## 1.使用c++17标准

在.pro文件中添加：CONFIG +=c++17

## 2.Qt网络通信相关模块

|  |
| --- |
|  |

其中[Qt WebChannel](../qtwebchannel/qtwebchannel-index.html)可以实现一次调用(RPC)。Qt WebView一般用于安卓系统。[Qt WebEngine](../qtwebengine/qtwebengine-index.html)主要用于pc操作系统平台。

在qml在常用的web类型是**WebEngineView，需要**import QtWebEngine 1.10

主要需要在main.cpp的开头初始化QtWebEngine。需要#include #include <QtWebEngine/QtWebEngine>

|  |
| --- |
| QtWebEngine::initialize();//必须在创建应用程序实例之前初始化 |

实例

|  |  |
| --- | --- |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include <QtWebEngine/QtWebEngine>  int **main**(int **argc**, char \***argv**[])  {  QtWebEngine::initialize();  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtWebEngine 1.10  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml Web")  WebEngineView{  anchors.fill: *parent*  url:"https://wwww.facebook.com"  }  } |

缺点：运行非常的慢

WebEnginView也可以加载本地的html文件

如：

|  |  |  |  |
| --- | --- | --- | --- |
| <!—index.html -->  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>web engine test</title>      <link rel="stylesheet" href="style.css">  </head>  <body>      <p class="newFont">hello Qml Web</p>  </body>  <script src="./index.js"></script>  </html> | //index.js  //alert("hello from js!!!")  window.onload=function(){      alert("hello from js!!!")  } | | /\* style.css \*/  .newFont{      font-weight: bold;      font-style: italic;      color: brown;  } |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include <QtWebEngine/QtWebEngine>  int **main**(int **argc**, char \***argv**[])  {  QtWebEngine::initialize();  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtWebEngine 1.10  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml Web")  WebEngineView{  anchors.fill: *parent*  url:"https://wwww.facebook.com"  }  } | |

|  |  |
| --- | --- |
|  |  |

## 3.html与qml通信

实例，等到页面加载完毕后就运行js代码

|  |  |  |  |
| --- | --- | --- | --- |
| <!—index.html -->  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>web engine test</title>      <link rel="stylesheet" href="style.css">  </head>  <body>      <p class="newFont">hello Qml Web</p>  </body>  <script src="./index.js"></script>  </html> | //index.js  //alert("hello from js!!!")  window.onload=function(){      alert("hello from js!!!")  }  function add(num1,num2){      return num1 + num2;  } | | /\* style.css \*/  .newFont{      font-weight: bold;      font-style: italic;      color: brown;  } |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include <QtWebEngine/QtWebEngine>  int **main**(int **argc**, char \***argv**[])  {  QtWebEngine::initialize();  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtWebEngine 1.10  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml Web")  WebEngineView{  anchors.fill: *parent*  // url:"https://wwww.facebook.com"  url:"./index.html"  onLoadProgressChanged: {  if(*loadProgress* === 100){  let *strJs* = 'add(100,200)'  *runJavaScript*(*strJs*,function(result){  *console*.log(*result*)  })  }  }  }  } | |

以上这种方式是不推荐的，我们推荐使用WebChannel

## 4.使用WebChannel

Import Statement: import QtWebChannel 1.15

**注意：需要使用Qt WebChannel JavaScript API，通过WebEngine加载qrc:///qtwebchannel/qwebchannel.js文件来使用**

**文件的具体位置：**

|  |
| --- |
| **C:\Qt\Examples\Qt-5.15.2\webchannel\shared\** |

**发布项目的时候需要把这个文件拷贝到项目目录下，并且加入项目中，使用自己的url**

实例

|  |  |  |  |
| --- | --- | --- | --- |
| <!—index.html -->  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>web engine test</title>      <link rel="stylesheet" href="style.css">      <script src="qrc:///qtwebchannel/qwebchannel.js"></script>  </head>  <body>      <p class="newFont">hello Qml Web</p>  </body>  <script src="./index.js"></script>  </html> | //index.js  //alert("hello from js!!!")  window.onload=function(){      alert("hello from js!!!")  }  function add(num1,num2){      return num1 + num2;  }  new QWebChannel(qt.webChannelTransport, function(channel) {      //console.log(JSON.stringify(channel))      let qmlChannel = channel.objects.qmlChannel      qmlChannel.printFun("this is a message from QWebChannel...")  }) | | /\* style.css \*/  .newFont{      font-weight: bold;      font-style: italic;      color: brown;  } |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include <QtWebEngine/QtWebEngine>  #include <QtWebChannel/QtWebChannel>  int **main**(int **argc**, char \***argv**[])  {  QtWebEngine::initialize();  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtWebEngine 1.10  import QtWebChannel 1.15  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml Web")  WebEngineView{  anchors.fill: *parent*  // url:"https://wwww.facebook.com"  url:"./index.html"  QtObject{  id:*webEngineChannel*  WebChannel.id: "qmlChannel"  function *printFun*(value){  *console*.log(*value*)  }  }  webChannel: WebChannel{  registeredObjects: [*webEngineChannel*] //注册一个通道对象  }  }  } | |

