Creating GUI applications for Qt and WP7 using HTML

This article presents an approach to develop mobile HTML based applications that can be executed in different platforms such as Qt and WP7.

Introduction

21 Aug 2011

Most of modern platforms offer APIs to render and execute web based content into one application. That can be achieved using webkit/browser components/objects, provided by different platforms, such as Symbian (using Qt) and Windows Phone 7.

These APIs also offers mechanisms that enable the communication between the two worlds:

- Web/HTML using Javascript and
- Native environment (Qt C++ or WP7 C#)

That way, you can develop a unique GUI using HTML+, CSS3 and JavaScript, and just develop specific and critical code for each native platform (e.g. a more complex algorithm running in a separated thread for C++ and C#)

Overview

Advantages

- Using the same GUI on different platforms
  - Just have to develop one GUI for all platforms
  - Can make use of new features supported by HTML5 and CSS3
- Efficient and critical code can be implemented in native code

Disadvantages

- It is not possible to use UI features from the native UI
  - Such as WP7 Metro UI layout
- HTML based UI can have performance bottlenecks

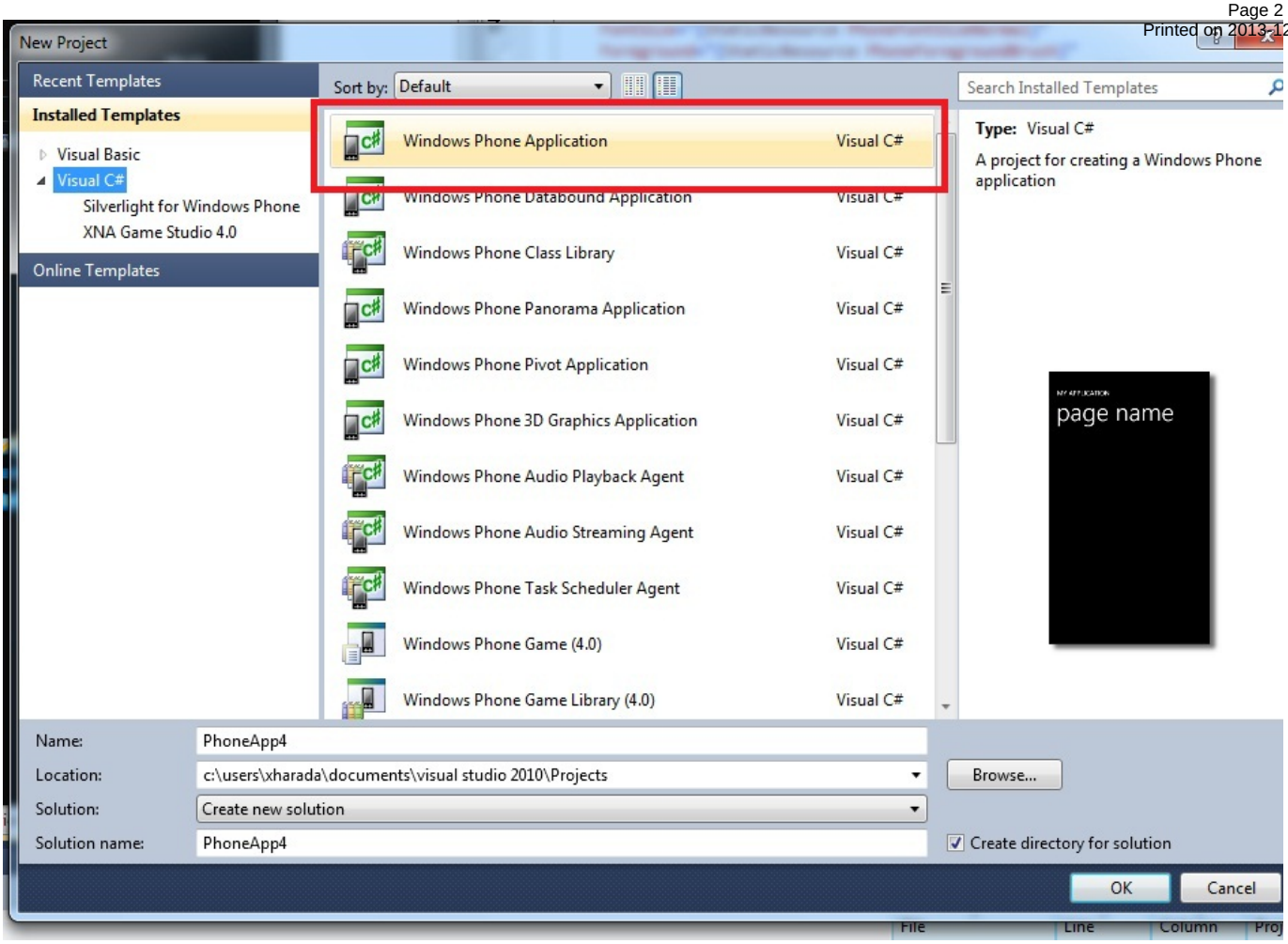
Creating HTML based application for WP7

For Windows Phone 7 we are going to use the WebBrowser class.

Step 1 - Creating WP App

Create a simple application using Microsoft Visual Studio 2010 Express for Windows Phone. You can call it WebTest.

(C) Copyright Nokia 2013. All rights reserved.



Step 2 - Adding WebBrowser Component

Add a WebView component in your application and enable Script support and Notify callback.

- MainPage.xaml

```
<Grid x:Name="LayoutRoot" Background="Transparent">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*"/>
  </Grid.RowDefinitions>
  <!--TitlePanel contains the name of the application and page title-->
  <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
     <TextBlock x:Name="ApplicationTitle" Text="WebTest Application" Style="{StaticResource PhoneTextNormalStyle}"/>
  </StackPanel>
  <!--ContentPanel - place additional content here-->
  <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
     <phone:WebBrowser HorizontalAlignment="Left" Name="webBrowser1" VerticalAlignment="Top" Height="690" Width="456" IsScriptEnabl
