the first column from tds table
tablePlotter = TablePlotter (tds, 0, 10)  #tds is a dataset defined somwhere
 else
```

TablePlotter(TableDataset tds, Bool1d flags, integer xIndex, int yIndex)

This constructor will initiate an instance of TablePlotter with specified x and y data.

Arguments

`TableDataset` **`tds`** `[INPUT, MANDATORY, default=no default value]`

A TableDataset to plot and view. The tds contains one or more columns.

`Bool1d` **`flags`** `[INPUT, MANDATORY, default=no default value]`

The flags specify which columns are de-selected and selected

`integer` **`xIndex`** `[INPUT, MANDATORY, default=no default value]`

The xIndex specifies the x axis data

`int` **`yIndex`** `[INPUT, MANDATORY, default=no default value]`

The yIndex speicies the y axis data

Example

create an TablePlotter object with tds, flags and x and y column indices

```
from hersechl.ia.gui.explorer.table import TablePlotter  #import the
 TablePlotter
#Plot the fourth column vs the second column
tpl = TablePlotter(tds, flags, 1, 3) #tds, flags are defined somewhere else
```

TablePlotter(TablePlotter tds, Bool1d flags)

The Bool1d flag indicate which columns are selected and which columns are de-selected.

Arguments

`TablePlotter` **`tds`** `[INPUT, MANDATORY, default=no default value]`

The tds is the TableDataset to be plotted. It contains one or more columns.

`Bool1d` **`flags`** `[INPUT, MANDATORY, default=no default value]`

The flags speicifies which columns are de-selected and which columns are selected

Methods

JComponent getComponent()

This method will return a TablePlotter as a component so that users can plug it in his/her own applications.

Return

JComponent

- return the TablePlotter as a component

Examples

Use TablePlotter as a plug-in

JComponent getComponent()

```
<pre>
from herschel.share.component import *
from javax.swing import *
from java.awt import *
from herschel.ia.io.fits import FitsArchive
from herschel.ia.dataset import TableDataset
from herschel.ia.dataset import Product
from herschel.ia.dataset.gui import *
from herschel.ia.gui.explorer.table import TablePlotter
fits=FitsArchive();
#Change to your data path
path="/home/zhang/Development/data/"
filename = "loadCurve.fits"
#load the table
try:
  fits.reader = FitsArchive.HCSS_READER
  p = fits.load(path+filename)
except java.io.IOException:
  fits.reader = FitsArchive.STANDARD_READER
   p = fits.load(path+filename)
#create a TableDataset
table =p.default
#create a container to hold the controls/components
pane=JPanel()
#set the layout, there are many different kinds users can choose according
 to his/her need
pane.setLayout(BoxLayout(pane, BoxLayout.Y_AXIS))
#add control one, a lael
title = JLabel("Display TablePlotter")
pane.add(title)
#Add TablePlotter to the pane
tablePlotter=TablePlotter(table)
pane.add(tablePlotter.component)
pane.setPreferredSize(Dimension(tablePlotter.component.width,
 tablePlotter.component.height))
#add to your application
frame =JFrame("TablePlotter as Plug-in demo");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
frame.getContentPane().add(pane, BorderLayout.CENTER)
frame.pack
frame.visible=1
</pre>
```

- example 2

```
from herschel.ia.gui.explorer.table import
from herschel.share.component import
wm=WindowManager.getDefault()
tpl = TablePlotter(tds)
wm.addWindow('My Application', tpl.component, 1)
```

String getDescription()

Return

String

- the description of this class

String getName()

Return

String

- the nam of the application

setObject(Object data)

Argument

`Object` **`data`** `[INPUT, MANDATORY, default=no default value]`

If data is a TableDataset, it will be processed and plotted. If data is not a TableDataset, nothing will be displayed.

Example

Pass the dataset to TablePlotter instance

```
<pre>
tpl = TablePlotter()
tpl.object = tds #tds defined somewhere else
</pre>
```

LayerStruct getActiveLayerStruct()

Through the active LayerStruct object, extracted table and flags can be retreived.

Return

LayerStruct

- return the current active LayerStruct

Example

Get the extracted dataset and flags

```
tpl = TablePlotter(tds) #tds defined somewhere else
extracteDataset = tpl.activeLayerStruct.extractedTableDataset
flags = tpl.activeLayerStruct.flgs
```

History

- `2006-10-06 - first`: version of TablePlotter
- `2007-12-21 - Major`: GUI changes

3.330. TAN

Full Name:	herschel.ia.numeric.toolbox.basic.Tan
Alias:	TAN
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Tan

Description

Computes the trigonometric tangent of an number or array

Gives the tangent of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

Example

Example 1: Apply TAN on a Float1d

```
x=Float1d([0,0.5])
print  TAN(x) # [0.0,0.5463025]
```

API Summary

Jython Syntax
`<y>=TAN(<x>)`

Properties
`any type x [INPUT, MANDATORY, default=no default value]`
`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

API details

Properties

`any type x [INPUT, MANDATORY, default=no default value]`

The input is in radians, and may be of any type.

`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- ARCTAN
- COS
- SINH
- TANH

3.331. TANH

Full Name:	herschel.ia.numeric.toolbox.basic.TanH
Alias:	TANH
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import TanH

Description

Computes the trigonometric hyperbolic tangent of an number or array

Gives the hyperbolic tangent of a number or array. The input is in radians, and may be of any type. Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

API Summary

Jython Syntax
`<y>=TANH(<x>)`

Properties
`any type x [INPUT, MANDATORY, default=no default value]`
`any type y [OUTPUT, NOT_MANDATORY, default=no default value]`

Miscellaneous

Does not work for complex values.

API details

Properties

any type x [INPUT, MANDATORY, default=no default value]
The input is in radians, and may be of any type.

any type y [OUTPUT, NOT_MANDATORY, default=no default value]
Returns a float array for float input array, complex array for input complex array, and double array for any other numeric array type.

See also

- TAN
- ARCTAN

3.332. TaskWrapper

Full Name:	herschel.ia.dataflow.TaskWrapper
Type:	Java Class -
Import:	from herschel.ia.dataflow import TaskWrapper

Description

Wraps a task into a process.

TaskWrapper is a special process which allows reuse of a Task instance within stream type environment. It automatically creates an connector for each parameter of the Task instance (as passed as an arg for the constructor of this class).

In default mode (= state mode) it will enable the user to build up an input array (if present) for each Task parameter. In practice it means it creates (a) an input connector which deals with instances of the array's component type and (b) a control connector allowing the user to select the number (default = 1) of instances to gather in an ArrayList before these can be passed to the tasks input parameter as an array.

I.e. for the input parameters of the type [] this class allows you to maintain a state: in case fifo = true (default) the array(list) as maintained by this object is using the fifo concept for the case the array size = "number as set by the related control". Otherwise (i.e fifo=false) the array will be emptied after the number (as set by the control) is reached. Everytime the array is filled it is passed to the task's parameter, the task's "perform()" method is called and the tasks output(s) are passed to the related output(s) of this TaskWrapper object.

Example

Example 1: how to use a TaskWrapper with AverageTask