## 5.WebChannel js api信号处理，方法一

|  |  |  |  |
| --- | --- | --- | --- |
| <!—index.html -->  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>web engine test</title>      <link rel="stylesheet" href="style.css">      <script src="qrc:///qtwebchannel/qwebchannel.js"></script>  </head>  <body>      <p class="newFont">hello Qml Web</p>      <button id="button1">click</button>  </body>  <script src="./index.js"></script>  </html> | //index.js  //alert("hello from js!!!")  window.onload=function(){      alert("hello from js!!!")  }  function add(num1,num2){      return num1 + num2;  }  new QWebChannel(qt.webChannelTransport, function(channel) {      //console.log(JSON.stringify(channel))      let qmlChannel = channel.objects.qmlChannel      qmlChannel.printFun("this is a message from QWebChannel...")//远程调用      let btn = document.getElementById("button1")      btn.onclick = function(){          let user = qmlChannel.getDatas() //返回的是一个Promise对象          user.then(function(data){              //console.log(JSON.stringify(data))              alert(JSON.stringify(data))          }).catch(function(err){              alert(err)        })      }  }) | | /\* style.css \*/  .newFont{      font-weight: bold;      font-style: italic;      color: brown;  } |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include <QtWebEngine/QtWebEngine>  #include <QtWebChannel/QtWebChannel>  int **main**(int **argc**, char \***argv**[])  {  QtWebEngine::initialize();  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtWebEngine 1.10  import QtWebChannel 1.15  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml Web")  WebEngineView{  anchors.fill: *parent*  // url:"https://wwww.facebook.com"  url:"./index.html"  QtObject{  id:*webEngineChannel*  WebChannel.id: "qmlChannel"  function *printFun*(value){  *console*.log(*value*)  }  function *getDatas*(){  let *data* = {name:"Mary",password:"12345"}  return *data*  }  }  webChannel: WebChannel{  registeredObjects: [*webEngineChannel*] //注册一个通道对象  }  }  } | |

## 6.WebChannel js api信号处理，方法二

|  |  |  |  |
| --- | --- | --- | --- |
| <!—index.html -->  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>web engine test</title>      <link rel="stylesheet" href="style.css">      <script src="qrc:///qtwebchannel/qwebchannel.js"></script>  </head>  <body>      <p class="newFont">hello Qml Web</p>      <button id="button1">click</button>  </body>  <script src="./index.js"></script>  </html> | //index.js  //alert("hello from js!!!")  //alert("hello from js!!!")  window.onload=function(){      alert("hello from js!!!")  }  function add(num1,num2){      return num1 + num2;  }  new QWebChannel(qt.webChannelTransport, function(channel) {      //console.log(JSON.stringify(channel))      let qmlChannel = channel.objects.qmlChannel      qmlChannel.printFun("this is a message from QWebChannel...")//远程调用      let btn = document.getElementById("button1")      let flag = false;      btn.onclick = function(){          if(!flag){           qmlChannel.userData.connect(function(user){              //alert(JSON.stringify(user))               console.log("name:"+user.name,"password:"+user.password)           })              flag = true         }          qmlChannel.getDatas()      }  }) | | /\* style.css \*/  .newFont{      font-weight: bold;      font-style: italic;      color: brown;  } |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include <QtWebEngine/QtWebEngine>  #include <QtWebChannel/QtWebChannel>  int **main**(int **argc**, char \***argv**[])  {  QtWebEngine::initialize();  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtWebEngine 1.10  import QtWebChannel 1.15  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml Web")  WebEngineView{  anchors.fill: *parent*  // url:"https://wwww.facebook.com"  url:"./index.html"  QtObject{  id:*webEngineChannel*  signal userData(var user);  WebChannel.id: "qmlChannel"  function *printFun*(value){  *console*.log(*value*)  }  function *getDatas*(){  let *data* = {name:"Mary",password:"12345"}  //return data  *userData*(*data*);  }  }  webChannel: WebChannel{  registeredObjects: [*webEngineChannel*] //注册一个通道对象  }  }  } | |

