CDevStudio Documentation

Simon Wächter

April 13, 2014

Contents

1 Introduction

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

1.1 Idea and content 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. It also gives an overview about the design of the project. For the code documentation check out the code documentation generated by Doxygen

2 Documentation

As said in the introduction chapter, there are two documentations for the CDevStudio project.

2.1 Code documentation

The Doxygen documentation gives an overview about the whole 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 (The document you are reading) 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 main application (CDevStudio), a platform (CDevStudioPlatform) with a plugin API and the loaded plugins. These plugins have access to the API provided by the platform (CDevStudioPlatformPlugin).

3.1 CDevStudio

CDevStudio is the main program. It contains the graphical user interface and uses CDevStudioPlatform for UI creation, plugin loading and interaction.

3.2 CDevStudioPlatform

CDevStudioPlatform is the platform for the main application. It is responsible for loading plugins, for the plugin communication and the backend (I/O interaction). The plugin API is accessible over CDevStudioPlatformPlugin.

3.3 Plugin Projects

The Plugin 'Projects' provides some basic project templates.

3.4 Plugin Project Explorer

The Plugin 'Project Explorer' provides a basic project explorer view for the current project.

4 Workflow

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 (5.2.0 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 libraries to the application directory (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 checkout -b feature-new-stuff develop
- Do your stuff
- git commit -m "Feature"
- 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/about_about.html
  - Update src/pluginprojects/PluginProjects.cpp
  - Update src/pluginprojectexplorer/PluginProjectExplorer.cpp
  - Update src/CMakeLists.txt
- Update Debian package
  - Add new changelog entry
  - Update the Lintian overrides
  - 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 Arch Linux package
  - Update package/linux_archlinux_tarxz/PKGBUILD
- Update Windows package
  - Update package/windows_exe/cdevstudio.nsis
- Integrate the new release
  - git commit -m "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.