```
<pre>
from herschel.ia.dataflow import *
from mytasks import AverageTask
tw = TaskWrapper("avg", AverageTask(), True, True, True)
df = DataFlow("mydf")
df.addProcess(tw)
# The rest is obvious...
</pre>
```

Limitations

no limitation

See also

- reference

History

- 26-5-2005 JCG: change javadoc format for help support.

3.333. ThreadDynamicProcessImpl

Full Name:	herschel.ia.dataflow.ThreadDynamicProcessImpl
Type:	Java Class -
Import:	from herschel.ia.dataflow import ThreadDynamicProcessImpl

Description

Provides the thread-based model for implementing processes.

Class to be extended. It provides thread-based model. The only method that in general should be overrriden is run method. The most interested methods are start, stop and run. The run method has always to be overriden, since it is where the execution of the process is done. As this class is handling a Thread, run method has to follow some rules. The typical run method should be like:

```
public void run() {
while( isRunning() ) {
try {
input = _input.getData();
output =doJob(input);
_output.dataReady(output);
} catch( InterruptedException ex ) {
break;
} catch( Exception ex2 ) {
handlingException(ex2);
}
}
```

The way a thread is stopped is simply letting it to return from the run method. isRunning() will stop the thread, but sometimes in the run method is stucked, waiting in a blocked method (as _input.getData() in the example above). So, the stop method not only makes that isRunning() will return false, but it will try to interrupt the thread just in case. That means that it is important to let run() method to be finished in case an InterruptedException is catched. If not, probably the thread will be eating all the CPU in an infinite loop. For a simpler thread-based super class, see herschel.ia.dataflow.template.ThreadDynamicProcessImplTemplate

IMPORTANT NOTE: This class provides two methods start/stop. These two methods are managing an internal thread. That means that when `start` is called, it will launch a thread, that may be stopped with `stop`. So, if you are using your own thread, redefine `start/stop` methods but do not call `super.start()/super.stop()` because two threads will be launched (unless this is really what you want to do).

Example

Example 1: how to implement a ThreadDynamicProcessImpl

```
<pre>
import  herschel.ia.dataflow.*;
public class ProcessFFT extends ThreadDynamicProcessImpl {
   private final ProcessInput _input;
   private final ProcessOutpu _output;
  public ProcessFFT(String name) {
      super(name);
      _input = createInput("input", Product.class);
      _output = createOutput("output", Product.class);
 }
 public void run() {
      while( isRunning() ) {
          try {
             Product product = (Product)_input.nextData();
```

Example 1: how to implement a ThreadDynamicProcessImpl

```
            // a doStuff method should be defined and would do the
 processing.
            Product new_product = doStuff(product);
            _output.dataReady( new_product );
         } catch ( InterruptedException ex ) {
             System.out.println("Interrupted exception thrown:" + ex+ ".
 Exiting thread");
             break;
         }
     } // while
  }
}
</pre>
```

Limitations

no limitation

See also

- reference

History

- 26-5-2005 jcg: change javadoc format for help support.

3.334. tiledImage

Full Name:	herschel.ia.toolbox.image.TiledImageTask
Alias:	tiledImage
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import TiledImageTask
Category:	task/image

Description

A Task to convert an image to a TiledImage.

A Task to convert an image to a TiledImage, taking the flag into account. TiledImageTask is a task which returns the TiledImage of an Image.

Examples

Example 1: Return the TiledImage, taking into account the flag.

```
tiledImage(image = im, flag = True)
```

Example 2: Return the TiledImage, not taking into account the flag.

```
tiledImage(image = im, method=CutLevels.MEDIAN_FILTER)
```

API Summary

Jython Syntax

```
TiledImage(image, True)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
boolean flag [INPUT, MANDATORY, default=true]
TiledImage result [OUTPUT, MANDATORY, default=No default value]
```

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The image to use for returning the TiledImage.

`boolean flag [INPUT, MANDATORY, default=true]`

True if the flag should be taken into account.

`TiledImage result [OUTPUT, MANDATORY, default=No default value]`

The final TiledImage.

3.335. Transform2dTask

Full Name:	herschel.ia.toolbox.image.Transform2dTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import Transform2dTask

Description

An abstract Task for two dimensional transforms.

An abstract Task that calculates the Fourier Transform and the power spectrum.

API Summary

Properties
`Numeric2dData image [INPUT, MANDATORY, default=No default value]`
`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`
`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Numeric2dData image [INPUT, MANDATORY, default=No default value]`

The input image.

`Complex2d transform [OUTPUT, MANDATORY, default=No default value]`

The transform.

`Double2d spectrum [OUTPUT, MANDATORY, default=No default value]`

The spectrum.

3.336. translate

Full Name:	herschel.ia.toolbox.image.TranslateTask
Alias:	translate
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import TranslateTask
Category:	task/image

Description

The Translate task for Images.

Translate is a task which translates an Image. The Wcs is also adapted. The image is translated over the given parameters (if ra = 1h00m00s, the image will be moved 1h00m00s to the right. At the left, 0.0's will be added which will be masked out.)

Examples

Example 1: translating an image using image coordinates

```
translate(image = im, x = 5, y = 7)
```

Example 2: translating an image using sky coordinates

```
translate(image = im, ra = 0.03, dec = 0.03)
```

API Summary

Jython Syntax

```
translate()
translate(image, ra, dec)
translate(image, x, y)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
double x [INPUT, OPTIONAL, default=0.0]
double y [INPUT, OPTIONAL, default=0.0]
Angle ra [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]
Angle dec [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]
Image translatedImage [OUTPUT, MANDATORY, default=No default
value]
```

API details

Properties

`Image image [INPUT, MANDATORY, default=No default value]`

The Image to translate

double x [INPUT, OPTIONAL, default=0.0]
The number of pixels to translate in x-direction

double y [INPUT, OPTIONAL, default=0.0]
The number of pixels to translate in y-direction

Angle ra [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]
The amount of degrees to move in right ascension

Angle dec [INPUT, OPTIONAL, default=Angle(0.0, ANGLE.DEGREES)]
The amount of degrees to move in declination

Image translatedImage [OUTPUT, MANDATORY, default=No default value]
Result of the translation of the Image

See also

- ???

3.337. TRANSPOSE

Full Name:	herschel.ia.numeric.toolbox.matrix.MatrixTranspose
Alias:	TRANSPOSE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.matrix import MatrixTranspose

Description

Transposes a matrix.

Example

Example 1: Transposing a 2D array

```
x=Int2d([ [1,2],[3,4],[5,6] ])
print TRANSPOSE(x) # [ [1,3,5],[2,4,6] ]
```

API Summary

Jython Syntax

