CDevStudio Documentation

Simon Wächter

January 16, 2014

Contents

1 Introduction

This developer guide gives an overview about the CDevStudio project, the requirements for it and some common work flow patterns.

1.1 Idea of this documentation

The idea is to document everything that is needed for this project in one document. With this documentation, a new person should be able to work for the project.

1.2 Content of this documentation

This documentation gives an overview about the design of the project and some common work flow patterns. For a code documentation check out the code documentation generated by Doxygen.

2 Documentation

There are two documentation for the CDevStudio project.

2.1 Code documentation

The code is documented with Doxygen. This documentation gives an overview about the codebase of the project, means classes, methods etc.

2.2 Developer documentation

As addition to the code documentation, there is a developer documentation. This documentation gives an overview about the design and implementation of the project. It does not contain code specific details.

3 Codebase

CDevStudio is based on a 3 layer structure. This means, the program is divided into a graphical (CDevStudio, CDevStudioCodeEdit), a business (CDevStudioSystemPlatform and CDevStudioProjectPlatform) and a backend (CDevStudioBackend) layer.

3.1 CDevStudio

CDevStudio is the main program. It contains the graphical user interface and uses CDevStudioCodeEdit to display code. It accesses CDevStudioProjectPlatform for project creation, lading, saving, editing etc.

3.2 CDevStudioCodeEdit

CDevStudioCodeEdit is able to display and highlight the code of a project.

3.3 CDevStudioSystemPlatform

CDevStudioSystemPlatform is one of the two business layer of the project. The layer is able to read files and translations. For I/O interactions, it uses the CDevStudioBackend layer.

3.4 CDevStudioProjectPlatform

CDevStudioProjectPlatform is one of the two business layer of the project. The layer is able to create, load, delete and save projects. For I/O interactions, it uses the CDevStudioBackend layer.

3.5 CDevStudioBackend

CDevStudioBackend is the backend layer. It is responsible for I/O interactions.

4 Work flow

There are a few techniques that are needed for this project: A working toolchain and some Git knowledge.

4.1 Requirements

CDevStudio has a few requirements. You can split them into build and package requirements:

- Build requirements
  - Working C/C++ toolchain with support for the C++11 standard
  - Git
  - CMake (2.8.11 or higher)
  - Qt 5 (5.1.0 or higher)
  - Doxygen (Code documentation - Optional)
  - Latex (Developer documentation - Optional)
- Package requirements
  - Linux Debian: build-essentials, dh_make, devscripts
  - Linux Fedora: rpmbuild
  - Linux Arch Linux: base-devel
  - Linux Windows: NSIS

4.2 Initialize workspace

For the workspace initialization, please clone the repository and create a new branch:

- git clone http://github.com/swaechter/cdevstudio
- cd cdevstudio
- git checkout -b develop origin/develop

4.3 Build project

Now the project is initialized and you can start with a first build. CDevStudio uses CMake as build system. It can generate project files for different IDE's. To run CMake and build the project run these commands:

- mkdir build
- cd build
- cmake ..
- make (or the build command of your toolchain)
- Copy the library (Depends on the platform)

4.4 Create a new feature

Now it's the time to create a new feature branch, add your feature and push it back to the develop branch:

- git pull origin develop
- git checkout develop
- git merge feature-new-stuff
- git push
- git branch -d feature-new-stuff

4.5 Create a new release

There are a few things that have to be done for each new release:

- Create a new release branch
  - git checkout -b release-x.x.x develop
- Generate documentation

  - Run the doxygen documentation generation
  - Update this documentation and create a PDF
- Update codebase
  - Update src/cdevstudio/data/desktop/cdevstudio.desktop
  - Update src/cdevstudio/data/man/cdevstudio.1.gz
  - Update src/cdevstudio/data/text/about_about.html
  - Update src/cdevstudiocodeedit/CMakeLists.txt
  - Update src/cdevstudiosystemplatform/CMakeLists.txt
  - Update src/cdevstudioprojectplatform/CMakeLists.txt
  - Update src/cdevstudiobackend/CMakeLists.txt
- Update Debian package
  - Add new changelog entry
  - Update package/linux_debian_deb/create_package.sh
- Update Fedora package
  - Update package/linux_fedora_rpm/cdevstudio.spec
  - Update package/linux_fedora_rpm/create_package.sh
- Update Windows package
  - Update package/windows_exe/package.nsis
- Integrate the new release
  - git checkout master
  - git merge release-x.x.x
  - git push
  - git checkout develop
  - git merge release-x.x.x
  - git push
  - git branch -d release-x.x.x
- Create the new release number
  - git tag -a x.x.x -m "New release" master
  - git push –tags

4.6 Create a package

To create a package, run the create_package.sh or create_package.bat script in the package directory.