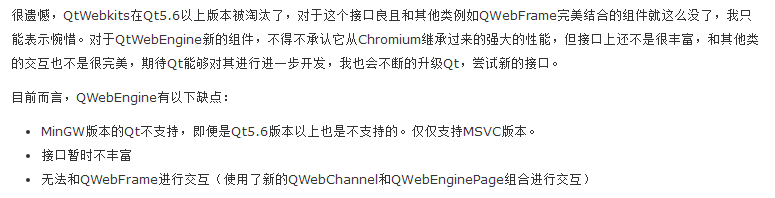
<https://www.cnblogs.com/liushui-sky/p/7850759.html>

当前项目为c/s客户端，采用qt4.8.7，需要使用仪表盘、折线图、柱状图等，曾经使用过qwt和自定义的图形控件，但是都不尽如人意。最近发现ECharts控件不错。为此就要在qt端使用web的技术。为此使用了QWebview的控件。关于它的使用网上有很多，一开始也没有深究，借鉴了前人的经验立即就使用了，而且也能正常使用。当时主要使用view->page()->mainFrame()->evaluateJavaScript这种方式。使用的形式是将需要显示的数据由qt主程序读取数据库，将返回的数据组成json格式的字符串传递给js执行。但是由于定时刷新频率较快1~5s，数据量较大（光数据就有172800~34560个）。所以导致程序内存增长很快。这是始料未及的，原以为qt的浏览器控件应该会像浏览器一样有垃圾回收，但是无奈，貌似没有或者不起作用，找了很久也没有相关资料。网上很多说早期的QWebKit存在内存泄漏、不完美、兼容性不好等诸多问题，看来是的了，毕竟早期qt团队貌似只是简单的应用，融合的不是特别好，尤其是4.8版本，好像5以后稍微好点（这些也就是网上道听途说的，是否为真，读者自判，o(\*￣︶￣\*)o）。

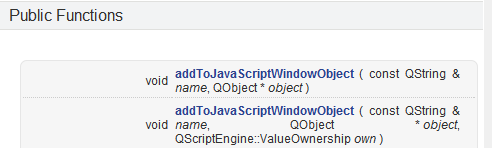
于是乎就要解决这个内存持续增长的问题，网上既然如前面说QWebKit不是很好，qt5.6以后改为了QtWebEngine，本想忍痛升级吧，结果看到下图



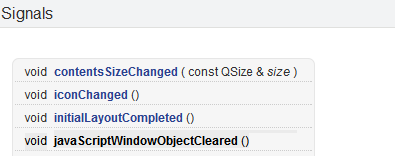
居然mingw版不支持，好吧，这不改动太大了，还要装msvc版，算了，另寻它图吧。于是又回到QWebKit/QWebview ，看有没有好的方式。

在于做网页的同事交流后，试过各种的办法（有考虑是否可以加入定期垃圾回收，貌似不可行，在对QWebKit/QWebview运行机制不了解的情况下作罢，关于qt的书实在太少了，关于有介绍QWebKit/QWebview的书就更没有了）。

最终还是继续查看qt的帮助文档（类的介绍英文版），查到QWebFrame类中还有下面的函数（作用：把窗体对象给js用来调用。具体看help）



进而，查到void javaScriptWindowObjectCleared ()函数（作用：This signal is emitted whenever the global window object of the JavaScript environment is cleared, e.g., before starting a new load.）



以上的函数合起来给我们的思路就是让js来主动调用qt程序的对象。而evaluateJavaScript的模式是qt调用js。刚好两者有点类似互逆的形式。

这个网上有资料，搜索QT和JavaScript互调就会有资料，随便列出两份，读者可以自己看。

http://blog.csdn.net/b711183612/article/details/50593068

http://blog.csdn.net/allenjiao/article/details/44963131

以下是我项目中的应用，简单列点代码。

构造函数中：

[复制代码](javascript:void(0);)

1 connect(ui->webLine->page()->mainFrame(), SIGNAL(javaScriptWindowObjectCleared()),

2 this, SLOT(sltJavaScriptFromWinObject()));

3 ui->webLine->load(QUrl("file:///"+qApp->applicationDirPath()+"/Echart/page/commLineChart.html"));

4

5 ui->webLine->settings()->setObjectCacheCapacities(0,0,0);