```
<y>=TRANSPOSE(<x>)
```

Property

```
Array2dData x [INPUT, MANDATORY, default=no default value]
```

Miscellaneous

TRANSPOSE is an alias for MatrixTranspose.FUNCTION

API details

Property

`Array2dData x [INPUT, MANDATORY, default=no default value]`

Input must be a 2d array as defined by the numeric library.

3.338. transpose

Full Name:	herschel.ia.toolbox.image.TransposeTask
Alias:	transpose
Type:	Java Task -
Import:	from herschel.ia.toolbox.image import TransposeTask
Category:	task/image

Description

The Transpose task for Images.

TransposeTask is a task which transposes an Image. The Wcs is also adapted. The following types of are possible :

- FLIP_VERTICAL : The image is flipped upside-down
- FLIP_HORIZONTAL : The image is flipped left-right
- FLIP_DIAGONAL : The image is mirrored around the diagonal
- FLIP_ANTIDIAGONAL : The image is mirrored around the antidiagonal
- ROTATE_90 : The image is rotated 90 degrees
- ROTATE_180 : The image is rotated 180 degrees
- ROTATE_270 : The mirror is rotated 270 degrees

Examples

Example 1: Flipping an image horizontal

```
transpose(image = im, type = TransposeTask.FLIP_HORIZONTAL)
```

Example 2: Flipping an image antidiagonal

```
transpose(im, TransposeTask.FLIP_ANTIDIAGONAL)
```

API Summary

Jython Syntax

```
transpose()
transpose(image, TransposeTask.FLIP_VERTICAL)
```

Properties

```
Image image [INPUT, MANDATORY, default=No default value]
TransposeType type [INPUT, MANDATORY,
default=Transpose.FLIP_VERTICAL]
Image transposedImage [OUTPUT, MANDATORY, default=No default
value]
```

API details

Properties

Image image [INPUT, MANDATORY, default=No default value]

The Image to transpose

TransposeType type [INPUT, MANDATORY, default=Transpose.FLIP_VERTICAL]

The type of transposition to apply (FLIP_VERTICAL, FLIP_HORIZONTAL, FLIP_DIAGONAL, FLIP_ANTIDIAGONAL, ROTATE_90, ROTATE_180 or ROTATE_270)

Image transposedImage [OUTPUT, MANDATORY, default=No default value]

Result of the transposition of the Image

3.339. UNIQ_SORTED

Full Name:	herschel.ia.numeric.toolbox.basic.UniqSorted
Alias:	UNIQ_SORTED
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import UniqSorted

Description

Returns the unique elements of an array.

An index uniq is available as a special case of UNIQ_SORTED i.e. UNIQ_SORTED.BY_INDEX which returns an index array into the original array. UNIQ_SORTED.COUNT_UNIQ_VALUES returns the number of unique values in an array; although it works also with unsorted arrays, passing an already sorted array is much more efficient.

Examples

Example 1: Apply UNIQ_SORTED on a Double1d

```
x=Double1d([-1,4,2,7,2,-6,-8,3,2])
print UNIQ_SORTED(SORT(x))              # [-8.0,-6.0,-1.0,2.0,3.0,4.0,7.0]
```

Example 2: Apply UNIQ_SORTED.BY_INDEX

```
x=SORT(Double1d([-1,4,2,7,2,-6,-8,3,2]))
print UNIQ_SORTED.BY_INDEX(x)     # [0,1,2,3,6,7,8]
```

Example 3: Apply UNIQ_SORTED.COUNT_UNIQ_VALUES

```
x=Double1d([4, 5, 7, 4, 8, 8, 3, 1, 5])
print UNIQ_SORTED.COUNT_UNIQ_VALUES(SORT(x))  # 6
```

API Summary

Jython Syntax

```
<y>=UNIQ_SORTED(<x>)
```

Properties

any sorted array **x** [INPUT, MANDATORY, default=no default value]

an array **y** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

any sorted array x [INPUT, MANDATORY, default=no default value]

The input array can only be an array of rank 1

an array y [OUTPUT, MANDATORY, default=no default value]

Returns an array containing only the unique elements from the input array.

3.340. UNIQ

Full Name:	herschel.ia.numeric.toolbox.basic.Uniq
Alias:	UNIQ
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Uniq

Description

Returns the unique elements of an array.

An index uniq is available as a special case of UNIQ i.e. UNIQ.BY_INDEX which returns an index array into the original array. UNIQ.COUNT_UNIQ_VALUES returns the number of unique values in an array; although it works also with unsorted arrays, passing an already sorted array is much more efficient. The UNIQ.IS_UNIQ function, which tests if the given array indeed contains unique elements, is deprecated. It is strongly recomended that you sort the array and use UNIQ_SORTED due to performance issues.

Examples

Example 1: Apply UNIQ on a Double1d

```
x=Double1d([-1,4,2,7,2,-6,-8,3,2])
print UNIQ(x)                    # [-8.0,-6.0,-1.0,2.0,3.0,4.0,7.0]
print UNIQ(SORT(x))              # [-8.0,-6.0,-1.0,2.0,3.0,4.0,7.0]
```

Example 2: Apply UNIQ.BY_INDEX

```
x=Double1d([-1,4,2,7,2,-6,-8,3,2])
print UNIQ.BY_INDEX(x)         # [0,1,2,3,5,6,7]
print UNIQ.BY_INDEX(SORT(x))   # [0,1,2,3,6,7,8]  SORT(x) =
 [-8.0,-6.0,-1.0,2.0,2.0,2.0,3.0,4.0,7.0]
```

Example 3: Apply UNIQ.COUNT_UNIQ_VALUES

```
x=Double1d([4, 5, 7, 4, 8, 8, 3, 1, 5])
print UNIQ.COUNT_UNIQ_VALUES(x)  # 6
```

Example 4: Apply UNIQ.IS_UNIQ

```
x=Double1d([-1,4,7,-2,-6,-8,3,2])
print UNIQ.IS_UNIQ(x)        # 0 (false) - Note that this function is
 deprecated
print UNIQ.IS_UNIQ(SORT(x))  # 1 (true) - Note that this function is deprecated
```

API Summary

Jython Syntax

```
<y>=UNIQ(<x>)
```

Properties

```
any sorted/unsorted array x [INPUT, MANDATORY, default=no default
value]
```

```
an array y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any sorted/unsorted array x [INPUT, MANDATORY, default=no default value]
The input array can only be an array of rank 1

an array y [OUTPUT, MANDATORY, default=no default value]
Returns an array containing only the unique elements from the input array.

3.341. VARIANCE

Full Name:	herschel.ia.numeric.toolbox.basic.Variance
Alias:	VARIANCE
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import Variance

Description

Yields the variance value of the elements in the input array.

The standard deviation is equivalent to SQRT(VARIANCE(x)).

