Windows Phone 8 - Using native pages

Overview

This tutorial explains how to create and integrate native and web "pages" in a Windows Phone 8 application by using the `WL.NativePage.show()` API to open a native page from JavaScript.

With this method, you can have data sent from JavaScript to the open native page and specify a callback to be called after the native page closes.

This tutorial covers the following topics:

- Connecting to the plugin from the JavaScript code
- Creating a native page
- Returning control to the web view
- Sample application

Connecting to the plugin from the JavaScript code

1. Implement the `WL.NativePage.show()` to open the native page.

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

    - `nativePageClassName`: The name of a native WP8 class to start.
    - `backFromNativePage`: A callback function to call when the native page closes.
    - `params`: An optional custom parameters object to pass to the native code.
2. To handle the callback function:

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

    The `backFromNativePage(data)` parameter can pass data back to the web part of an application after the native closes.

Creating a native page

For Windows Phone 8, implement the native page 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, use the `WLNativePage.Data:(NSDictionary*)data` method.

In the example below, 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 the JSON.NET (http://json.codeplex.com/) page.

```
InitializeComponent();
        if (WLNativePage.Data != null) {
            Dictionary<string, string> data = JsonConvert.DeserializeObject<Dictionary<string, string>>(WLNativeP
age.Data);
            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.

Objective-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/v700/UsingNativePagesInHybridAppsProject.zip) the Studio project.