Setup Doxygen for code documentation

This guide illustrates how to set up the Doxygen config file to document C/C++, python, java, and Fortran code. See also https://doxygen.nl/manual/index. html.

Step-by-step guide

Prerequisites: You must have a working version of Doxygen installed on your system.

1. Navigate to the top-level directory of the code you want to document.

Use doxygen -g <config-file> to create a new config file. If you don't provide a name for config-file, Doxygen will generate a file named Doxyfile.

```
# Our files are located in /path/to/src/new_project
cd new_project
# in project folder, do
doxygen -g <config-file>
# replacing <config-file> with your chosen file name, or leaving it empty
```

2. In the configuration file, there are some general settings that need to be changed from their default values:

```
# _____
# GENERAL SETTINGS
# set the project name
PROJECT_NAME = "My New Project"
# _____
# SPECIFYING WHAT SHOULD BE DOCUMENTED
# to include files in subdirectories, we need to change the RECURSIVE tag from NO to YES
RECURSIVE = YES
# to map and generate documentation for all files, set EXTRACT_ALL from NO to YES
EXTRACT_ALL = YES
# to include private class members in the documentation, set EXTRACT_PRIVATE from NO to YES
EXTRACT PRIVATE = YES
# to include static class members in the documentation, set EXTRACT_STATIC from NO to YES
EXTRACT STATIC = YES
# ------
# BETTER OUTPUT DESIGN
# to improve readability of output, set HIDE_SCOPENAMES from NO to YES
HIDE_SCOPE_NAMES = YES
# _____
# INCLUDING SOURCE CODE IN DOCUMENTATION
# to cross-reference source files with the documentation, SOURCE_BROWSER needs to be set to YES
SOURCE BROWSER = YES
# to be able to apply input filters to the source files later on, set
FILTER_SOURCE_FILES = YES
```

There are several options to further customize Doxygen. We will discuss some of them in the following.
 a. including and excluding files

```
# assuming our project has the following structure:
# new project
# |- Doxyfile
# |- python_files
# |
    - a_file.py
    |- another_file.py
# |
#
 - examples
# |
       - example1.py
# |
       | ...
#
       - example100.py
  #
  - cpp_files
    - headerfiles
# |
# | | - some_header.h
# | | - another_header.h
# | |- src
# |
           - some file.cpp
#
                   - another_file.cpp
# |
           - examples
              |- more_examples.cpp
# |
# |- yet_another_folder
# include by folder: we want to ignore yet_another_folder
INPUT = python_files cpp_files
# exclude all example files
EXCLUDE_PATTERNS = */examples/*
# only include .py and .cpp files
FILE_PATTERNS = .py .cpp
```

 b. language-specific settings: It is possible to optimize Doxygen for a specific programming language. If your project contains only of C /C++, python/java, or Fortran files, you can specify this in the settings.

```
# Set the OPTIMIZE_OUTPUT_FOR_C tag to YES if your project consists of C sources
# only. Doxygen will then generate output that is more tailored for C. For
# instance, some of the names that are used will be different. The list of all
# members will be omitted, etc.
# The default value is: NO.
OPTIMIZE_OUTPUT_FOR_C = NO
# Set the OPTIMIZE_OUTPUT_JAVA tag to YES if your project consists of Java or
# Python sources only. Doxygen will then generate output that is more tailored
# for that language. For instance, namespaces will be presented as packages,
# qualified scopes will look different, etc.
# The default value is: NO.
OPTIMIZE_OUTPUT_JAVA = NO
# Set the OPTIMIZE_FOR_FORTRAN tag to YES if your project consists of Fortran
# sources. Doxygen will then generate output that is tailored for Fortran.
# The default value is: NO.
OPTIMIZE_FOR_FORTRAN = NO
```

c. There are many ways to further customize Doxygen. The autogenerated config file includes detailed descriptions of every setting.

Related articles

- Setup Doxygen for code documentation
- Install Doxygen