  </Grid>
</Grid>
```

Step 3 - Creating HTML based GUI

Create your HTML application + Javascript. You can use this HTML and JavaScript as Example.

- index.html:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=400, user-scalable=no, initial-scale=2.0, minimum-scale=1.0, maximum-scale=1.0" />
<link rel="stylesheet" type="text/css" href="web_test.css" />
<SCRIPT LANGUAGE="JavaScript" src="script.js">
</SCRIPT>

<title>Web Application</title>
</head>
<body>
<h1 id='mytitle'>Web Application</h1>
<form>
<input id="button" type="button" class="button-standard" style="width:200;height:100" value="Find Platform" onClick="testing()"/>
</form>
</body>
</html>
```

- web_test.css

(C) Copyright Nokia 2013. All rights reserved.

```
body {
background-color: Black;
color:White;
}

.button-standard   {
width: 200px;
height: 100px;
}
```

- script.js

```
function testing() {
    window.external.Notify("platformName");
}

function setting(val) {
    document.getElementById('mytitle').innerHTML = val;
}
```

This HTML application will change the mytitle element every time the user presses the button. The flow should be the following:

1. Users presses button;
2. Button event call function `testing()` from JavaScript;
3. Function `testing()` calls native call of Windows Phone 7;
4. Native code execute what it is necessary, and calls function setting(val) of JS;
5. Function `setting(val)` changes the `mytitle` element of HTML.

On Windows Phone 7, when you want to notify the native application using JavaScript, you should use the method: `window.external.Notify();`

In our example, everytime the JavaScript calls the notify method, the following method: `webBrowser_callback()` – look at XML file defined before.

Step 4 - Loading HTML content using WP7 App

Now, you need to load your HTML application using Windows Phone 7 C# code. If you want to load the code using a remote URL, it is just necessary to load the URL in the WebBrowser object:

- MainPage.xaml.cs

```
webBrowser1.Navigate(new Uri("http://www.developer.nokia.com", UriKind.RelativeOrAbsolute));
```

However, in our example we want to load the application locally. For that it is necessary to setup the Storage for your application. First it is necessary to create a Directory on your Storage. The following method will do that for you:

```
private void CreateDirOnStore(string strDir)
{
    IsolatedStorageFile isoStore = IsolatedStorageFile.GetUserStoreForApplication();
    if   (!isoStore.DirectoryExists(strDir))
          isoStore.CreateDirectory(strDir);
}
```

Then, you can add each file in the Storage using the following method:

```
private void SaveFileToIsoStore(string strFileName)
{
     IsolatedStorageFile isoStore = IsolatedStorageFile.GetUserStoreForApplication();
     //remove the file if exists to allow each run to independently write to
     // the Isolated Storage
     if   (isoStore.FileExists(strFileName) == true)
     {
         isoStore.DeleteFile(strFileName);
     }
     StreamResourceInfo sr = Application.GetResourceStream(new Uri(strFileName,   UriKind.Relative));
     using   (BinaryReader br = new BinaryReader(sr.Stream))
     {
         byte[]   data = br.ReadBytes((int)sr.Stream.Length);
         //save file to Isolated Storage
          using   (BinaryWriter bw = new BinaryWriter(isoStore.CreateFile(strFileName)))
         {
              bw.Write(data);
              bw.Close();
         }
     }
}
```

After this, it is just necessary to create the directory and add the files in your constructor:

- MainPage.xaml.cs

```
namespace WebTest {
       public partial class MainPage : PhoneApplicationPage {
               public MainPage()
               {
                   InitializeComponent();
                   this.Loaded   += new RoutedEventHandler(Load_WebPage);
               }

               void Load_WebPage(object sender, RoutedEventArgs e)
               {
                   //Set the start folder of the widget and navigate to the start page
                   CreateDirOnStore("web");
                   SaveFileToIsoStore("web/index.html");
                   SaveFileToIsoStore("web/web_test.css");
```


(C) Copyright Nokia 2013. All rights reserved.

```
                SaveFileToIsoStore("web/script.js");

                webBrowser1.Base = "web";
                webBrowser1.Navigate(new Uri("index.html", UriKind.Relative));
            }

