# 如何制作Genie插件

陶轶栋

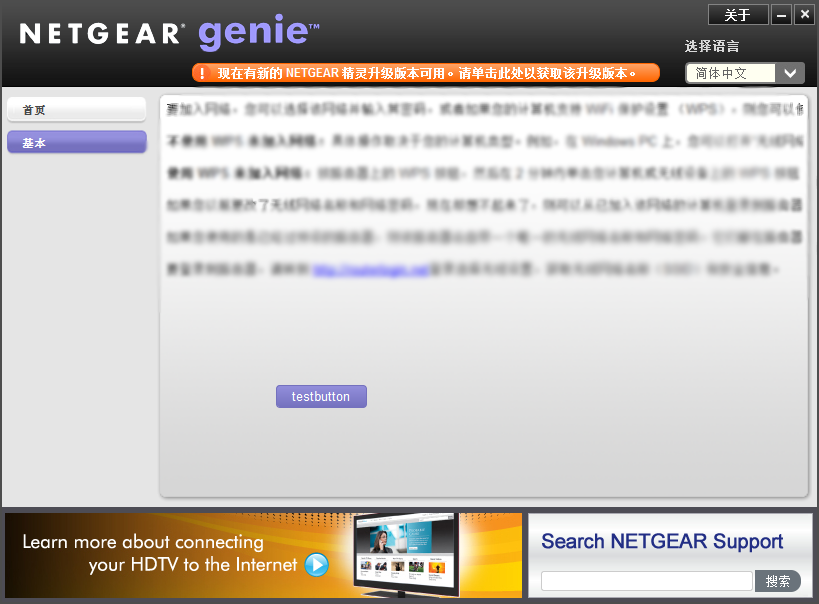
## 插件简介

Genie的插件利用了Qt提供的插件技术，我们自己定义了Genie插件的接口，只要插件实现了这个接口，并且在被Genie框架加载后，能够提供给插件一些必要的信息（如插件名称、版本号等等），就能在框架中正常的运行了。

每个插件在框架中对应一个大按钮和一个左边列表中的按钮，这两个按钮由框架来绘制，如下图所示



同时插件需要提供给框架一个QWidget实例, 插件加载的时候，将这个QWidget交给框架，之后每次当用户点击左边按钮或大按钮，框架就会跳转到这个QWidget所在的页面，如图所示



插件所提供的逻辑和功能都通过这个widget来展示。Genie框架将各种各样的插件集合到一起，并根据优先级排列之，于是Genie就变成了一个各种功能的集合。如图



目前这个集合主要用于NETGEAR路由器的管理，以后我们也可以集合一些别的插件来做别的用途。

## 开发环境

Qt，安装qt-sdk-win-opensource-2010.05.exe ，请在制作最终版本的时候，一定要用qt的这个版本来打包。另外，Qt的官网<http://qt.nokia.com> 上有支持VC的版本，用来调试代码会比QtCreator好用一点，如果喜欢，可以down下来做调试用，但如果插件要在框架上跑，则必须用与框架相同的qt版本来打包

Python31，用来执行打包的批处理文件，

## 开发示例简介

我们的开发示例程序分为slib和test两个部分，slib实现了Genie的接口，并最终生成一个静态链接库，test则用来实现插件自己特有的功能，test将slib链接进入最终的生成文件（即Genie插件）。Slib需要重新编译，因为它与你的编译环境有着密切的关系，比如，如果用vc2008编译test，而slib是用vc2005编译出来的，则链接过程不能成功。虽然最终生成的插件需要用Mingw版本的Qt（qt-sdk-win-opensource-2010.05.exe）来编译，但是你的调试过程中用这个版本就不太方便，你可以选择自己认为得心应手的编译环境来进行调试（我们自己使用vc+qt插件），重新编译slib是为了使之与你的编译环境一致。Slib中封装了插件自身的信息，以及插件与框架之间的信息传递过程，这些东西对于插件开发者来说，不需要太多的关注。

test文件夹是一个示例程序源文件。用户自定义插件时，可以选择直接从test工程进行修改。如果你想使用自己定义的文件名和类名，直接在test中改掉就行了（注意不要改动slib中代码）。

