<http://blog.csdn.net/jklfjsdj79hiofo/article/details/25972629>

1 Qt和Javascript交互的两种方式

2 实例分析

       2.1 前端——HTML和Javascript

       2.2 后台——Qt

3 小结

4 附录

1 Qt和Javascript交互的两种方式

        为了充分发挥Javascript的跨平台特性，Qt和JavaScript交互将不采用“Qt对Javascript公开对象引用”的方式（即通过QWebFrame的addToJavaScriptWindowObject方法向Javascript提供对象引用）。这种方式虽然使得Qt和JavaScript能直接的交互，但是Javascript端对于Qt端的依赖将影响Javascript发挥跨平台性。该方式调用原理如图1-1中左图所示：

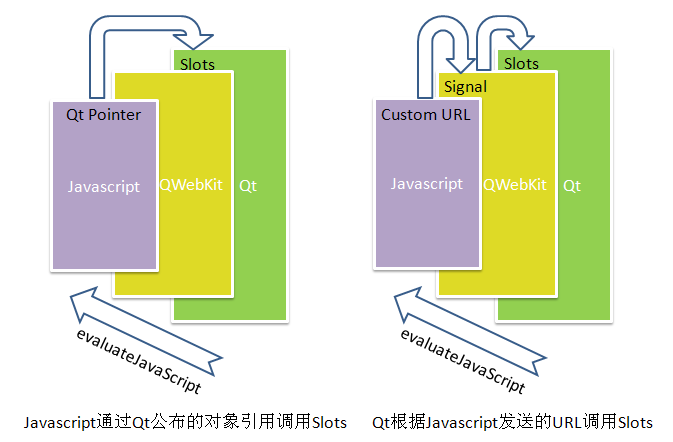


图1-1

        从图中不难看出，左边的方式中Javascript对Qt对象的依赖不利于Javascript往其它平台移植。

        而右边的“自定义URL”方式则不一样，Javascript不再选择利用Qt对象去调用Qt方法，而是直接向Webkit浏览器发送自定义URL，Webkit捕获到URL后通过QWebKit自身提供的Signal来关联Qt中的Slots函数，并向Slots函数传递URL，Slots函数就可以根据URL中的信息作出相应处理。在“自定义URL”方式中，底层平台不论是Qt，还是Android，上层的Javascript只需发送“自定义URL”即可。这样Javascript的平台无关性就能得以充分发挥。如图1-2所示：

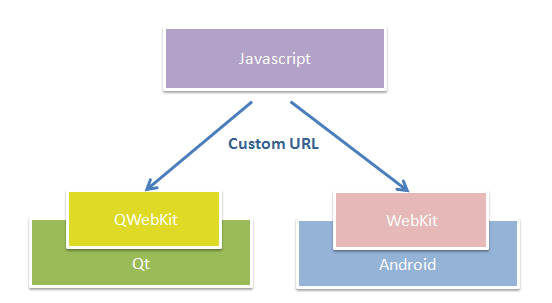


图1-2

2实例分析

         对“自定义URL”方式有了初步认识后，下面通过一个实例来叙述这种“自定义URL”的方式中可能涉及到的技术点。

2.1 前端——HTML和Javascript

         例子中涉及2个html文件a.html、b.html和一个Javascript文件js.js。文件b.html很简单，就是一句欢迎语，文件a.html运行如图2-1所示：



图2-1

其对应的代码如下：

<html>

<script language="JavaScript" type="text/javascript" src="js.js"></script>

<body>

         <center>

                            <span>text a:</span>

                            <input type="text" name="aText" id="aText" />

                            <input type="submit" value="Go" onclick="javascript:sendURL()" />

                            <a href="b.html">b.html</a>

                       <font color=red><div id=warning></div></font>

         </center>

</body>

</html>

该html文件中引用到的js.js代码如下：

function sendURL()

{

         var value = document.getElementById("aText").value.replace(/\s/g,"");

         if(value != "")

         {

                   if(confirm("Go to " + value +"?"))