6

7 connect(ui->webLine,SIGNAL(loadFinished(bool)),this,SLOT(sltInitial()));

[复制代码](javascript:void(0);)

槽函数：

void 类名::sltJavaScriptFromWinObject()

{

ui->webLine->page()->mainFrame()->addToJavaScriptWindowObject("mywebkit",this);

}

js将要调用的函数（重点注意：这个函数必须是公共槽函数public slots:，注：其实根据其他文章说明“方法前需要Q\_INVOKABLE修饰”，具体见参考。此外这里又有两种方式，1、getDataLine是由qt获得数据最后让js来直接获取数据字符串。2、其它两个函数则是直接让js调用获取数据的函数。最后证明内存的消耗是差不多的。但是1的使用需要js定时调用，qt主程序也要定期刷新数据（频率要比前者高），而2则直接在js中设置刷新频率即可，故在消耗差不多的前提下最终选择2方法）

1 public slots:

2 QString TestrefreshRTData();

3 QString TestrefreshLine();

4 QString getDataLine();

TestrefreshLine函数（包括cjson的使用）：

[复制代码](javascript:void(0);)

1 QString 类名::TestrefreshLine()

2 {

3 float fMinVol=qAbs(mpDataNHour.values().first().value(KEY\_VOL));

4 float fMaxVol=qAbs(mpDataNHour.values().first().value(KEY\_VOL));

5 float fMinCur=qAbs(mpDataNHour.values().first().value(KEY\_CUR));

6 float fMaxCur=qAbs(mpDataNHour.values().first().value(KEY\_CUR));

7 //先创建空对象

8 cJSON \*json = cJSON\_CreateObject();

9 cJSON \*array=NULL;

10 cJSON\_AddItemToObject(json,"commLineData",array=cJSON\_CreateArray());

11 QMap<QString,MP\_VALUE>::iterator it;

12 for ( it = mpDataNHour.begin(); it != mpDataNHour.end(); ++it )

13 {

14 QString datetime=it.key();

15 float vol=qAbs(it.value().value(KEY\_VOL));

16 float cur=qAbs(it.value().value(KEY\_CUR));

17 fMinVol=fMinVol>vol?vol:fMinVol;

18 fMaxVol=fMaxVol<vol?vol:fMaxVol;

19 fMinCur=fMinCur>cur?cur:fMinCur;

20 fMaxCur=fMaxCur<cur?cur:fMaxCur;

21 cJSON \*obj=cJSON\_CreateObject();

22 cJSON\_AddItemToArray(array,obj);

23 cJSON\_AddStringToObject(obj,"date",datetime.toStdString().c\_str());

24 cJSON\_AddStringToObject(obj,"vol",QString("%1").arg(vol).toStdString().c\_str());

25 cJSON\_AddStringToObject(obj,"cur",QString("%1").arg(cur).toStdString().c\_str());

26 }

27

28 fMinCur=fMinCur\*(1-PROPORTION\_CUR);

29 fMaxCur=fMaxCur\*(1+PROPORTION\_CUR);

30 fMinVol=fMinVol\*(1-PROPORTION\_VOL);

31 fMaxVol=fMaxVol\*(1+PROPORTION\_VOL);

32 cJSON\_AddStringToObject(json,"minCur",QString::number(fMinCur, 'f', DECIMAL\_CUR).toStdString().c\_str());

33 cJSON\_AddStringToObject(json,"maxCur",QString::number(fMaxCur, 'f', DECIMAL\_CUR).toStdString().c\_str());

34 cJSON\_AddStringToObject(json,"minVol",QString::number(fMinVol, 'f', DECIMAL\_VOL).toStdString().c\_str());

35 cJSON\_AddStringToObject(json,"maxVol",QString::number(fMaxVol, 'f', DECIMAL\_VOL).toStdString().c\_str());

36

37 char \*out=cJSON\_Print(json); // json对象转字符串

38

39 cJSON\_Minify(out); // 去掉字符串中的换行和缩进

40 QString sz = QString(out);

41 free(out);//注意：这个千万别忘记，网上的很多人的资料把这个忘记了，造成内存泄漏

42 cJSON\_Delete(json);

43

44 /\*\*\*\*\*\*\*\*\*qt调用js，内存持续增长\*\*\*\*\*\*\*\*\*\*/

45 // sz.replace(QRegExp("\""), "\\\"");

