



<b>ID</b>	CS001425	<b>Creation date</b>	June 16, 2009
<b>Platform</b>	S60 3rd Edition, FP1, FP2 S60 5th Edition	<b>Tested on devices</b>	Nokia 5800 XpressMusic
<b>Category</b>	Qt for Symbian	<b>Subcategory</b>	Base/System

**Keywords (APIs, classes, methods, functions):** QPluginLoader, Q\_DECLARE\_INTERFACE, Q\_INTERFACES, Q\_EXPORT\_PLUGIN2

## Overview

Qt applications can be extended with Qt plug-ins. This code snippet demonstrates how to load and initialise plug-ins dynamically in your application using `QPluginLoader`.

Defining the plug-in interface and implementing the plug-in are described in additional snippet articles.

**Note:** In order to use this code, you need to have Qt for S60 installed on your platform.

## Important

- When using Qt for S60 libraries from the (qt\_libs\_armv5\_udeb.sisx) on a device, be sure to build both the plug-in and the loading application in **DEBUG(UDEB)** mode. This is because Qt uses a build key for each plug-in to ensure that only compatible plug-ins are loaded, and the build mode **[debug|release]** is part of that build key.
- In Symbian we have to define the variable `EPOCALLOWDLLDATA` as true for the plug-ins (library) because Qt macros have initialised global data; see the plug-in project files for more information.

## Preconditions

## Project (.pro) file

The DLL (plug-in) needs to have at least the same set of capabilities as the process (application) that loads the plug-in.

## Header (pluginwidget.h)

Needed headers, methods and variables.

```
// Qt plugin loader that loads plugins dynamically
#include <QPluginLoader>
#include <QtGui>
// Our defined plugin interface ExamplePluginInterface
#include "exampleplugininterface.h"

// Methods
bool loadPlugins(QString pluginDir);
void createPlugins();
void unloadPlugins();

private:
    // Array to store all plugins
    QList<QPluginLoader*> plugins;
```

## Source (yourapp.cpp)

Load the plug-ins.

```
bool YourApp::loadPlugins(QString pluginDir)
{
    QDir pluginsDir(QLibraryInfo::location
(QLibraryInfo::PluginsPath));

    // "exampleplugins" is the folder where you are exported plugins
    // by Qt macro Q_EXPORT_PLUGIN2(exampleplugins, YourPlugin);
    pluginsDir.cd("exampleplugins");

    foreach (QString fileName, pluginsDir.entryList(QDir::Files))
    {
        // Create plugin loader
        QPluginLoader* pluginLoader =
        new QPluginLoader(pluginsDir.absoluteFilePath(fileName));
        // Store loader to array
        plugins.append(pluginLoader);
        // Load plugin
        bool ret = pluginLoader->load();
        if (!ret)
        {
            QMessageBox::information(this, "YourApp",
            QString("Could not load plugin %1").arg(fileName));
        }
    }
}
```

```

// Did we found any plugins?
if (plugins.isEmpty())
    return false;
else
    return true;
}

```

### Create the plug-ins.

```

void YourApp::createPlugins()
{
    // Show data of all media plugins in different tabs
    for (int i=0 ; i<plugins.count() ; i++)
    {
        QPluginLoader* pluginLoader = plugins[i];
        // Create plugin instance
        QObject *plugin = pluginLoader->instance();
        if (plugin)
        {
            // Plugin instance created

            // Cast plugin to ExamplePluginInterface,
            // that is common for all plugins
            ExamplePluginInterface* pluginIF
            = qobject_cast<ExamplePluginInterface*>(plugin);

            // Signal / slot, if needed
            //QObject::connect(this, SIGNAL(send(QString *)),
            //                pluginIF, SLOT(someSlot(QString *)));
        }
        else
        {
            // Could not create plugin instance, delete pluginloader
            delete pluginLoader;
            plugins.removeAt(i);
            i--;
        }
    }
}

```

### Unload the plug-ins.

```

void YourApp::unloadPlugins()
{
    // Unload plugins and clear plugin array
    foreach (QPluginLoader* pluginLoader, plugins)
    {
        pluginLoader->unload();
        delete pluginLoader;
    }
    plugins.clear();
}

```

## See also

- [CS001390 - Defining a Qt plug-in interface](#)
- [CS001391 - Implementing the Qt plug-in interface](#)
- For more information about plug-ins, see [How to Create Qt Plugins](#).

## Postconditions

The Qt plug-in is loaded and initialised dynamically.