test project Documentation

Release 1

Colin Kennedy

July 02, 2015

Contents

My introduction to this project

Requirements:

This project requires xyz package.

Contents:

CHAPTER 1

Test Project Tutorial

1.1 This is a tutorial on how to use Test Project

And now you're reading the tutorial

Goodbye!

CHAPTER 2

Project Summary

2.1 Goals Achieved

Goal 1 - Wake up before you go-go

Goal 2 - Wasn't hanging on like a yo-yo My introduction to this project

2.2 Lessons Learned

Every dog has his day in da hood

CHAPTER 3

Auto Generated Documentation

This module illustrates how to write your docstring in OpenAlea and other projects related to OpenAlea.

class testproject.**MainClass1**

This class docstring shows how to use sphinx and rst syntax

The first line is brief explanation, which may be completed with a longer one. For instance to discuss about its methods. The only method here is *function1()*'s. The main idea is to document the class and methods's arguments with

•**parameters**, **types**, **return** and **return types**:

```
:param arg1: description
:param arg2: description
:type arg1: type description
:type arg1: type description
:return: return description
:rtype: the return type description
```

•and to provide sections such as **Example** using the double commas syntax:

```
:Example:

followed by a blank line !
```

which appears as follow:

Example

followed by a blank line

•Finally special sections such as **See Also**, **Warnings**, **Notes** use the sphinx syntax (*paragraph directives*):

```
.. seealso:: blabla
.. warnings also:: blabla
.. note:: blabla
.. todo:: blabla
```

Note:

There are many other Info fields but they may be redundant:

- param, parameter, arg, argument, key, keyword: Description of a parameter.
- type: Type of a parameter.
- raises, raise, except, exception: That (and when) a specific exception is raised.

- var, ivar, cvar: Description of a variable.
- returns, return: Description of the return value.
- rtype: Return type.

Note: There are many other directives such as versionadded, versionchanged, rubric, centered, ... See the sphinx documentation for more details.

Here below is the results of the *function1()* docstring.

function1(*arg1*, *arg2*, *arg3*)

returns (arg1 / arg2) + arg3

This is a longer explanation, which may include math with latex syntax . Then, you need to provide optional subsection in this order (just to be consistent and have a uniform documentation. Nothing prevent you to switch the order):

•parameters using `:param <name>: <description>`

•type of the parameters `:type <name>: <description>`

•returns using `:returns: <description>`

•examples (doctest)

•seealso using `.. seealso:: text`

•notes using `.. note:: text`

•warning using `.. warning:: text`

•todo `.. todo:: text`

Advantages:

- Uses sphinx markups, which will certainly be improved in future version
- Nice HTML output with the See Also, Note, Warnings directives

Drawbacks:

- Just looking at the docstring, the parameter, type and return sections do not appear nicely

Parameters

- **arg1** (*int, float,...*) – the first value
- **arg2** (*int, float,...*) – the first value
- **arg3** (*int, float,...*) – the first value

Returns arg1/arg2 +arg3

Return type int, float

Example

```
>>> import template
>>> a = template.MainClass1()
>>> a.function1(1,1,1)
2
```

Note: can be useful to emphasize important feature

See also:

`MainClass2`

Warning: arg2 must be non-zero.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

t

F

M

T