        ….
```

Step 5 - Implementing Communication between domains

Implement the Notify callback in your C# code:

- Mainpage.xaml.cs

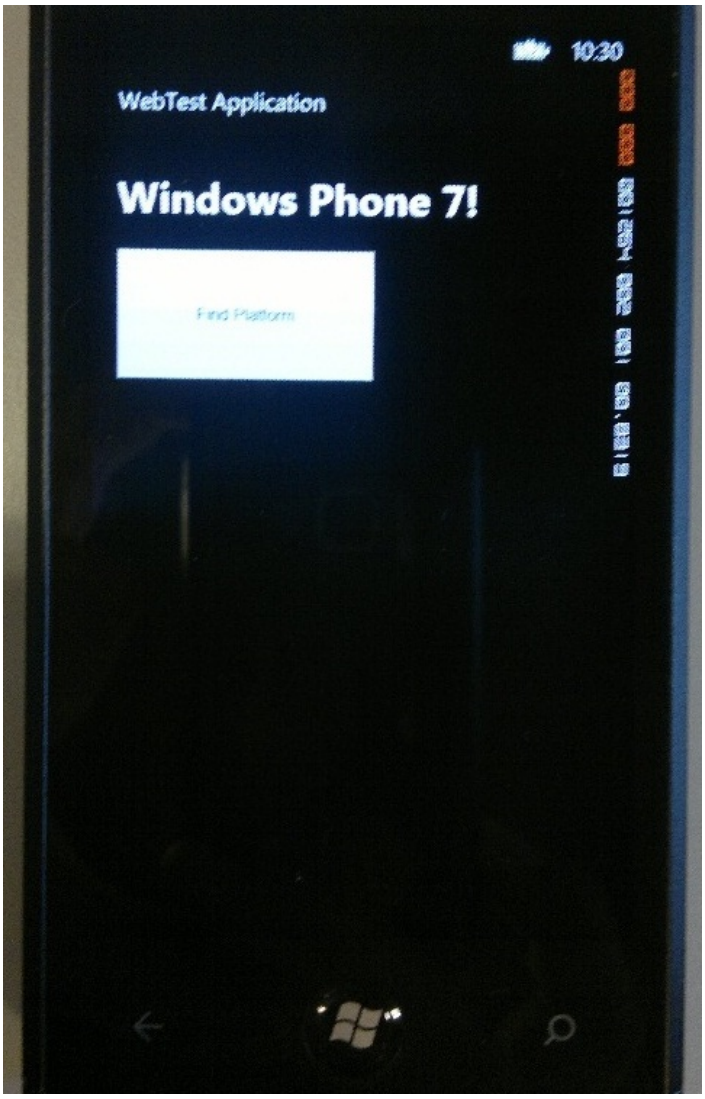
```
...
        void webBrowser_callback(object sender, NotifyEventArgs e)
        {
            //Here, you can do a swicth/case
            //Do more stuff, and use invokeScript to return to the web environment
            webBrowser1.InvokeScript("eval", new string[] { "setting('Windows Phone 7!')"});
        }
...
```

This callback just returns to the JS setting method the name of the platform using the InvokeScript method from WebBrowser class.

Step 6 - Executing application

Now, you can execute your application and see the result.

Following a picture of the application running on a real Windows Phone 7.



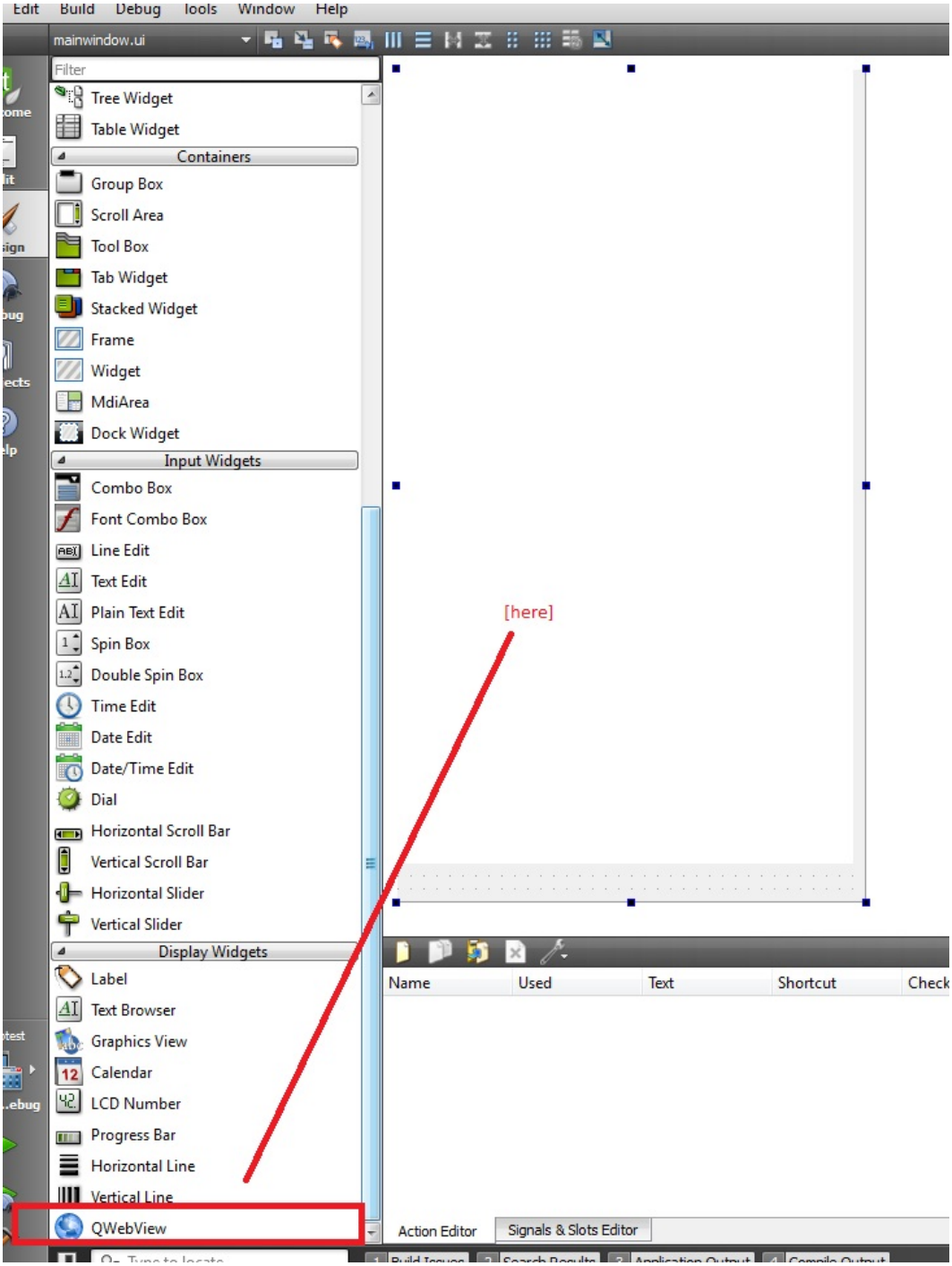
Creating HTML based application for QtWebkit

For Qt application we are going to use the QWebView class [1]

Step 1 - Creating Qt Application with QWebView

(C) Copyright Nokia 2013. All rights reserved.

Create your project in Qt Creator for mobile platforms. Use the Qt Designer to add your `QWebView` component in the UI, as showed in the following figure.



Configure your `QWebView` with the size that is better for your purpose.

Step 2 - Preparing Communication Object

Create an object that will be loaded in the JavaScript to be used as your bridge between Native and JavaScript code. As we want to use the same interface as the one provided by WP7 application, we create a `QObject` with a method using the same name as WP7 application: `Notify(QString)`.

- jsobject.cpp