首先打开slib.pro工程文件,编译通过后会生成slib.lib或者slib.a 静态库。然后打开test.pro 工程文件，编程成功后就生成了一个GeniePlugin\_LibPlugin.exe 可执行文件，运行它后则可以看到一个窗口。这时可能会有些奇怪，为什么不是GeniePlugin\_LibPlugin.dll呢。其实生成exe文件是给插件自己做调试用的，如果要生成能够加载到Genie框架中的插件，则需要利用makeplugin.py这个脚本了。安装python31，然后用文本编辑器打开makeplugin.py，并根据本机环境，修改前面的这么几行（例如我们的qt-sdk-win-opensource-2010.05.exe安装到了d:\Qt\2010.05）：

QTDIR='d:/Qt/2010.05/qt'

Qtsdk的目录，默认位置在安装目录下的qt目录

MINGWDIR='d:/Qt/2010.05/mingw'

mingw目录，默认位置在安装目录下的mingw目录

MAKE=MINGWDIR+"/bin/mingw32-make.exe"

定义make命令路径，不需要改动

QMAKE=QTDIR+"/bin/qmake.exe"

qmake命令的路径

PLUGIN\_SOURCE='test'

你的插件所在的目录名称，我们的示例用的是test

PLUBIN\_PRO='testplugin.pro'

插件的工程文件的名称

SLIB\_SOURCE='slib'

Slib所在的目录

运行脚本makeplugin.py，则GeniePlugin\_LibPlugin.dll 就生成出来了，这时可以将其拷贝进入Genie安装目录中的bin目录，并重启Genie，就能看到应有的效果了。如果需要开发自己的插件，只需要在test.pro 这个工程当中进行改动即可，slib这个静态库由我们来提供，并且我们在每次升级Genie框架接口的时候，都会提供新的slib，这样插件只需要重新链接自己的程序即可，而不用费太多的精力去关注接口具体做了哪些改动。

## 开发示例源代码解析

我们之所以将插件分成这么两个部分，首先是为了方便第三方开发的需要，让其能够更加方便快捷的实现自己的插件；其次，我们的插件也在不断的改进和自我完善，我们每次升级框架和插件接口，都会提供新的slib，这样就可以避免框架升级给插件开发者带来的工作量了。当然，插件开发者在实现自己的插件的时候没有改动过slib中的东西（我们不鼓励这么做，虽然开放了这部分的源代码）

我们在slib中提供了两个类，QgenieBasePluginLib和QgenieBaseWidget，开发者需要做的仅仅是在自己的插件实现中派生这两个类。QgenieBasePluginLib为插件接口的基类，而QgenieBaseWidget 则用于派生出在框架中显示的窗口(QWidget)。

下面来讲解一下test部分的代码结构，QMyWidget.h QmyWidget.cpp QmyWidget.ui 这三个文件用来定义插件的QWidget，可以从Qt提供的Wizard来生成一个普通的QWidget，然后将他的基类从QWidget改为QgenieBaseWidget 就行了。开发者可以根据插件的功能来设计这个QgenieBaseWidget 的外观。



Testplugin.h testplugin.cpp 这两个文件从QgenieBasePluginLib 派生出了自己的插件类型

setUiEnable();

允许显示ui，这一行保留就行了

setVersion("2.2.21.2");

设置插件的版本号，关于版本号，下面有更详细的说明

setWidget(new QMyWidget);

初始化窗口，注意这里不用设置该窗口的父窗口，因为框架会将其设置上去，另外，插件卸载（也就是析构函数）中，请不要释放这个窗口，因为这个操作已经由框架完成了

setPluginName("1900");

设置在大按钮和左边按钮中显示的插件名字，1900表示多语言文件中对应的条目，如果不需要多语言，则直接讲文本设置在这里就行了，例如

setPluginName("TestPlugin");

前提是这个字符串不能被顺利的转变成数字

setPriority(100);

设置插件的优先级，优先级主要用于大按钮和左边按钮的排列顺序，与插件加载的顺序没有关系

setBigbuttonImgName("bbt\_demo");

