Defining a Qt plug-in interface

Overview

Qt applications can be extended with Qt plug-ins, which means that some of the application functionality is implemented as common plug-in libraries loaded into the application memory at runtime. To be extendable, the application has to define a common plug-in interface (`examplepluginterface.h`). Note that the application does not see the plug-ins directly but only methods and data that are defined into the interface. The detection and loading of plug-ins is handled using QPluginLoader.

This code snippet demonstrates how to define the lower-level Qt plug-in interface. Implementing the interface and using the plug-in are described in additional snippet articles.

Header (examplepluginterface.h)

The plug-in interface for all "example plug-ins". The plug-in must implement the `virtual sayHello()` method and the `virtual someSlot()` slot.

```
#include <QObject>
#include <QString>

class ExamplePluginInterface : public QObject
    {
    public:
        virtual ~ExamplePluginInterface() {}
        virtual QString sayHello() = 0;

    public slots:
        virtual void someSlot(QString value) = 0;

    protected: // Common data for the plugins
        QString data;
    };

// This Qt macro associates the given Identifier
// "Forum.Nokia.wiki.pluginsnippet.ExamplePluginInterface/1.0"
// to the interface class called ExamplePluginInterface.
// The Identifier must be unique.
Q_DECLARE_INTERFACE(ExamplePluginInterface,
    "Forum.Nokia.wiki.pluginsnippet.ExamplePluginInterface/1.0");
```

See also

- Implementing the Qt plug-in interface
- Loading and initialising a Qt plug-in dynamically
- For more information about, see How to Create Qt Plugins.

Postconditions

The Qt plug-in interface is defined.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.