                   {

                            location.href = value;

                   }

         }

         else

         {

                   document.getElementById("warning").innerHTML = "Please Input Target";

         }

}

2.2 后台——Qt

         从a.html和js.js端的代码不难看出，当用户在a.html页面上文本框内输入内容后，点击“Go”按钮就会触发js.js中的sendURL方法，该方法会把用户在文本框内输入的内容当作URL发送，那么后台的Qt端如何得到这个URL呢？

**2.2.1 loadStarted信号**

         通过查阅Qt帮助文档找到这样一个信号——loadStarted()，该信号在QWebView、QWebPage和QWebFrame均存在。当开始加载一个新的网页时产生该信号，所以接收Javascript产生的URL可以尝试从该信号入手。

**2.2.2 requestedUrl函数**

         QWebFrame有个方法叫做requestedUrl()。当产生URL请求而加载Web页面时，调用该方法能过获取到请求的URL。看来接收Javascript产生的URL，这个方法不可或缺。

**2.2.3 获取URL**

      下面结合前面选择的信号和方法来获取Javascript产生的URL。

1、为loadStarted信号设置Slots函数

QWebPage \* page = view->page();

connect(page, SIGNAL(loadStarted()), this, SLOT(doLoad()));

2、接收URL

void MyMainWindows::doLoad()

{

    QUrl url = view->page()->mainFrame()->requestedUrl();

    qDebug()<<"doLoad...."<<url;

    /\*解析URL\*/

    parseURL(url);

}

parseURL的具体实现参见附录。

**2.2.4 演示实例**

        接收成功后，就可以由Qt解析URL，由URL中的信息来选择应该执行什么操作。

        在本例子中，如果URL是字符串“print”，那么就调用“print”相关模块（简单的输出信息，然后跳转到b.html页面）。图2-2为实例运行后的主界面：



图2-2

        输入print，点击“Go”按钮，界面切换，如图2-3所示：

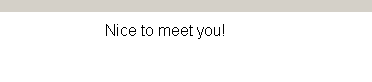


图2-3

        同时观察控制台输出，如图2-4所示：

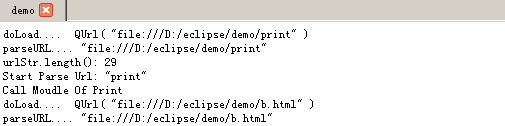


图2-4

        从输出信息中可以看出，接收到的URL为文件绝对路径，解析后比较URL，如果等于“print”，那么就执行“Call Moudle Of Print”，并加载b.html。

        注意，加载b.html页面时同样会因为产生loadStarted信号而执行预先设置好的Slots函数。至此演示完毕！

3小结

         在小结之前先来看a.html中埋下的一个小伏笔。

         在a.html中有个超链接“b.html”，该链接的URL指向b.html页面，正常情况下点击该链接就会跳转到b.html页面，但为何在本实例程序中点击b.html不会有任何跳转呢？

         这其中的原因在于QWebKit中的linkClicked信号，当点击Web页面中的超链接时就会产生该信号。本例中为该信号设置Slots函数，函数中没有对超链接的URL进行加载，所以点击超链接后不会产生任何跳转。类似于linkClicked这样的信号还有很多，可以根据不同的需求选择使用。

        对比两种交互方式，“Qt对Javascript公开对象引用”的方式在Javascript调用Qt的过程中表现得较为直接，而“自定义URL”的交互方式很好地解决了跨平台问题。总之，方案没有好坏之分，各有所长，根据实际需求选取即可。

4附录

1、parseURL方法具体实现如下：

void MyMainWindows::parseURL(QUrl url)

{

    QString urlStr = url.toString();

    qDebug()<<"parseURL...."<<urlStr;

    /\*

     \* 因为每次调用load都会触发loadStarted信号而执行该方法，

     \* 所以必须保证load网页时不会解析URL，否则load不到网页。

     \*/

    if(urlStr.right(5).operator != (QString(".html")))

    {

        int len = urlStr.length();

        qDebug()<<"urlStr.length():"<<len;

        QString ret = urlStr.right(len-24);

        qDebug()<<"Start Parse Url:"<<ret;

        if(ret.operator==(QString("print")));

        {

            qDebug()<<"Call Moudle Of Print";

            /\*如果找不到该url，app exited with code -1073741819\*/

            view->load(QUrl("../b.html"));

        }

    }

}

2、使用linkClicked信号前必须调用setLinkDelegationPolicy：

page->setLinkDelegationPolicy(QWebPage::DelegateAllLinks);

connect(page, SIGNAL(linkClicked(QUrl)), this, SLOT(parseURL(QUrl)));