46 // QString szValue = QString("optionData(\"%1\");").arg(sz);

47 // ui->webLine->page()->mainFrame()->evaluateJavaScript(szValue);

48

49 /\*\*\*\*\*\*\*\*\*\*\*js调用qt对象\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

50 QString szValue = QString("%1").arg(sz);

51 return szValue;

52 // return szValue.toStdString();//不用转换成string,QString可识别。

53

54 }

[复制代码](javascript:void(0);)

js:

[复制代码](javascript:void(0);)

1 //------------------------------------折线图-------------------------------------------------->

2 $("#clc").css("width",window.innerWidth).css("height",window.innerHeight);

3

4 //var lineChart=echarts.getInstanceByDom(document.getElementById("clc"));

5 //echarts.dispose(lineChart);

6 var lineChart = echarts.init(document.getElementById("clc"));

7 var xData=[];

8 var dyData=[];

9 var dlData=[];

10 var maxVol=10;

11 var minVol=0;

12 var maxCur=10;

13 var minCur=0;

14 var timeSplit=5;//时间间隔 每5秒执行一次

15

16 var lineOption = {

17 legend: {

18 data:["单位:V","单位:A"],

19 x:"center",

20 y:"top",

21 textStyle:{

22 color:"#fff"

23 },

24 formatter: function (name) {

25 if(name=="单位:V"){

26 return "电压";

27 }else{

28 return "电流";

29 }

30 }

31 },

32 grid:{

33 x:45,

34 x2:45,

35 y:45,

36 y2:45

37 },

38 tooltip: {

39 trigger: 'axis',

40 formatter: function (params,ticket,callback) {

41 if(params.length==2){

42 return "时间="+params[0].data[0]+"<br/><span style='color:#1e96f4;'>电压值="+params[0].data[1]+"</span>" +

43 "<br/><span style='color:#fd4171;'>电流值="+params[1].data[1]+"</span>";

44 }else {

45 if(params[0].seriesName=="单位:V"){

46 return "时间="+params[0].data[0]+"<br/><span style='color:#1e96f4;'>电压值="+params[0].data[1]+"</span>";

47 }else{

48 return "时间="+params[0].data[0]+"<br/><span style='color:#fd4171;'>电流值="+params[0].data[1]+"</span>";

49 }

50 }

51

52

53 },

54 axisPointer: {

55 type: 'line'

56 }

57 },

58 xAxis: [

59 {

60 type: 'time',//#8296a2

61 show:true,

62 boundaryGap: false,

63 splitLine: {

64 show: false

65 },

66 axisLine:{

67 lineStyle:{

68 color:"#fff"

69 }

70 },

71 axisTick:{

72 alignWithLabel:true

73 },

74 splitNumber:30,

75 min:0,

76 max:0

77 //data:xData

78 }

79 ],

80 yAxis: [

81 {

82 name:"单位:V",

83 nameTextStyle:{

84 color:"#fff"

85 },

86 type: 'value',

87 splitLine: {

88 show: true,

89 lineStyle:{

90 color:"#5c6f85"

91 }

92 },

93 position:"left",

94 axisLine:{

95 lineStyle:{

96 color:"#fff"

97 }

98 },

99 min:minVol,

100 max:maxVol

101 },

102 {

103 name:"单位:A",

104 nameTextStyle:{

105 color:"#fff"

106 },

107 nameLocation:"end",

108 type: 'value',

109 position:"right",

110 boundaryGap: [0, '100%'],

111 splitLine: {

112 show: false,

113 },

114 position:"right",

115 axisLine:{

116 lineStyle:{

117 color:"#fff"

118 }

119 },

120 min:minCur,

121 max:maxCur

122 }

123 ],

124 series: [

125 {

126 name:"单位:V",

127 type: 'line',

128 smooth: true,

129 showSymbol: false,

130 hoverAnimation: false,

131 itemStyle:{

132 normal:{

133 color:"#1e96f4",

134 lineStyle: {

135 width: 1,

136 }

137 }

138 },

139 data: dyData

140

141 } ,

142 {

143 name:"单位:A",

144 yAxisIndex:1,

145 type: 'line',

146 smooth: true,

147 showSymbol: false,

148 hoverAnimation: false,

149 itemStyle:{

150 normal:{

151 color:"#fd4171",

152 lineStyle: {

153 width: 1,

154 }

155 }

156 },

157 data: dlData

158 }

159

160 ]