```
#include "jsobject.h"
JSObject::JSObject(QObject *parent) :
    QObject(parent)
{
}

void JSObject::Notify(QString str)
{
    //Here, you can do a swicth/case
    //Do more stuff, and use invokeScript to return to the web environment
    emit setting("setting('Qt Platform')");
}
```


(C) Copyright Nokia 2013. All rights reserved.

Step 3 - Loading HTML application UI

Now, we should load our QObject in the HTML based application, and create a method to call/evaluate script in the JS domain.

First, load our `QObject` in JavaScript domain in the `addJavaScriptObject` SLOT. Then, in the SLOT implementation, you should define the same name for JS Window Object as the one used by WP7: "external". Now, the JSObject will receive the calls made to `Window.external` object in the JS domain.

Then, you should define the return path: Native C++ -> JS Script. This could be achieve calling the `evaluateScript()` method from `QWebView`, as showed in `callScript(QString)` SLOT

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    jsobject = new JSObject(this);
    ui->setupUi(this);
    //load our java script object in the webview object
    connect(ui->webView->page()->mainFrame(),
            SIGNAL(javaScriptWindowObjectCleared()),
            this,
            SLOT(addJavaScriptObject()));
    connect(jsobject, SIGNAL(setting(QString)), this, SLOT(callScript(QString)));

    //loading html file
    ui->webView->setUrl(QUrl("qrc:///index.html"));
    ui->webView->showMaximized();
}
MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::addJavaScriptObject()
{
//insert the object in JS domain using the same format as WP7
    this->ui->webView->page()
                      ->mainFrame()
                      ->addToJavaScriptWindowObject("external",
                                                    this->jsobject);
}

void MainWindow::callScript(QString script)
{
    this->ui->webView->page()->mainFrame()->evaluateJavaScript(script);
}
```

At the end, it is necessary to add the same HTML, CSS and JavaScript files in the resource file of your Qt application (qrc). Then, at the end of the MainWindow constructor the *index.html* file should be loaded in `QWebView` object.

Step 4 - Executing Application

Execute the application, and see that the interface looks the same as the one loaded by WP7, however, its returns a different string.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.