设置大按钮中显示的图片名称，上面这一行将实际的资源路径设置成了":/bbticon/bbt\_demo.png" ，这里需要插件在自己的资源中提供bbt\_demo.png这个文件，参见rc/rc.qrc。另外，需要给图片取一个不容易与其他插件重复的名字，所以制作插件的时候，请不要沿用我的"bbt\_demo" 了

setLanFname("LanguageDemo.txt");

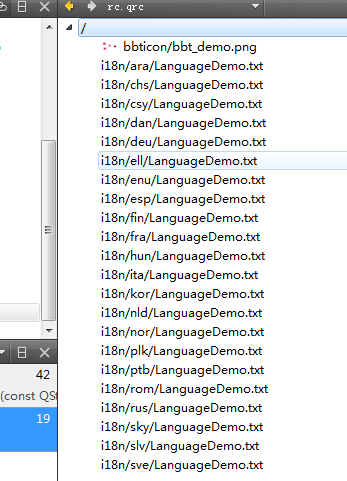
设置多语言文件的名称，与setBigbuttonImgName一样，你需要给自己的多语言文件取一个不容易与其它插件产生冲突的名字。关于多语言的制作，下面会有更详尽的说明

需要注意的是，这些设置都是在构造函数中完成的，而且必须是在构造函数中。因为QgenieBasePluginLib 会用这些信息与框架交互，如果当框架询问插件这些信息的时候，你还没有将其设置好，则会出问题了。

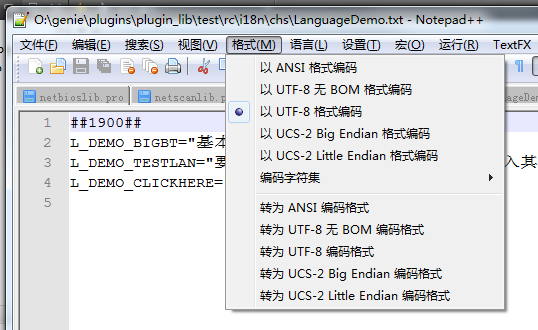
## 如何制作多语言

Genie的多语言机制利用了Qt提供的多语言机制，但是两者不完全相同。

首先，需要在资源文件中提供多语言文本文件，目录结构如下图所示



注意这里提到的目录结构，是在资源文件中的目录结构，它可能与本地磁盘上的目录结构不是完全对应。另外，多语言文件注意要用utf8编码



其次，你需要告诉框架，你的插件需要哪个多语言文件

setLanFname("LanguageDemo.txt");

再次，当框架的语言改变的时候，需要响应这个事件，并从框架重新获取多语言文本，重载 QWidget::changeEvent函数就行了，如

void QMyWidget::*changeEvent*(QEvent \*e)

{

QWidget::*changeEvent*(e);

if(e->type() ==QEvent::LanguageChange)

{

ui->label->setText(getI18nLan(1901));

QString strOk=getI18nLan(1902);

QString style=QString(BBT\_MODE\_CLICKHERE).arg(strOk);

setBigButtonStyle(style);

}

}

另外，需要注意一下阿拉伯语，这种语言的左右顺序是和其它语言反过来的，文字的反转你不用做额外的工作，但是布局的反转则需要在你的窗口中做一些工作，Qt的Layout能帮助你完成反转的功能，而反转之后界面有没有乱掉就不好说了，可能需要你调整一下

## 如何调试插件

直接利用 slib.pro 和testplugin.pro 两个工程文件编译项目，会生成一个可执行文件，运行则会显示自己的插件窗口，这个可执行程序与Genie的框架没有任何关系，可以单独用来调试，调试完成后执行makeplugin.py 则可以生成插件了。当然单独运行的UI效果可能与在框架中运行的不太一样，这个需要在做成插件后再做细致调整

在执行makeplugin.py之前，需要用文本编辑器打开它，修改它的几个全局变量

BUILDPATH="buildpath"

## 关于版本号

Genie框架和插件的版本号的格式为 xx.xx.xx.xx总格4个段，每段为一位或者两位的数字，如.2.2.20.45。插件的版本号不一定与框架的完全一致，但是必须保证前三段完全一样。例如框架版本2.2.20.3 可以正确的加载 2.2.20.45版本的插件, 但是不能正确加载2.2.19.2。所以，开发插件的时候，版本号的前三段需要与框架保持一致，而最后一段则留给插件开发者做自己的版本管理。

