Manage Projects in Qt

Creating new Qt projects

After installing the integration, Eclipse's **New Project** dialog will contain a **Qt** folder with the following project templates:

- **Qt Console Project**: A basic Qt console application.
- **Qt Gui Project**: A basic Qt GUI application with one form.

The Console and Gui project wizards allow you to specify the Qt modules required by the project. The wizards will also generate a skeleton class that you can use to get started.

To be able to build the created project you have to tell Eclipse where to find Qt. This is explained in the next section.

Basic Qt version management

The Qt Eclipse integration offers its own simple Qt version management, enabling you to use multiple versions of Qt in parallel (e.g., Qt 4.3.0 and 4.4.0). To add or remove Qt versions, click **Window|Preferences**... and select the **Qt** page. Click **Add**, then enter a name for the Qt version and the paths to Qt's bin and include directory (e.g., "C:\Trolltech\Qt-4.4.0\bin" and "C:\Trolltech\Qt-4.4.0\include"). The **Default** button sets the currently selected Qt version as the default version, i.e. this Qt version will be used by default when a new project is created.

Note for the Windows platform: Please choose a Qt version that has been built with MinGW (e.g. the Open Source edition of Qt) since the CDT plugin which the Qt integration relies on supports debugging with the MinGW tools and not the Visual Studio tools. If you chose a Visual Studio based Qt version, you will only be able to build your application, but not to debug it. In order to nevertheless build with Visual Studio's NMake, be sure to start Eclipse with all the required Visual Studio related environment variables set.

Qt can be build on various platforms using different compilers. This means depending on the Qt version, Eclipse' build system has to set at least the **QMAKESPEC** environment variable and the make tool associated to it. By default this is done automatically, but if you want to control it manually, uncheck the corresponding checkbox on the lower end of the Qt page.

To specify that a project should use a particular Qt version, open the **Project|Properties** dialog and select the **Qt Properties** page. There you can select any Qt version added via the preferences dialog. If the Qt version points to the "<Default>" version, the Qt version marked as default in the preferences dialog is used. This means whenever the default version is changed, the project will automatically use the newly specified default version.

Using the Qt Eclipse Integration does not require a globally set Qt environment variable.

Importing projects

Eclipse can only build projects which are part of the current workspace. This means if you want to build an already existing project which is not part of the workspace, the project has to be imported first.

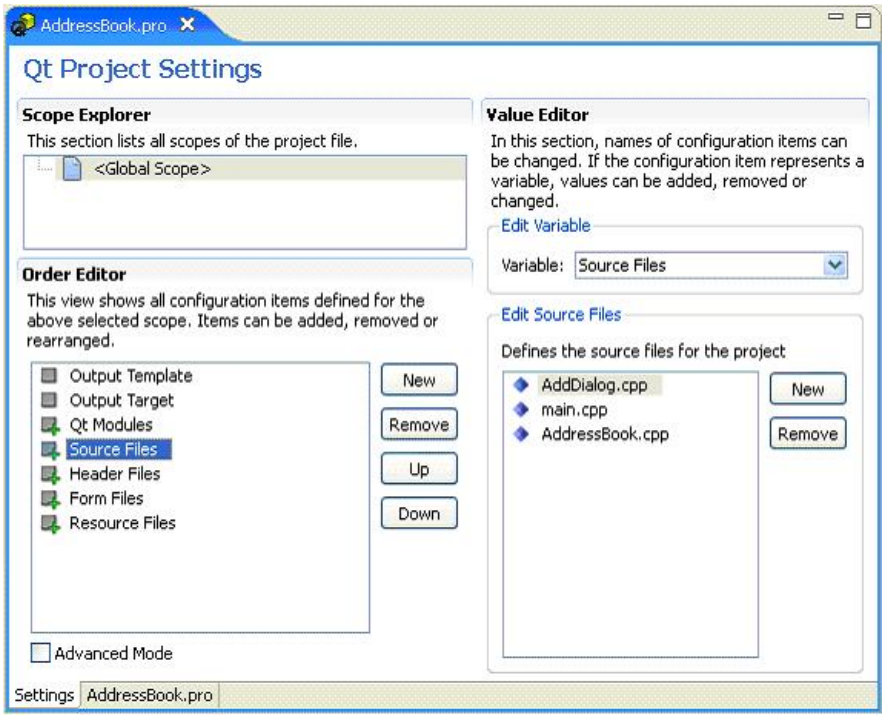
To do this, open the **File|Import**... dialog, select the **Qt|Qt Project** item and click **Next**. In the invoked Qt import wizard, specify the .pro file name, check the project name and click **Finish**. The imported project will use the default Qt version.

Project settings

The build process of Qt projects in Eclipse is based on .pro files like the normal command line build. This means that all changes relevant to the compiler or linker have to be made in the .pro file. To ease up this task, the Qt Eclipse integration offers a graphical .pro file editor.

.pro File Editor

(C) Copyright Nokia 2014. All rights reserved.



The .pro file editor consists of three parts:

- **Scope Explorer**: Lists all scopes which are directly defined in the .pro file or included via a .pri file. The scope explorer is a read only view.
- **Order Editor**: The order editor displays all items which are part of the selected scope in the explorer above. By clicking the **New** button, new scopes, variables or blocks can be added. Since the order of items in a .pro file is important, items can be reordered by clicking **Up or Down**.
- **Value Editor**: This part of the .pro file editor allows you to change the current item's name from the order editor. In case that the item is a variable, the value editor enables you to assign values to the variable.

Suppose you want to add a special header file for the compilation on X11 systems. To achieve that, select the global scope item in the scope explorer. The press the **New** button of the order editor and click the **Add Scope** item in the appearing menu. The scope explorer as well as the order editor will now show the newly added scope. By default the scope is called Mac, so we have to change it to X11. This is done by selecting the scope in the order editor and typing X11 in the **Edit Scope** line edit of the value editor.

Now we select the X11 scope in the scope explorer and add a new variable. The **Edit Variable** field in the value editor lists all available variables. When you select Header Files, the field below changes to **Edit Header Files**. If you click **New**, a new item will be added and highlighted for inplace editing. So, just type the file name and you're done.

If you're interested how the edited .pro file looks like, click on the tab right next to the Settings tab on the lower left corner of the editor.

As you may have noticed, the **Edit Variable** combo box offers you only a small subset of variables available in qmake. If you want to use other variables or remove values from certain variable (e.g. SOURCES = rkv.cpp) you have to use the **advanced mode** of the .pro file editor. To enable this mode check the Advanced Mode check box in the lower left corner of the editor. The usage of the .pro file editor when it is in advanced mode is pretty much equal to the editor in normal mode. One difference in view of the variable names is that the advanced view shows all variables by its real name instead of the alias names used in the normal mode.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.