HOW TO USE AN ADD DOCUMENTATION

We use the Sphinx module to easily create documentation for any kind of code. It can read the docstrings of all the different objects in a code file to make it appear in the documentation. We also use here the autosummary extension that creates automatically all the files and supports for this documentation. It is based on the ReStructuredText langage so if you need some information about its syntax, please go here : http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html.

Composition of the doc folder:

In the doc folder you will find:

- **_build** folder with mainly the html pages created with sphinx (not by default for Sphinx)
- **_static** folder with nothing really interesting for this tutorial
- **_templates** folder with the autosummary template
- **generated** folder created by autosummary and containing all the generated files (not by default for Sphinx)
- **conf.py** file that contains all the documentation configuration
- **make.bat** file not used here
- **Makefile** file that contains all the option for ‘make’ command
- ***.rst** files

The documentation is built thanks to all these ***.rst** files.

Toctree construction:

Example:

```
.. toctree:: generated (option)
   :option1:
   :option2:

   example
```

In this case, Sphinx will look for an **example.rst** file in a *./generated* folder.
If you don’t add the *generated* option , Sphinx will look for an **example.rst** file in the same directory.
We have different options available:
- maxdepth: express the number of tabs and subtabs displayed by the documentation. If you don’t use it, Sphinx will display all. It seems that this option is not available with autosummary !
- caption: just a kind of title
Other options are available but not used here.

Warning : The names is in the toctree refer to another .rst file but not to a tab in the created documentation !

Structure of the documentation:

The first stage is the **index.rst** (which is called in the conf.py file as the master_doc).
It contains the toctree with the main tabs of the documentation (*mpsrad, howto* and *credits* here).

Considering the toctree construction, you’ll find an .rst file each time you have a name in a toctree. In the same directory of index.rst, you’ll find a .rst file for all the main subpackages and submodules of the package.

The documentation for each one (except for the gui subpackage) is then generated by the sphinx extension **autosummary**.

I will illustrate how it works with one of them.

Here we have the beginning of the mpsrad.fronted.rst file

```
Frontend package
================

dbr module
``````````

.. automodule:: mpsrad.frontend.dbr

.. autosummary::
   :toctree: generated

   dbr
```

**Frontend package** designates the main title of this tab because it is underline by a serie of = (exactly the same length as the title).

**dbr module** is the subtitle because underlined wit a serie of ` (again the same length)

**.. automodule:: name of the module** : Sphinx will display the docstrings of the module found in this path but not detail the different classes and function.

**.. autosummary::** It creates a toctree (note that it is here the option of autosummary) and a generated folder where to find the mpsrad.frontend.dbr.dbr.rst file automatically created as well.

Autosummary will decompose each module in class, then class in methods and attributes and create a file for each one of them. That is the reason why we have so many files in the generated folder.

If we don't use autosummary, we have to build each file manually. You can see an example in the mpsrad.gui.rst file. There, to display the different classes, you have to add **.. autoclass:: name of the class** under the **.. automodule :: name of the module** corresponding. Then you can carry on if you want the different functions with **.. autofunction:: name of the function** of each class and so on.

Be careful of the indentation in the .rst files because it is 3 spaces.

<u>Add documentation:</u>

If you want to add a subpackage/submodule in the package:

1. In mpsrad.rst and under the subpackages/submodule header, add mpsrad.newname to the list.
2. Create in the same directory a file : mpsrad.newname.rst
3. Copy the syntax and structure of the mpsrad.backend.rst file (for instance)
4. Adapt the names to your new folder and its content.
5. List the classes you want to document in the autosummary toctree and the autosummary will make the rest.
   Don't forget the *generated* to have all the .rst files created in the same directory.

You can now see the docstrings of your code if it matches with the ReStructuredText syntax.
Make sure not to have lines of simply code because Sphinx will maybe run it and abort the documentation of this file. To avoid that make sure to have just functions or classes in your code.
```