**https://github.com/robotology/how-to-document-modules/blob/master/README.md**

**Introduction**

This template repository provides a detailed guide on how your modules, and possibly their API you exposed as *IDL services*, should be documented.

A clean and handy way to deal with your documentation is to exploit the [**GitHub Pages**](https://pages.github.com/), which represents a nice tool to publish software documentation online.

Everything is basically done through the special branch called **gh-pages**. The branch must contain at the root level the **index.html** file pointing to the static documentation (generally generated via [**doxygen**](http://www.doxygen.org)) stored somewhere within the branch itself. Further, the static documentation might be composed of many products (sometimes images), thus it would be worth saving space on the repository by not retaining any history for those files.

First off, we explain how to **create the infrastructure** for the documentation using *git* and keep it **up-to-date** throughout the changes the code undergoes. Then, we will dig into how we can use **doxygen** to generate the documentation.

Let's start ✨

**Creating the infrastructure**

From master, create a new branch called gh-pages on your repository. Then, locally do:

git checkout master

git fetch origin

git branch gh-pages --track origin/gh-pages

git checkout gh-pages

Now you have the same branch gh-pages also on your machine which tracks the *remote* gh-pages.

Next step is to create the **static documentation locally**:

* Provide thus a **sub-directory** called **doxygen** where you have to put the file **generate.txt** that tells doxygen how to produce the documentation. You can find a template generate.txt file [here](https://github.com/robotology/how-to-document-modules/blob/gh-pages/doxygen/generate.txt) within this repository.

Inside the generate.txt file, the most important parameter you are required to fill in is **INPUT**, which specifies the directories containing your code to be documented (a recursive search is typically done). An example follows:

INPUT = ../src \

../idl\_dox \

generated-from-xml

Typically, to be neat, the file generate.txt contains instructions to generate the documentation under **doxygen/doc** via the parameter **OUTPUT\_DIRECTORY**.

To find out more about other doxygen parameters, we suggest you to look directly at the online guide.

* Now type:

cd doxygen

doxygen ./generate.txt

This will generate your documentation. The new directory **doxygen/doc** will appear containing all your documentation.

* Create the index.html file at the root level of your repository that provides the entry point to your documentation. This is an [**example**](https://github.com/robotology/how-to-document-modules/blob/gh-pages/index.html) you have to tailor to your specific repository.
* Stage, commit and push:

git add ./doxygen

git add ./index.html

git commit -m "provided doxygen documentation"

git push origin gh-pages

After publishing the changes, you can visit the page <http://robotology.github.com/how-to-document-modules> (of course use your github account and repository name in the url) and you will be redirected to the online documentation.

Finally, it is also a good practice to cite that url from within the README.md file.

**Updating the documentation**

By creation, the special gh-pages branch should always mirror the master branch and should contain two things more: the doxygen directory along with the index.html file. Regarding the commit history, gh-pages should be always [one commit ahead of master](https://github.com/robotology/how-to-document-modules/network).

Whenever you update master branch then, do the following to update the documentation accordingly:

git checkout gh-pages

git rebase master

cd doxygen

rm -rf ./doc

doxygen ./generate.txt

git add ./doc

git log -1

git commit --amend

git push --force-with-lease

git checkout master

The **git log -1** command serves as verification and does display the very last commit message on the gh-pages branch, which must be "*provided doxygen documentation*", that is the one specified initially at creation time. The combination of **git commit --amend** and **git push --force-with-lease** aim to modify the latest stored commit instead of creating a brand new one and eventually force publishing it. This way, we always retain only one commit for the documentation instead of dealing with its whole history.