Adding Qt Designer Plugins

You can use Qt APIs to create plugins that extend Qt applications. This enables you to add your own widgets to Qt Designer. The most flexible way to include a plugin with an application is to compile it into a dynamic library that is shipped separately, and detected and loaded at runtime.

The applications can detect plugins that are stored in the standard plugin subdirectories. For more information on how to create and locate plugins and to change the default plugin path, see How to Create Qt Plugins.

For more information about how to create plugins for Qt Designer, see Using Custom Widgets with Qt Designer.

Locating Qt Designer Plugins

Qt Designer fetches plugins from the standard locations and loads the plugins that match its build key. Qt Designer is delivered both as a standalone application and integrated into Qt Creator. The correct folder to place the plugins depends on whether you use the standalone Qt Designer or the integrated Qt Designer.

The integrated Qt Designer fetches plugins from the `\bin\plugins\designer` directory in the Qt Creator installation directory on Windows and Linux. For information about how to configure plugins on macOS, see Configuring Qt Designer Plugins on macOS.

To check which plugins were loaded successfully and which failed, choose **Tools** > **Form Editor** > **About Qt Designer Plugins**.

The standalone Qt Designer is part of the Qt library used for building projects, located in `<Qt_version>\<compiler>\bin` in the Qt installation directory. It fetches plugins from the `\plugins\designer` subdirectory of `bin`. To check which plugins were loaded successfully and which failed, choose **Help** > **About Plugins**.

Configuring Qt Designer Plugins on macOS

On macOS, a GUI application must be built and run from a bundle. A bundle is a directory structure that appears as a single entity when viewed in the Finder. A bundle for an application typcially contains the executable and all the resources it needs.

Qt Creator uses its own set of Qt Libraries located in the bundle, and therefore, you need to configure the Qt Designer plugins that you want to use with Qt Creator. For more information about how to deploy applications to macOS, see Qt for macOS - Deployment.

The following example illustrates how to configure version 5.2.1 of the Qwt - Qt Widgets for Technical Applications library for use with Qt Creator:

1. To check the paths used in the Qwt library, enter the following `otool` command:

```
otool -L /Developer/Applications/Qt/plugins/designer/libqwt_designer_plugin.dylib
```

The output for Qwt 5.2.1 indicates that the plugin uses Qt core libraries (QtDesigner, QtScript, QtXml, QtGui and QtCore) and libqwt.5.dylib:

```
/Developer/Applications/Qt/plugins/designer/libqwt_designer_plugin.dylib:
        libqwt_designer_plugin.dylib (compatibility version 0.0.0, current version 0.0.0)
        libqwt.5.dylib (compatibility version 5.2.0, current version 5.2.1)
        QtDesigner.framework/Versions/4/QtDesigner (compatibility version 4.6.0, current version 4.6.2)
        QtScript.framework/Versions/4/QtScript (compatibility version 4.6.0, current version 4.6.2)
        QtXml.framework/Versions/4/QtXml (compatibility version 4.6.0, current version 4.6.2)
        QtGui.framework/Versions/4/QtGui (compatibility version 4.6.0, current version 4.6.2)
```

```
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 125.0.1)
```

2. You must copy the Qt Designer plugin and the Qwt library files to the following locations:
   - `libqwt_designer_plugin.dylib` to `Qt Creator.app/Contents/PlugIns/designer`
   - `libqwt.*.dylib` to `Qt Creator.app/Contents/Frameworks`

   Enter the following commands:

```
sudo cp /Developer/Applications/Qt/plugins/designer/libqwt_designer_plugin.dylib \
        /Developer/Applications/Qt/Qt\ Creator.app/Contents/MacOS/designer
sudo cp -R /usr/local/qwt-5.2.1/lib/*  \
        /Developer/Applications/Qt/Qt\ Creator.app/Contents/Frameworks/
```

3. Enter the following `otool` command to check the libraries that are used by the Qwt library:

```
otool -L /usr/local/qwt-5.2.1/lib/libqwt.5.dylib
```

   The command returns the following output:

```
/usr/local/qwt-5.2.1/lib/libqwt.5.dylib:
        libqwt.5.dylib (compatibility version 5.2.0, current version 5.2.1)
        QtGui.framework/Versions/4/QtGui (compatibility version 4.6.0, current version 4.6.2)
        QtCore.framework/Versions/4/QtCore (compatibility version 4.6.0, current version 4.6.2)
        /usr/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version 7.9.0)
        /usr/lib/libgcc_s.1.dylib (compatibility version 1.0.0, current version 438.0.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 125.0.1)
```

4. Enter the following `install_name_tool` command to fix the references of the libraries:

```
cd /Developer/Applications/Qt/Qt\ Creator.app/Contents/MacOS/designer
sudo install_name_tool -change
    QtCore.framework/Versions/4/QtCore \
    @executable_path/../Frameworks/libQtCore.4.dylib \
    libqwt_designer_plugin.dylib
sudo install_name_tool -change QtGui.framework/Versions/4/QtGui \
    @executable_path/../Frameworks/libQtGui.4.dylib \
    libqwt_designer_plugin.dylib
sudo install_name_tool -change QtXml.framework/Versions/4/QtXml \
    @executable_path/../Frameworks/libQtXml.4.dylib \
    libqwt_designer_plugin.dylib
sudo install_name_tool -change QtScript.framework/Versions/4/QtScript \
    @executable_path/../Frameworks/libQtScript.4.dylib \
    libqwt_designer_plugin.dylib
sudo install_name_tool -change QtDesigner.framework/Versions/4/QtDesigner \
    @executable_path/../Frameworks/libQtDesigner.4.dylib \
    libqwt_designer_plugin.dylib
sudo install_name_tool -change libqwt.5.dylib \
    @executable_path/../Frameworks/libqwt.5.dylib \
    libqwt_designer_plugin.dylib

cd /Developer/Applications/Qt/Qt\ Creator.app/Contents/Frameworks
sudo install_name_tool -change \
    QtCore.framework/Versions/4/QtCore \
```

```
QtGui.framework/Versions/4/QtGui \
@executable_path/../Frameworks/libQtGui.4.dylib \
libqwt.5.2.1.dylib
```

Matching Build Keys

The Qt Creator that is included in pre-built Qt packages on Windows is built with the Microsoft Visual Studio compiler, whereas the version of Qt shipped for building applications is configured and built to use the MinGW/g++ compiler. Plugins built by using this version of Qt cannot be loaded by Qt Creator because the build-keys do not match. The plugins can only be used in the standalone version of Qt Designer. Choose **Help** > **About Qt Creator** to check the Qt version Qt Creator was built with.

To use Qt Designer plugins that were built for the shipped Qt version, make sure that Qt Creator is built with the same compiler by either recompiling Qt Creator using MinGW or recompiling Qt with Microsoft Visual Studio, depending on which configuration you want to use for your applications.

‹ Developing Widget Based Applications

Optimizing Applications for Mobile Devices ›