

Doxygen

⇒ Doxygen is the de facto standard tool for generating documentation from annotated C++ sources.

⇒ Doxygen can help you in 3 ways:

1) Generate on-line documentation (HTML)

~~2)~~ 2) You can configure doxygen to extract the code structure from undocumented source file.

↳ Doxygen can also visualize the relations between the various elements by means of

↳ Dependency graphs
↳ Inheritance diagram
↳ Collaboration diagram

} Generated automatically

3) You can also use doxygen for creating manual documentation.


★ Getting started

⇒ The executable doxygen is the main program that parses the sources and generates the documentation.

Step 1: Creating a Configuration file

- ⇒ Dxygen uses a configuration file to determine all of its settings.
- ⇒ Each project should get its own configuration file.
- ⇒ To simplify the creation of a configuration file, dxygen can create a template configuration file for you.

`dxygen -g <config-file>`

name of configuration file

- ⇒ The configuration file has a format that is similar to that of a (Simple) Makefile.

TAGNAME = VALUE or

TAGNAME = VALUE1 VALUE2

- ⇒ You can probably leave the values of most tags in a generated template configuration file to their default value.
- ⇒ You should assign the start directory or directories to the **INPUT** tag.
- ⇒ Add one or more file patterns to the **FILE-PATTERNS** tag.
(example *.cpp *.h)
 - ↳ Only files that match one of the patterns will be parsed.
- ⇒ For recursive parsing of a source tree you must set the **RECURSIVE** tag to YES.
- ⇒ To further fine-tune the list of files that is parsed, the **EXCLUDE** and **EXCLUDE-PATTERNS** tags can be used.

Step 2: Running Doxygen

⇒ To generate the documentation you can now enter:

`doxygen <config-file>`

⇒ The default output directory is the directory in which doxygen is started.

↳ The root directory to which the output is written can be changed using the **OUTPUT_DIRECTORY**.

⇒ The format specific directory within the output directory can be selected using the **HTML_OUTPUT** tag in the configuration file.

Step 3: Documenting the Source

⇒ If the **EXTRACT_ALL** option is set to NO in the configuration file (the default), then doxygen will only generate documentation for documented entities.

⇒ For members, classes and namespace there are basically two options:

① Place a Special documentation block in front of the declaration or definition of the member, class or namespace.

② Place a special documentation block somewhere else and put a structured command in the documentation block.

↳ Structured command links a documentation block to a certain entity that can be documented (e.g. member, class, namespace)

⇒ Files can only be documented using the second option, since there is no way to put a documentation block before a file.

⇒ During parsing the following steps take place:

⊛ Markdown formatting is replaced by corresponding HTML or special commands.

⊛ Special Commands inside the documentation are executed.

⊛ If a line starts with some whitespace followed by more asterisks (*) and then optionally more whitespace, then all whitespace and asterisks are removed.

⊛ All resulting blank lines are treated as a paragraph separator.

⊛ Links are created for words corresponding to documented classes.

↳ Unless the word is preceded by a % then the word will not be linked and the % sign is removed.

⊛ Links to members are created when certain patterns are found in the text.

★ Documenting the Code

Special Comment blocks

→ A Special Comment block is a C/C++ style comment block with some additional markings, so doxygen knows it is a piece of structured text that needs to end up in the generated documentation.

⇒ For each entity in the Code there are two type of descriptions:

- ① A brief description
 - ② Detailed description
- { Both are optional }

⇒ For methods and functions there is also a third type of description, the so called **body description**.

↳ Consists of the concatenation of all comment blocks found within the body of the method or function

```
/**  
 * --- text ---  
 */
```

```
/*!  
/*! --- text ---  
/*!
```

→ Comment Block

⇒ Tags for documenting functions, Classes, Methods and so on.

/*! @brief <brief-description>

/*! <detailed-description>

/*! @param <PARAM> <DESCRIPTION>

/*! @param[in] <PARAM> <DESCRIPTION>

/*! @param[out] <PARAM> <DESCRIPTION>

/*! @param[in, out] <PARAM> <DESCRIPTION>

/*! @return <DESCRIPTION>

/*! @remark <additional side note>

⇒ Tags for documenting file:

#!/@file <FILENAME>

#!/@author <AUTHOR_NAME>

#!/@brief <BRIEF>

#!/@date <DATE>

Note: Those tags should be placed at top of file