Example

Example 1: Apply VARIANCE on a Float1d

```
x=Float1d([1,3,2,3,4])
print VARIANCE(x) # 1.3
```

API Summary

Jython Syntax

```
<y>=VARIANCE(<x>)
```

Properties

```
any scalar array x [INPUT, MANDATORY, default=no default value]
double y [OUTPUT, MANDATORY, default=no default value]
```

API details

Properties

any scalar array x [INPUT, MANDATORY, default=no default value]

The input array integral or floating-point arrays; the former implicitly transformed to a double array.

double y [OUTPUT, MANDATORY, default=no default value]

Returns a double

See also

- MEAN
- MEDIAN
- STDDEV

3.342. VelocityPosMapComputeTask

Full Name:	herschel.ia.toolbox.cube.VelocityPosMapComputeTask
Type:	Java Task -
Import:	from herschel.ia.toolbox.cube import VelocityPosMapComputeTask

Description

VelocityPosMapComputeTask The task to compute Velocity position map from a Simplecube

API Summary

Properties
`simpleCube` **`simplecube`** `[INPUT, MANDATORY, default=no default value]`
`axis` **`unknown`** `[INPUT, Boolean, default=MANDATORY]`
`coordSlitArray` **`unknown`** `[INPUT, Double2d, default=no default value]`
`widthSlit` **`unknown`** `[INPUT, Integer, default=no default value]`
`referenceLayer` **`unknown`** `[INPUT, Integer, default=no default value]`
`totalWeight` **`unknown`** `[INPUT, no default value, default=no default value]`
`velocityMap` **`unknown`** `[OUTPUT, Double3d, default=no default value]`
`velocityMapAxis` **`unknown`** `[OUTPUT, Double2d, default=no default value]`
`velocityMapAxisProd` **`unknown`** `[OUTPUT, SimpleImage, default=no default value]`
`cubeVelocityMap` **`unknown`** `[OUTPUT, SimpleCube, default=no default value]`

API details

Properties

`simpleCube simplecube [INPUT, MANDATORY, default=no default value]`

The spectralCube as a SimpleCube containing the spectral and velocities information

`axis unknown [INPUT, Boolean, default=MANDATORY]`

Define the type of map to generate axis coordSlitArray widthSlit referenceLayer totalWeight

`coordSlitArray unknown [INPUT, Double2d, default=no default value]`

new version array of pixel to be read : manage the width of the slit the order of the information is index, X, Y , weight

`widthSlit unknown [INPUT, Integer, default=no default value]`

width of the slit

referenceLayer unknown [INPUT, Integer, default=no default value]

The layer considered as reference for the computation of the velocities

totalWeight unknown [INPUT, no default value, default=no default value]

The inside total weight of the array of pixel to be read

velocityMap unknown [OUTPUT, Double3d, default=no default value]

The 2D map of velocity

velocityMapAxis unknown [OUTPUT, Double2d, default=no default value]

The map of velocity along the axis to come soon:

velocityMapAxisProd unknown [OUTPUT, SimpleImage, default=no default value]

The map of velocity along the axis, stored in a SimpleCube

cubeVelocityMap unknown [OUTPUT, SimpleCube, default=no default value]

a cube containing the map of velocity at maximmum of emission, the dispersion map, an error value ...

3.343. VelocityPosMapPlotting

Full Name:	herschel.ia.gui.cube.VelocityPosMapPlotting
Type:	Java Class -
Import:	from herschel.ia.gui.cube import VelocityPosMapPlotting

Description

PositionVelocityDiagramPlotting.

A class to deal with PositionVelocityDiagramComputeTask.

API Summary

Constructors
`VelocityPosMapPlotting(type toolbox)`
`VelocityPosMapPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)` Constructor for VelocityPosMapPlotting.

Methods
`setBegin(Double begin)` Sets the begin of the straight line, associated with this
`setEnd(Double end)` Sets the end of the straight line, associated with this
`ImageFigure getLine()` Returns the straight line (ImageFigure), along which we wish to
`Double getBegin()` Returns the begin of the drawn line in PixelCoordinates.
`Double getEnd()` Returns the end of the drawn line in PixelCoordinates.
`Double1d IntensityVelocityPosMapPlotting()` Returns the intensity of the pixels along the straight line,
`Point2D[] getPlotPoints()` Returns the intensity of the pixels along the straight line,
`void getCoordPoints()` Returns the intensity of the pixels along the straight line,
`String getSkyCoordinates()` Returns the String version of the sky coordinates

API details

Constructors

`VelocityPosMapPlotting(type toolbox)`
Argument

VelocityPosMapPlotting(type toolbox)

`type` **toolbox** `[INPUT, MANDATORY, default=no default value]`

The CubeSpectrumAnalysisToolbox, for which you want to extract a velocity position map

VelocityPosMapPlotting(boolean useAsComponent, CubeSpectrumAnalysisToolbox toolbox)

When the new VelocityPosMapPlotting is component-based, a window for plotting the histogram is opened; otherwise nothing will be shown. jparameter useAsComponent Indication whether the new VelocityPosMapPlotting should be component-based.

Arguments

`boolean` **useAsComponent** `[INPUT, MANDATORY]`

`CubeSpectrumAnalysisToolbox` **toolbox** `[INPUT, MANDATORY]`

Methods

setBegin(Double begin)

Velocity position map to the given point (in UserCoordinates).

Argument

`Double` **begin** `[INPUT, MANDATORY]`

setEnd(Double end)

Velocity position map to the given point (in UserCoordinates).

Argument

`Double` **end** `[INPUT, MANDATORY]`

ImageFigure getLine()

create the diagram velocity position based on the Bresenham's Line-Drawing Algorithm

Return

ImageFigure

The straight line (ImageFigure), along which we wish to create the diagram velocity position based on the Bresenham's Line-Drawing Algorithm

Double getBegin()

Return

Double

Returns the begin of the drawn line in PixelCoordinates.

Double getEnd()

Return

Double

Returns the end of the drawn line in PixelCoordinates.

Double1d IntensityVelocityPosMapPlotting()

associated with the given Velocity position map.

Double1d IntensityVelocityPosMapPlotting()

Return

Double1d

Returns the intensity of the pixels along the drawn straight line, associated with the given Velocity position map.

Point2D[] getPlotPoints()

associated with the given Velocity position map.

Return

Point2D[]

Returns the intensity of the pixels along the drawn straight line, associated with the given Velocity position map.

void getCoordPoints()

associated with the given Velocity position map.

Return

void

Returns the intensity of the pixels along the drawn straight line, associated with the given Velocity position map.

String getSkyCoordinates()

(WorldCoordinates) of the begin and end of the drawn straight line.

Return

String

Returns the String version of the sky coordinates (WorldCoordinates) of the begin and end of the drawn straight line.

3.344. WcsExplorer

Full Name:	herschel.ia.gui.image.WcsExplorer
Type:	Java Class -
Import:	from herschel.ia.gui.image import WcsExplorer

Description

An explorer to view a Wcs.