## 7.使用WebView

#### Import Statement: import QtWebView 1.15

**注意：WebView的用法和WebEngineView差不多但是，它是没有WebChannel的，需要自己手动创建**

**需要使用到WebSocketServer：import QtWebSockets 1.15**

**另外如果想实现转发功能需要自己实现QWebChannelAbstractTransport类，#include <QWebChannelAbstractTransport>**

**在项目我们.pro中添加：**QT += webchannel

**新建一个类从QwebChannelAbstractTransport派生，需要重写**void ***sendMessage***(const QJsonObject &**message**) override;方法和

添加一个void **textMessageReceived**(const QString &**message**);方法

需要添加一个信号：void **messageChanged**(QString **msg**);

还需要添加一个WebChannel实例来注册对象

实例

|  |  |  |  |
| --- | --- | --- | --- |
| <!—index.html -->  **<!DOCTYPE** html**>**  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>QML WebViw Demo</title>  <link rel="stylesheet" href="./style.css">  <script src="qrc:///qtwebchannel/qwebchannel.js"></script>  </head>  <body>  <p class="newFont">Qml WebViw Demo</p>  <button id="button1">Click</button>  </body>  <script src="./index.js"></script>  </html> | //index.js  //创建WebSocket  let socket =new WebSocket("ws://127.0.0.1:52222")  socket.onopen = function(){      console.log("the socket is open....")      new QWebChannel(socket, function(channel) {          //console.log(JSON.stringify(channel))          let qmlChannel = channel.objects.qmlChannel          qmlChannel.printFun("qml custom socket...")//远程调用      })  } | | /\* style.css \*/  .newFont{      font-weight: bold;      font-style: italic;      color: brown;  } |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include <QtWebEngine/QtWebEngine>  #include "webchanneltransport.h"  int **main**(int **argc**, char \***argv**[])  {  QtWebEngine::initialize();  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  qmlRegisterType<WebChannelTransport>("WebModule",1,0,"WebChannelTransport");  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtWebEngine 1.1  import QtWebView 1.15  import QtWebSockets 1.15  import QtWebChannel 1.15  import WebModule 1.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml WebView Demo")  WebChannelTransport{  id:*webChTransport*  }  WebView{  id:*webview*  anchors.fill: *parent*  url:"./index.html"  WebSocketServer{ //这是一个服务器，需要在js端创建一个WebSocket来和它通信  id:*webSckSrv*  listen: true//打开监听  port:52222 //设置端口号  onClientConnected: { //clientConnected()信号的槽函数  // console.log("client connected...")  if(*webSocket*.status === WebSocket.Open){  *wChannel*.connectTo(*webChTransport*)  *webSocket*.onTextMessageReceived.connect(*webChTransport*.textMessageReceived)  *webChTransport*.onMessageChanged.connect(*webSocket*.sendTextMessage)  }  }  }  WebChannel{  id:*wChannel*  registeredObjects: [  QtObject{  WebChannel.id: "qmlChannel"  function *printFun*(value){  *console*.log(*value*)  }  }  ]  }  }  } | |
| //webchanneltransport.h  #ifndef WEBCHANNELTRANSPORT\_H  #define **WEBCHANNELTRANSPORT\_H**  #include <QWebChannelAbstractTransport>  #include<QtCore>  class **WebChannelTransport**:public QWebChannelAbstractTransport  {  Q\_OBJECT  public:  // WebChannelTransport();  using QWebChannelAbstractTransport::QWebChannelAbstractTransport;  public slots:  void ***sendMessage***(const QJsonObject &**message**) override;  void **textMessageReceived**(const QString &**message**);  signals:  void **messageChanged**(QString **msg**);  };  #endif // WEBCHANNELTRANSPORT\_H | | //webchanneltransport.cpp  #include "webchanneltransport.h"  #include<QDebug>  void WebChannelTransport::***sendMessage***(const QJsonObject &**message**)  {  qDebug()<< \_\_FUNCTION\_\_<<" "<<message;  QJsonDocument **doc**(message);  emit messageChanged(doc.toJson());  }  void WebChannelTransport::**textMessageReceived**(const QString &**message**)  {  qDebug()<< \_\_FUNCTION\_\_<<" "<<message;  auto **msgJson** = QJsonDocument::fromJson(message.toUtf8()).object();  emit messageReceived(msgJson,this);  } | |