Windows Phone 8 - Using native pages

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/6.3/adding-native-functionality/windows-phone-8-using-native-pages-hybrid-applications.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

In this tutorial, integration of native and web "pages" in a Windows Phone 8 application will be explained by using the WL.NativePage.show() API to open a native page from JavaScript. With this method, data can be sent from JavaScript to the opened native page, and specify a callback to call after the native page closes.

Connecting to the plugin from the JavaScript code

As a first step, |WL.NativePage.show() | needs to be implemented in order to open the native page:

```
function openNativePage(){
  var params = {
    nameParam : $('#nameInput').val()
  };
  WL.NativePage.show(nativePageClassName, backFromNativePage, params)
;
}
```

- nativePageClassName: The name of a native Windows Phone 8 UIViewController instance to start.
- backFromNativePage: A callback function to call when the native page closes.
- params: optional custom parameters object to pass to the native code.

To handle the callback function:

```
function backFromNativePage(data){
  alert("Received phone number is: " + data.phoneNumber);
}
```

• backFromNativePage(data): After the native closes, it can pass data back to the web part of an application.

Creating a native page

For WP8, the native page must be implemented as a Windows Phone User Control, or extend an existing one.

Step 1

In the new User Control, add using Cordova.Extension.Commands; to the .cs file, and use the same package and class name as in the WL.NativePage API call in the JavaScript:

```
using Cordova.Extension.Commands;
using Newtonsoft.Json;

namespace NativePagesInHybridApp
{
   public partial class HelloNative : UserControl
   {
```

Step 2

To retrieve custom data parameters that are passed from the web view, the WLNativePage.Data method should be used. In the below example, the data is sent as a JSON string. For this purpose, the external JSON.NET library is used to convert the incoming JSON string to a native dictionary. For more information, see http://json.codeplex.com/

(http://json.codeplex.com/)

```
InitializeComponent();
  if (WLNativePage.Data != null) {
     Dictionary<string, string> data = JsonConvert.DeserializeObject<Dictionary<string, string>>(WLNativePage.Dat
a);
     NameReceivedTextBlock.Text = "Hello " + data["nameParam"];
     tb_returnValue.Text = "1234567890";
}
```

Returning control to the web view

When the native page needs to switch back to the web view, it calls the WLNativePage.backFromNative method. Data can be passed back to the web view as parameters to the call:

C#:

```
void DoneButton_Click(object sender, RoutedEventArgs e) {
    Dictionary<string, string> data = new Dictionary<string, string> {
        { "phoneNumber", tb_returnValue.Text }
    };

string json = JsonConvert.SerializeObject(data, Formatting.Indented);
    WLNativePage.backFromNative(this, json);
}
```

JavaScript:

```
function backFromNativePage(data){
   alert("Received phone number is: " + data.phoneNumber)
;
}
```

Sample application

Click to download

(http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/UsingNativePagesInHybridAppsProject.zip) the Studio project.