API Summary

Constructors
`WcsExplorer()` The constructor of a new WcsExplorer.
`WcsExplorer(Object object)` The constructor of a new WcsExplorer associated with the given object.

Methods
`String` **`getName()`** Returns the name for this WcsExplorer.
`String` **`getDescription()`** Returns the description for this WcsExplorer.
`boolean` **`canHandle(Class className)`** Checks whether this WcsExplorer can
`setObject(Object object)` Sets the object for this WcsExplorer to the given object.
`Wcs` **`getObject()`** Returns the object for this WcsExplorer.
`Class` **`getVariableType()`** Returns the expected variable type for this WcsExplorer.
`JComponent` **`getComponent()`** Returns the component that is responsible for displaying the
`boolean` **`makeEditorContent()`** Creates the editor content for this Explorer.
`addDataObjectListener(DataObjectListener arg0)` Add the given listener to this WcsExplorer
`removeDataObjectListener(DataObjectListener arg0)` Removes the given listener for this WcsExplorer,

API details

Constructors

`WcsExplorer()`

WcsExplorer(Object object)

Argument

Object **object** [INPUT, MANDATORY]

Methods

String getName()

Return

String

Returns the name for this WcsExplorer.

String getDescription()

Return

String

Returns the description for this WcsExplorer.

boolean canHandle(Class className)

handle objects of the given class.

Argument

Class **className** [INPUT, MANDATORY]

Return

boolean

Returns true of this WcsExplorer can handle objects of the given class; false otherwise.

setObject(Object object)

Argument

Object **object** [INPUT, MANDATORY]

Wcs getObject()

Return

Wcs

Returns the object for this WcsExplorer.

Class getVariableType()

Return

Class

Returns the expected variable type for this WcsExplorer.

JComponent getComponent()

data object for this WcsExplorer.

Return

JComponent getComponent()

JComponent

Returns the component that is responsible for displaying the data object for this WcsExplorer.

boolean makeEditorContent()

Return

boolean

Returns true if the editor content for this WcsExplorer could be made; false otherwise.

addDataObjectListener(DataObjectListener arg0)

to receive data object events from it.

Argument

DataObjectListener **arg0** [INPUT, MANDATORY]

removeDataObjectListener(DataObjectListener arg0)

so that it no longer receives data object events by this explorer.

Argument

DataObjectListener **arg0** [INPUT, MANDATORY]

3.345. Wcs

Full Name:	herschel.ia.dataset.image.wcs.Wcs
Type:	Java Class -
Import:	from herschel.ia.dataset.image.wcs import Wcs
Category:	Image

Description

A class to create a Wcs.

This class creates a Wcs. The Wcs describes the World coordinate system, which is used to convert between image coordinates and world coordinates.

Example

Example 1: A basic example on how to create a Wcs

```
wcs = Wcs(crval1=30.0, crval2=-89.50, crpix1=109, crpix2=109)
```

API Summary

Constructors
Wcs()
The default constructor.
Wcs(int naxis)
Constructor for 2 or 3 dimensional Wcs.
Wcs(Wcs orig)
The copy constructor.

Methods
Wcs **copy**()
Returns a copy of the Wcs object.
setParameter(String paramname, Object param, String description)
Adds a new parameter to the Wcs.
isCompleteWcs()
Checks whether the Wcs is complete.
Object **getParameter**(String paramname)
Returns a parameter from the Wcs.
Object[] **getParameters**()
Returns all parameters.
removeParameter(String paramname)
Removes a parameter.
setNAxis(int naxis)
Sets the number of axes.
int **getNAxis**()
Returns the number of axes.

Methods
setImageIndex(Double1d index, Unit unit) Sets the image index for a non-equidistant 3rd dimension.
boolean **hasImageIndex**() Checks the image index
TableDataset **getImageIndex**() Returns the image index.
boolean **isEquidistantInZ**() Checks whether the Wcs is equidistant in the 3rd dimension.
setCrval1(double crval1) Sets crval1.
double **getCrval1**() Returns crval1
setCrval2(double crval2) Sets crval2.
double **getCrval2**() Returns crval2
setCrval3(double crval3) Sets crval3.
double **getCrval3**() Returns crval3
setCrpix1(double crpix1) Sets crpix1.
double **getCrpix1**() Returns crpix1
setCrpix2(double crpix2) Sets crpix2.
double **getCrpix2**() Returns crpix2
setCrpix3(double crpix3) Sets crpix3.
double **getCrpix3**() Returns crpix3
setCdelt1(double cdelt1) Sets cdelt1.
double **getCdelt1**() Returns cdelt1
setCdelt2(double cdelt2) Sets cdelt2.
double **getCdelt2**() Returns cdelt2
setCdelt3(double cdelt3)

Methods
Sets cdelt3.
`double` **`getCdelt3`**`()` Returns cdelt3
`boolean` **`checkCtypeValidity`**`(String ctype)` Checks the validity of the ctype parameter.
`setCtype1``(String ctype1)` Sets ctype1.
`String` **`getCtype1`**`()` Returns ctype1
`setCtype2``(String ctype2)` Sets ctype2.
`String` **`getCtype2`**`()` Returns ctype2
`setCtype3``(String ctype3)` Sets ctype3.
`String` **`getCtype3`**`()` Returns ctype3
`setCunit1``(String cunit1)` Sets cunit1.
`String` **`getCunit1`**`()` Returns cunit1
`setCunit2``(String cunit2)` Sets cunit2.
`String` **`getCunit2`**`()` Returns cunit2
`boolean` **`hasParameter`**`(String parameter)` Checks the availability of a parameter.
`setCunit3``(String cunit3)` Sets cunit3.
`String` **`getCunit3`**`()` Returns cunit3
`setEpoch``(double epoch)` Sets the epoch.
`double` **`getEpoch`**`()` Returns the epoch
`setEquinox``(double equinox)` Sets the equinox.
`double` **`getEquinox`**`()` Returns the equinox
`setRadesys``(String Radesys)` Sets the reference frame.

