Qt与Web混合开发技术

1. **适用场景**

Qt项目使用Web大量现成的组件/方案做功能扩展。

Qt项目中性能无关/频繁更新迭代的页面用html单独实现，Qt项目提供Web形式的SDK给用户做二次开发。

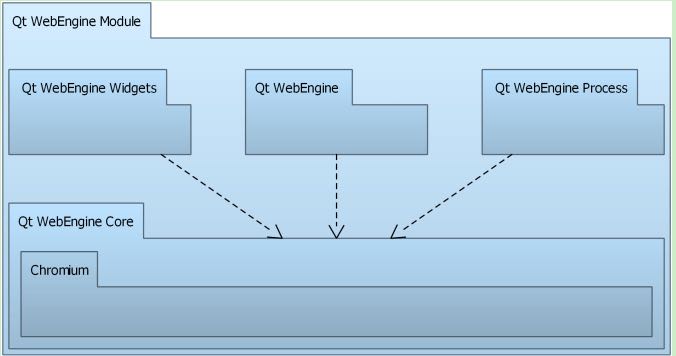
Web开发人员齐全而Qt/C++人手不足，此类非技术问题。

1. **WebEngine模块**

Qt提供了WebEngine模块以支持Web功能。Qt WebEngine基于google的开源浏览器chromium实现，类似的项目还有cef、miniblink等等。QtWebEngine可以看作是一个完整的chromium浏览器。

1. **WebEngine的架构**

QtWebEngine提供了C++和Qml的接口，可以在Widget/Qml中渲染HTML、XHTML、SVG，也支持CSS样式表和JavaScript脚本。QtWebEngine的架构图如下



基于Chromium封装了一个WebEngineCore模块，在此之上，WebEngine Widgets模块专门用于Widget项目，WebEngine 模块用于Qml项目，WebEngineProcess则是一个单独的进程，用来渲染页面、运行js脚本。Web在单独的进程里，我们开发的时候知道这一点就好了，不需要额外关注，只要在发布的时候，带上QTDIR目录下的可执行程序QtWebEngineProcess即可。底层使用了进程间共享OpenGL上下文的方式,实现多个进程的UI混合在一起。

1. **Windows平台渲染原理**

chromium在单独的进程处理html渲染，并将渲染结果存储在共享内存中；主窗口在需要重绘的时候，从共享内存中获取内容并渲染。

1. **通信WebChannel**

为了在不能影响各端代码执行的前提下实现Qt端于client 端的无缝双向通信。QWebChannel提供了在Server（C++应用）和client端（HTML/JS）之间点对点的通信能力。通过向client端的 QWebChannel 发布 QObject 的派生对象，进而实现在client端无缝读取来自Qt端的公共插槽 和QObject的属性值和方法。在整个通信过程中，无需任何手动序列化传入的参数。所有Qt端的属性更新，signal触发，都会自动且异步更新到client端。

1. **WebChannel流程**

WebChannel是在socket上建立的一种通信协议，这个协议的作用是把QObject暴露给远端的HTML。QObject 是 Qt 中对象模型的核心。该模型的核心特性是被称为 signal 和 slot 的对象通信机制。

大致使用流程：

Qt程序中，要暴露的QObject全部注册到WebChannel。

Qt程序中，启动一个WebSocketServer，等待Html的连接。

Html加载好qwebchannel.js文件, 然后去连接WebSocket。

连接建立以后，Qt程序中，由WebChannel接手这个WebSocket，按协议将QObject的各种“元数据”传输给远端Html。

Html端，qwebchannel.js处理WebSocket收到的各种“元数据”，用js的Object 动态创建出对应的QObject。

到这里两边算是做好了准备，可以互相调用了。

Qt端QObject数据变化只要发出信号，就会由WebChannel自动通知Web端；

Web端可以主动调用QObject的public的 invok函数、槽函数，以及读、写属性。