CMake Variable Injection

23/02/2020 By An7ar35

Contents

1 Introduction

Sometimes it's useful to inject CMake variables into the source code such as in the case of printing the project versions and do conditional builds.

Aside from CMake's internal variables, more can be created by using the `set(...)` command in the `CMakeList.txt` file. For example:

```
//Interal CMake variable assignment for project name and version
project(AwesomeProject VERSION 1.0.5)
//Custom variable declaration for the project's author
set(PROJECT_AUTHOR "ElDude")
//Internal CMake variable assignment for the project's URL
set(PROJECT_HOMEPAGE_URL "www.awesome-project.com")
```

The example of the versioning will be used to demonstrate the concept.

2 Skeleton C++ code for injection

We need to create some c++ skeleton code from which CMake will generate the actual code with the variables inserted from.

Note

The skeleton file is **not** included in the project as it is only used by CMake to generate the target file we actually want.

To insert a CMake variable just use the `@VARIABLE_NAME@` syntax. Let's create code for our declared variables above...

`src/cmake_variables.h`

```
#ifndef AWESOMEPROJECT_CMAKE_VARIABLES_H
#define AWESOMEPROJECT_CMAKE_VARIABLES_H

#include <string>

namespace awesome_project::cmake {
    inline static const std::string AUTHOR        = "@PROJECT_AUTHOR@";
    inline static const std::string URL           = "@PROJECT_HOMEPAGE_URL@";
    inline static const std::string VERSION       = "@PROJECT_VERSION@";
    static const unsigned           VERSION_MAJOR = @PROJECT_VERSION_MAJOR@;
    static const unsigned           VERSION_MINOR = @PROJECT_VERSION_MINOR@;
    static const unsigned           VERSION_PATCH = @PROJECT_VERSION_PATCH@;
}

#endif //AWESOMEPROJECT_CMAKE_VARIABLES_H
```

3 Getting CMake to generate from skeleton code

First, `configure_file(...)` is used to indicate what file to take as template and where to output the generated code to. Here our `src/cmake_variable.h` header file is used and the output is set to the `generated/` directory inside `CMAKE_BINARY_DIR`. The latter is Usually set as `cmake_build_debug/` or `cmake_build_release/` by default depending on the build type.

Inside `CMakeList.txt`

```
configure_file(src/cmake_variables.h ${CMAKE_BINARY_DIR}/generated/project_version.h )
include_directories( ${CMAKE_BINARY_DIR}/generated/ )
```

Reload the `CMakeList.txt` file to generate the file.

Note

Including the target directory for the generated file is required in order to be able to use the variables in the project.

The generated output file will then be:

`${CMAKE_BINARY_DIR}/generated/project_version.h`

```
#ifndef AWESOMEPROJECT_CMAKE_VARIABLES_H
#define AWESOMEPROJECT_CMAKE_VARIABLES_H

#include <string>

namespace awesome_project::cmake {
    inline static const std::string AUTHOR        = "ElDude";
    inline static const std::string URL           = "www.awesome-project.com";
    inline static const std::string VERSION       = "1.0.5";
    static const unsigned           VERSION_MAJOR = 1;
    static const unsigned           VERSION_MINOR = 0;
    static const unsigned           VERSION_PATCH = 5;
}

#endif //AWESOMEPROJECT_CMAKE_VARIABLES_H
```