Methods
`String` **`getRadesys`**`()` Returns the reference frame
`setCd1_1``(double cd1_1)` Sets the cd1_1 element.
`setCd1_2``(double cd1_2)` Sets the cd1_2 element.
`setCd1_3``(double cd1_3)` Sets the cd1_3 element.
`setCd2_1``(double cd2_1)` Sets the cd2_1 element.
`setCd2_2``(double cd2_2)` Sets the cd2_2 element.
`setCd2_3``(double cd2_3)` Sets the cd2_3 element.
`setCd3_1``(double cd3_1)` Sets the cd3_1 element.
`setCd3_2``(double cd3_2)` Sets the cd3_2 element.
`setCd3_3``(double cd3_3)` Sets the cd3_3 element.
`double` **`getCd1_1`**`()` Returns cd1_1
`double` **`getCd1_2`**`()` Returns cd1_2
`double` **`getCd2_1`**`()` Returns cd2_1
`double` **`getCd2_2`**`()` Returns cd2_2
`double` **`getCd1_3`**`()` Returns cd1_3
`double` **`getCd2_3`**`()` Returns cd2_3
`double` **`getCd3_1`**`()` Returns cd3_1
`double` **`getCd3_2`**`()` Returns cd3_2
`double` **`getCd3_3`**`()` Returns cd3_3
`setPc1_1``(double pc1_1)` Sets the pc1_1 element.
`double` **`getPc1_1`**`()`

Methods
Returns pc1_1
setPc1_2(double pc1_2) Sets the pc1_2 element.
double **getPc1_2**() Returns pc1_2
setPc1_3(double pc1_3) Sets the pc1_3 element.
double **getPc1_3**() Returns pc1_3
setPc2_1(double pc2_1) Sets the pc2_1 element.
double **getPc2_1**() Returns pc2_1
setPc2_2(double pc2_2) Sets the pc2_2 element.
double **getPc2_2**() Returns pc2_2
setPc2_3(double pc2_3) Sets the pc2_3 element.
double **getPc2_3**() Returns pc2_3
setPc3_1(double pc3_1) Sets the pc3_1 element.
double **getPc3_1**() Returns pc3_1
setPc3_2(double pc3_2) Sets the pc3_2 element.
double **getPc3_2**() Returns pc3_2
setPc3_3(double pc3_3) Sets the pc3_3 element.
double **getPc3_3**() Returns pc3_3
setProjection(String projection) Sets the projection.
String **getProjection**() Returns the projection.
MetaData **getMeta**() Return the Wcs as metadata.
String **toString**() Returns a string representation of the Wcs.

Methods
`double[] getWorldCoordinates(double row, double column)` Returns the world coordinates of the given image coordinates.
`double getZCoordinate(int depth)` Returns the world coordinates of the given layer.
`double[] getPixelCoordinates(double c1, double c2)` Returns the pixel coordinates.
`setCrota2(double crota2)` Sets crota2
`double getCrota2()` Returns crota2.
`boolean isValid()` Checks the possibility to convert from pixel to world coordinates.
`setWavelength(double wavelength)` Sets the wavelength in the Wcs.
`getWavelength()` Returns the wavelength which is stored in the Wcs

API details

Constructors

`Wcs()`

A constructor which creates a standard Wcs object. The standard Wcs sets naxis = 2 (NAXIS = 2)

Example

Typical example on how to create a Wcs.

```
wcs = Wcs(crval1=30.0, crval2=-89.50, crpix1=109, crpix2=109)
```

`Wcs(int naxis)`

A constructor which creates a Wcs object with the chosen number of axes. The standard Wcs has only parameter naxis set

Argument

`int` **`naxis`** `[INPUT, MANDATORY]`

Example

Typical example on how to create a Wcs.

```
wcs = Wcs(3)
wcs.setCrval1(30.0)
```

`Wcs(Wcs orig)`

Constructor for Wcs A constructor which creates a copy of an existing Wcs object.

Argument

`Wcs` **`orig`** `[INPUT, MANDATORY]`

Methods

Wcs copy()

Returns a copy of the Wcs object.

Return

Wcs

A copy of the Wcs.

setParameter(String paramname, Object param, String description)

Adds a new parameter to the Wcs.

Arguments

`String` **paramname** [INPUT, MANDATORY]

`Object` **param** [INPUT, MANDATORY]

`String` **description** [INPUT, MANDATORY]

isCompleteWcs()

Returns true if the Wcs has enough information to convert to and from World Coordinates.

Object getParameter(String paramname)

Returns the requested parameter of the Wcs.

Argument

`String` **paramname** [INPUT, MANDATORY]

Return

Object

The requested parameter.

Object[] getParameters()

Returns the list of all Wcs parameters.

Return

Object[]

The list of all Wcs parameters.

removeParameter(String paramname)

Removes the requested parameter of the Wcs.

Argument

`String` **paramname** [INPUT, MANDATORY]

setNAxis(int naxis)

Sets the number of axes of the Wcs.

Argument

`int` **naxis** [INPUT, MANDATORY]

int getNAxis()

Returns the number of axes of the Wcs.

int getNAxis()

Return

int

The number of axes.

setImageIndex(Double1d index, Unit unit)

Sets the image index for a non-equidistant 3rd dimension.

Arguments

Double1d **index** [INPUT, MANDATORY]

Unit **unit** [INPUT, MANDATORY]

boolean hasImageIndex()

Returns true if there is an image index.

Return

boolean

True if there is an image index.

TableDataset getImageIndex()

Returns the image index for a non-equidistant 3rd dimension.

Return

TableDataset

The image index for a non-equidistant 3rd dimension.

boolean isEquidistantInZ()

isEquidistantInZ returns true when the 3rd axis is equidistant. In this case, the crval3, crpix3 and cdelt3 keywords are used. If the 3rd axis is not equidistant, the ImageIndex is used.

Return

boolean

True if the Wcs is equidistant.

setCrval1(double crval1)

Sets the first coordinate of the center.

Argument

double **crval1** [INPUT, MANDATORY]

double getCrval1()

Returns the first coordinate of the reference pixel.

Return

double

The first coordinate of the reference pixel.

setCrval2(double crval2)

Sets the second coordinate of the center.

setCrval2(double crval2)

Argument

double **crval2** [INPUT, MANDATORY]

double getCrval2()

Returns the second coordinate of the reference pixel.

Return

double

The second coordinate of the reference pixel.

setCrval3(double crval3)

Sets the third coordinate of the center.

Argument

double **crval3** [INPUT, MANDATORY]

double getCrval3()

Returns the third coordinate of the reference pixel.

Return

double

The third coordinate of the reference pixel.

setCrpix1(double crpix1)

Sets the reference pixel position of axis 1. WARNING: CRPIX1 should be given in x,y coordinates. The standard specifies that the image starts at pixel (1,1) and not at pixel (0,0). So to set the first pixel, you should use 1 as value for crpix1.

Argument

double **crpix1** [INPUT, MANDATORY]

double getCrpix1()

Returns the reference pixel position of axis 1.

Return

double

The reference pixel position of axis 1.

setCrpix2(double crpix2)

Sets the reference pixel position of axis 2. WARNING: CRPIX2 should be given in x,y coordinates. The standard specifies that the image starts at pixel (1,1) and not at pixel (0,0). So to set the first pixel, you should use 1 as value for crpix2.

Argument

double **crpix2** [INPUT, MANDATORY]

double getCrpix2()

Returns the reference pixel position of axis 2.

Return

double

double getCrpix2()

The reference pixel position of axis 2.

setCrpix3(double crpix3)

Sets the reference pixel position of axis 3.

Argument

`double` **`crpix3`** `[INPUT, MANDATORY]`

double getCrpix3()

Returns the reference pixel position of axis 3.

Return

double

The reference pixel position of axis 3.

setCdelt1(double cdelt1)

Sets the pixel scale of axis 1.

