# QT插件使用

## 1基本插件制作流程：

1. 定义接口。接口定义为抽象基类，如IPluginInterface，实现高度封装。定义的头文件在最后需要通过Q\_DECLARE\_INTERFACE来唯一标识该接口，即通过一个全局ID绑定接口。比如：

Q\_DECLARE\_INTERFACE(IPluginInterface, “com.blog.test.plugin.test/1.0”)

1. 实现接口。PluginInterfaceImp类通过继承自抽象基类IPluginInterface (同时继承QObject)，实现了接口方法。PluginInterfaceImp类需要申明两个东西：

|  |
| --- |
| Q\_OBJECT  Q\_INTERFACES(PlugInterface) |

这样能保证接口具备QT的基本功能机制，比如信号槽、接口识别等。

1. 最后，在适当的地方申明Q\_EXPORT\_PLUGIN2，其第二个参数是之后在加载该插件得到的对象的类名，比如PluginInterfaceImp。

## 2项目中应用

项目中使用QT的插件，一般会不止一个，而插件作为QT独特的接口库使用方法，可以由统一的使用框架进行管理，方便使用。基本框架图如下：



* PluginManager：负责插件初始化和加载。

其中、插件初始化会对指定目录下的文件通过QLibrary::resolve进行加载。这样就获得了一个插件模块对象列表，可以存放到一个map中，通过ID来索引，值为IPluginFactory指针表示的PluginFactory对象。

加载则是获取指定ID的PluginFactory对象，然后通过GetPluginObj()获得实际插件对象。

* IPluginFactory：插件对象统一对外接口，其派生类实例对象对应着某个插件，因此能在派生类对象中实现GetPluginID()和GetPluginObj()方法，得到最终的插件对象。

这样，制作好的插件放在项目制定文件夹后，就可以在项目中使用这些插件了：

1. 初始化PluginManager:: Instance().Init();
2. 加载插件LoadPlugin(id);