## 注意事项

我们当时在开发Genie框架和插件的时候，曾经遇到过许多问题，如果你没有开发过Genie的插件，则请先仔细阅读本文的这个部分

1. 关于程序路径，请注意一下Qt、python，以及存放工程文件的路径，其中不要出现中文字符或者空格，否则有可能导致编译出现问题
2. 关于镜像编译，QtCreator提供的功能，就是将编译生成的中间文件放到别的目录里面去，以保持工程目录的干净。使用这个，可能会出现明明改过的代码，最后编译出的程序似乎跟没改一个样。估计是Qt自身的原因。而每次用QtCreator新建或者导入工程的时候，默认都是开了这个选项的，建议将其关掉。将下图中shadow build 后面的钩钩去掉就行了
3. 关于增量编译，如果使用mingw版本的qt编译代码，则gcc的增量编译功能也可能出现问题，倒是明明改过的代码，但是做出程序来和没改一样。但是vc版的qt则没有这个问题。所以个人推荐安装vc版本的qt进行调试，然后在换到mingw版本的qt来生成正式的插件
4. 界面样式的冲突，Genie框架使用了Qt提供的qss技术，框架给整个application设置了一个stylesheet，所以有时候你设置的ui样式（如字体、label的背景，layout的边距、内部间距等）与这个stylesheet产生冲突，这时你设置的样式会被框架的所覆盖，解决的办法是，用stylesheet来设置你的ui样式，而不是用setFont，setPixmap之类的函数，这样你的stylesheet就会覆盖掉系统的stylesheet。当然，你的stylesheet是设置在你的窗口上的（框架的设置在整个application上面）。如果你对qss不是太了解，则可以参考qt提供的文档来学习一下。
5. 请不要在插件中使用qml。Qml是Qt提供的较新的一项技术，能让你更快的开发更绚丽的界面。但是出于性能的考虑，Qt建议我们只要使用一个qml的view。Genie框架中目前没有使用Qml，但以后可能会加上，而如果你的插件中也用到了这个，则到时候整个application中就出现了两个这种view了，这样会比较麻烦。
6. 关于uuid，uuid是插件的一个唯一标识（参考Qt的Quuid类），现在的Genie版本任何一个格式合法的uuid都能使用，但是以后框架可能会利用uuid对插件进行严格的控制，所以请使用开发示例的所提供的uuid（uuid.h）
7. 请使用自己的文件名，我提供的test工程中用到的生成文件名，类名，资源文件名，这些都是可以改的，即使你懒得去改它，请至少把几个容易与其它插件产生冲突的几个改了
   1. 生成文件名 默认是GeniePlugin\_LibPlugin.dll，请改成与插件功能想匹配的名字，如GeniePlugin\_Powerline.dll
   2. 多语言文件名，默认为LanguageDemo.txt ，请改得不易与其它插件产生冲突，如LanguagePowerLine.txt
   3. 大按钮图标，默认为bbt\_demo.png，也请改一下
8. 如果你做出的插件不能正确的加载进入Genie框架，可能的原因如下
   1. 检查你编译插件时使用的Qt版本是不是与框架所使用的Qt版本一致
   2. 插件的版本号是不是符合要求（前面的章节已经说过）
   3. 插件的文件名是否符合要求，文件名必须符合格式GeniePlugin\_<xxxxx>.dll 否则框架不会去尝试加载它
   4. 检查一下你的uuid，如果两个插件的uuid一样，则只能加载一个，或者你的把别人的挤出去了，或者你被别人的挤出去了
   5. 动态库依赖的问题，如果你的插件还依赖了其它的动态库，则需要确保运行环境中，这些库是齐全的
9. 如果有任何疑问，或者对我们的设计有好的建议，欢迎邮件至

[Yifei19@gmail.com](mailto:Yifei19@gmail.com)

[Yidong.tao@gmail.com](mailto:Yidong.tao@gmail.com)