Argument

`double` **`cdelt1`** `[INPUT, MANDATORY]`

double getCdelt1()

Returns the pixel scale of axis 1.

Return

double

The pixel scale of axis 1.

setCdelt2(double cdelt2)

Sets the pixel scale of axis 2.

Argument

`double` **`cdelt2`** `[INPUT, MANDATORY]`

double getCdelt2()

Returns the pixel scale of axis 2.

Return

double

The pixel scale of axis 2.

setCdelt3(double cdelt3)

Sets the pixel scale of axis 3.

Argument

`double` **`cdelt3`** `[INPUT, MANDATORY]`

double getCdelt3()

Returns the pixel scale of axis 3.

double getCdelt3()

Return

double

The pixel scale of axis 3.

boolean checkCtypeValidity(String ctype)

Checks the validity of the ctype parameter.

Argument

`String` **ctype** [INPUT, MANDATORY]

Return

boolean

true if the ctype parameter is valid.

setCtype1(String ctype1)

Sets the projection type of axis 1.

Argument

`String` **ctype1** [INPUT, MANDATORY]

String getCtype1()

Returns the projection type of axis 1.

Return

String

The projection type of axis 1.

setCtype2(String ctype2)

Sets the projection type of axis 2.

Argument

`String` **ctype2** [INPUT, MANDATORY]

String getCtype2()

Returns the projection type of axis 2.

Return

String

The projection type of axis 2.

setCtype3(String ctype3)

Sets the projection type of axis 3.

Argument

`String` **ctype3** [INPUT, MANDATORY]

String getCtype3()

Returns the projection type of axis 3.

Return

String getCtype3()

String

The projection type of axis 3.

setCunit1(String cunit1)

Sets the unit of axis 1.

Argument

`String` **cunit1** [INPUT, MANDATORY]

String getCunit1()

Returns the unit of axis 1.

Return

String

The unit of axis 1.

setCunit2(String cunit2)

Sets the unit of axis 2.

Argument

`String` **cunit2** [INPUT, MANDATORY]

String getCunit2()

Returns the unit of axis 2.

Return

String

The unit of axis 2.

boolean hasParameter(String parameter)

Checks whether the parameter is available.

Argument

`String` **parameter** [INPUT, MANDATORY]

Return

boolean

True if the parameter is available.

setCunit3(String cunit3)

Sets the unit of axis 3.

Argument

`String` **cunit3** [INPUT, MANDATORY]

String getCunit3()

Returns the unit of axis 3.

Return

String

String getCunit3()

The unit of axis 3.

setEpoch(double epoch)

Sets the epoch.

Argument

`double` **epoch** `[INPUT, MANDATORY]`

double getEpoch()

Returns the epoch.

Return

double

The epoch.

setEquinox(double equinox)

Sets the equinox.

Argument

`double` **equinox** `[INPUT, MANDATORY]`

double getEquinox()

Returns the equinox.

Return

double

The equinox.

setRadesys(String Radesys)

Sets the reference frame.

Argument

`String` **Radesys** `[INPUT, MANDATORY]`

String getRadesys()

Returns the reference frame.

Return

String

The reference frame.

setCd1_1(double cd1_1)

Sets element (1, 1) of the corrected CD matrix CDi_j..

Argument

`double` **cd1_1** `[INPUT, MANDATORY]`

setCd1_2(double cd1_2)

Sets element (1, 2) of the corrected CD matrix CDi_j..

Argument

setCd1_2(double cd1_2)

double **cd1_2** [INPUT, MANDATORY]

setCd1_3(double cd1_3)

Sets element (1, 3) of the corrected CD matrix CDi_j..

Argument

double **cd1_3** [INPUT, MANDATORY]

setCd2_1(double cd2_1)

Sets element (2, 1) of the corrected CD matrix CDi_j..

Argument

double **cd2_1** [INPUT, MANDATORY]

setCd2_2(double cd2_2)

Sets element (2, 2) of the corrected CD matrix CDi_j..

Argument

double **cd2_2** [INPUT, MANDATORY]

setCd2_3(double cd2_3)

Sets element (2, 3) of the corrected CD matrix CDi_j..

Argument

double **cd2_3** [INPUT, MANDATORY]

setCd3_1(double cd3_1)

Sets element (3, 1) of the corrected CD matrix CDi_j..

Argument

double **cd3_1** [INPUT, MANDATORY]

setCd3_2(double cd3_2)

Sets element (3, 2) of the corrected CD matrix CDi_j..

Argument

double **cd3_2** [INPUT, MANDATORY]

setCd3_3(double cd3_3)

Sets element (3, 3) of the corrected CD matrix CDi_j..

Argument

double **cd3_3** [INPUT, MANDATORY]

double getCd1_1()

Returns element (1,1) of the corrected CD matrix CDi_j.

Return

double

Element (1,1) of the corrected CD matrix CDi_j.

double getCd1_2()

Returns element (1,2) of the corrected CD matrix CDi_j.

double getCd1_2()

Return

double

Element (1,2) of the corrected CD matrix CDi_j.

double getCd2_1()

Returns element (2,1) of the corrected CD matrix CDi_j.

Return

double

Element (2,1) of the corrected CD matrix CDi_j.

double getCd2_2()

Returns element (2,2) of the corrected CD matrix CDi_j.

Return

double

Element (2,2) of the corrected CD matrix CDi_j.

double getCd1_3()

Returns element (1,3) of the corrected CD matrix CDi_j.

Return

double

Element (1,3) of the corrected CD matrix CDi_j.

double getCd2_3()

Returns element (2,3) of the corrected CD matrix CDi_j.

Return

double

Element (2,3) of the corrected CD matrix CDi_j.

double getCd3_1()

Returns element (3,1) of the corrected CD matrix CDi_j.

Return

double

Element (3,1) of the corrected CD matrix CDi_j.

double getCd3_2()

Returns element (3,2) of the corrected CD matrix CDi_j.

Return

double

Element (3,2) of the corrected CD matrix CDi_j.

double getCd3_3()

Returns element (3,3) of the corrected CD matrix CDi_j.

Return

double

Element (3,3) of the corrected CD matrix CDi_j.

setPc1_1(double pc1_1)

Sets element (1, 1) of the linear transformation matrix PCi_j..

Argument

double **pc1_1** [INPUT, MANDATORY]

double getPc1_1()

Returns element (1,1) of the linear transformation matrix PCi_j.

Return

double

Element (1,1) of the linear transformation matrix PCi_j.

setPc1_2(double pc1_2)

Sets element (1, 2) of the linear transformation matrix PCi_j..

Argument

double **pc1_2** [INPUT, MANDATORY]

double getPc1_2()

Returns element (1,2) of the linear transformation matrix PCi_j.

Return

double

Element (1,2) of the linear transformation matrix PCi_j.

setPc1_3(double pc1_3)

Sets element (1, 3) of the linear transformation matrix PCi_j..