161 };

162 //lineChart.setOption(lineOption);

163

164 function optionData(obj){

165 var json=$.parseJSON(obj.toString());

166 minVol=parseFloat(json.minVol);

167 maxVol=parseFloat(json.maxVol);

168 minCur=parseFloat(json.minCur);

169 maxCur=parseFloat(json.maxCur);

170 var startDate=json.startDate;

171 var endDate=json.endDate;

172

173 dyData=[];

174 dlData=[];

175 datas=json.commLineData;

176 for(var i=0;i<datas.length;i++){

177 var vol=datas[i].vol;

178 var cur=datas[i].cur;

179 var dateTime=datas[i].date;

180 var volArr=[dateTime,vol];

181 var curArr=[dateTime,cur];

182 dyData.push(volArr);

183 dlData.push(curArr);

184 }

185

186 if(datas.length==0){

187

188 lineOption.xAxis[0].show=false;

189 }

190 lineOption.xAxis[0].min=startDate;

191 lineOption.xAxis[0].max=endDate;

192 //lineOption.xAxis[0].data=xData;

193 lineOption.yAxis[0].min=minVol;

194 lineOption.yAxis[0].max=maxVol;

195 lineOption.yAxis[1].min=minCur;

196 lineOption.yAxis[1].max=maxCur;

197 lineOption.series[0].data=dyData;

198 lineOption.series[1].data=dlData;

199 //lineChart.setOption(lineOption);

200 lineChart.setOption(lineOption);

201 }

202

203

204 function reset(){

205 lineOption.series[0].data=[];

206 lineOption.series[1].data=[];

207 lineChart.setOption(lineOption);

208 }

209

210

211 function setBorder(width,height){

212 $("#clc").css("width",width+"px").css("height",height+"px");

213 $(lineChart).resize();

214 }

215

216

217 /\*\*\*\*\*js调用qt对象，调试时可以加入try...catch...捕获错误\*\*\*/

218 $(function(){

219 var jsonObj=window.mywebkit.refreshRTData();

220 optionData(obj);

221 setInterval(function(){

222 jsonObj=window.mywebkit.refreshRTData();

223 optionData(obj);

224 },timeSplit\*1000);

225 });

226

227 /\*$(function(){

228 try{

229 // alert("start");

230 setInterval(function(){

231 try{

232 var obj=window.mywebkit.getDataLine();

233 // alert(obj);

234 optionData(obj);

235 }

236 catch(e)

237 {

238 alert("Error\_2!");

239 }

240 },5000);

241 // alert("ok");

242 }

243 catch(e) {

244 alert("Error\_1!");

245 }

246 });\*/

[复制代码](javascript:void(0);)

最后分别列出qt调用js、js调用qt对象的内存比较数据。

 qt调用js内存情况如下：

[复制代码](javascript:void(0);)

2017-11-7

107268KB 13:10 （开始运行）

200852KB 15:33

102888KB 2017-11-7 17:31（对远程QT客户端重启后开始运行）

138712KB 2017-11-8 08:37

451120KB 2017-11-8 17:08

538248KB 2017-11-9 09:56

连续运行后，内存持续增长425.16M

[复制代码](javascript:void(0);)

js调用qt对象内存情况（启动后会有一段时间的增长（10分钟内），包括鼠标在图形上操作会增长到60M多，之后稳定，没有持续增长，成功）：

[复制代码](javascript:void(0);)

1方法：

55600KB 2017-11-16 16:57

61056KB 2017-11-17 08:11 稳定在这个范围

2方法：

46588KB 2017-11-16 17:07

61064KB 2017-11-17 08:11 稳定在这个范围  
  
参考：http://www.cnblogs.com/liushui-sky/p/7851654.html

[复制代码](javascript:void(0);)

标签: [Qt学习之路](http://www.cnblogs.com/liushui-sky/tag/Qt%E5%AD%A6%E4%B9%A0%E4%B9%8B%E8%B7%AF/), [QWebKit](http://www.cnblogs.com/liushui-sky/tag/QWebKit/), [good](http://www.cnblogs.com/liushui-sky/tag/good/)