Argument

double **pc1_3** [INPUT, MANDATORY]

double getPc1_3()

Returns element (1,3) of the linear transformation matrix PCi_j.

Return

double

Element (1,3) of the linear transformation matrix PCi_j.

setPc2_1(double pc2_1)

Sets element (2, 1) of the linear transformation matrix PCi_j..

Argument

double **pc2_1** [INPUT, MANDATORY]

double getPc2_1()

Returns element (2,1) of the linear transformation matrix PCi_j.

Return

double

Element (2,1) of the linear transformation matrix PCi_j.

setPc2_2(double pc2_2)

Sets element (2, 2) of the linear transformation matrix PCi_j..

Argument

double **pc2_2** [INPUT, MANDATORY]

double getPc2_2()

Returns element (2,2) of the linear transformation matrix PCi_j.

Return

double

Element (2,2) of the linear transformation matrix PCi_j.

setPc2_3(double pc2_3)

Sets element (2, 3) of the linear transformation matrix PCi_j..

Argument

double **pc2_3** [INPUT, MANDATORY]

double getPc2_3()

Returns element (2,3) of the linear transformation matrix PCi_j.

Return

double

Element (2,3) of the linear transformation matrix PCi_j.

setPc3_1(double pc3_1)

Sets element (3, 1) of the linear transformation matrix PCi_j..

Argument

double **pc3_1** [INPUT, MANDATORY]

double getPc3_1()

Returns element (3,1) of the linear transformation matrix PCi_j.

Return

double

Element (3,1) of the linear transformation matrix PCi_j.

setPc3_2(double pc3_2)

Sets element (3, 2) of the linear transformation matrix PCi_j..

Argument

setPc3_2(double pc3_2)

double **pc3_2** [INPUT, MANDATORY]

double getPc3_2()

Returns element (3,2) of the linear transformation matrix PCi_j.

Return

double

Element (3,2) of the linear transformation matrix PCi_j.

setPc3_3(double pc3_3)

Sets element (3, 3) of the linear transformation matrix PCi_j..

Argument

double **pc3_3** [INPUT, MANDATORY]

double getPc3_3()

Returns element (3,3) of the linear transformation matrix PCi_j.

Return

double

Element (3,3) of the linear transformation matrix PCi_j.

setProjection(String projection)

Sets the projection type.

Argument

String **projection** [INPUT, MANDATORY]

String getProjection()

Returns the projection type.

Return

String

The projection type.

MetaData getMeta()

Returns the Wcs as metadata.

Return

MetaData

The Wcs as metadata.

String toString()

Returns a string representation of the values of the Wcs object. The format of this string is undefined and subject to change.

Return

String

String toString()

a string representation of the values of the Wcs object.

double[] getWorldCoordinates(double row, double column)

Returns the world coordinates when the image coordinates are given.

Arguments

`double row [INPUT, MANDATORY]`

`double column [INPUT, MANDATORY]`

Return

double[]

The corresponding world coordinates as a 2 dimensional array. When the ctype1 and ctype2 keywords of the wcs are defined as "RA---TAN" and "DEC--TAN", the first coordinates describes the right ascension and the second the declination.

double getZCoordinate(int depth)

Returns the world coordinates of the given layer.

Argument

`int depth [INPUT, MANDATORY]`

Return

double

The corresponding world coordinates of the Z axis.

double[] getPixelCoordinates(double c1, double c2)

Returns the pixelCoordinates of the given SkyCoordinates. The pixel- coordinates are the row and the column of the image!

Arguments

`double c1 [INPUT, MANDATORY]`

`double c2 [INPUT, MANDATORY]`

Return

double[]

A 2-dimensional array of doubles describing the pixel coordinates of the given world coordinates.

setCrota2(double crota2)

Sets the rotation angle in degrees.

Argument

`double crota2 [INPUT, MANDATORY]`

double getCrota2()

Returns the rotation angle in degrees.

Return

double

The rotation angle in degrees.

boolean isValid()

Checks if the conversion from pixel to world coordinates is possible and returns true if this is the case.

Return

boolean

true is the conversion from pixel to world coordinates is possible.

setWavelength(double wavelength)

Sets the wavelength in the Wcs (in micrometers). The WAVELNTH and the WAVEUNIT parameter are set.

Argument

`double` **`wavelength`** `[INPUT, MANDATORY]`

getWavelength()

Returns the wavenlength in micrometer.

3.346. WeightedMean

Full Name:	herschel.ia.numeric.toolbox.basic.WeightedMean
Type:	Java Class -
Import:	from herschel.ia.numeric.toolbox.basic import WeightedMean

Description

This class calculates the quadratically weighted mean and the uncertainty of it, based on

the assumption that the distribution of the errors is Gaussian, of an array of numbers and its corresponding uncertainties in the same units.

Formula:

```
Let "x" be a vector of n numbers for input and "dx" be a vector of "n" numbers
that represent the uncertainties of "x" in the same units.
Let "x[i]" be the i'st element of a vector "x".
Let sum() be the sum over all indices i from 1 to n.
The weighted mean wmean is then:
wmean = sum( x[i]/(dx[i]^2) ) / sum( 1/(dx[i]^2))
The uncertainty of the weighted mean is then:
ewmean = sqrt( sum( (x[i]-wmean)^2/(dx[i]^2) ) / sum( 1/(dx[i]^2) ) )
```

NaN numbers: if ignoreNaN is set to 'true', a NaN value, either in values or uncertainties, is not used in the procedure. If ignoreNaN if set to 'false' (default value) and a NaN is found, a NaN value is returned. NOTE: NaN values can be used for double, float and complex arrays only.

Example

Example 1: Untitled

```
values = Double1d([1,3,5])
errors = Double1d([0.1,0.2,0.3])
wmean = WeightedMean(values,errors)
print wmean.mean()
print wmean.error()
values = Complex1d([1+1j,3+1j,5+1j])
errors = Complex1d([0.1+1j,0.2+1j,0.3+1j])
wmean = WeightedMean(values,errors)
print wmean.mean()
print wmean.error()
</pre>
```

API Summary

Jython Syntax

```
wm = WeightedMean(<values>,<errors>,<ignoreNaN>)
wm.mean()
wm.error()
```

Properties

```
number array values [INPUT, MANDATORY, default=no default value]
number array errors [INPUT, MANDATORY, default=no default value]
boolean ignoreNaN [INPUT, OPTIONAL, default=false]
```

API details

Properties

number array values [INPUT, MANDATORY, default=no default value]

The input numbers array can be integer, long, float, double and complex one dimensional array.

number array errors [INPUT, MANDATORY, default=no default value]

The input uncertainties array can be integer, long, float, double and complex one dimensional array.

boolean ignoreNaN [INPUT, OPTIONAL, default=false]

If 'true' a NaN value will be ignored. By default is 'false'.

See also

- WeightedMean