Windows Phone 8 - Implementing Cordova plug-ins

Overview

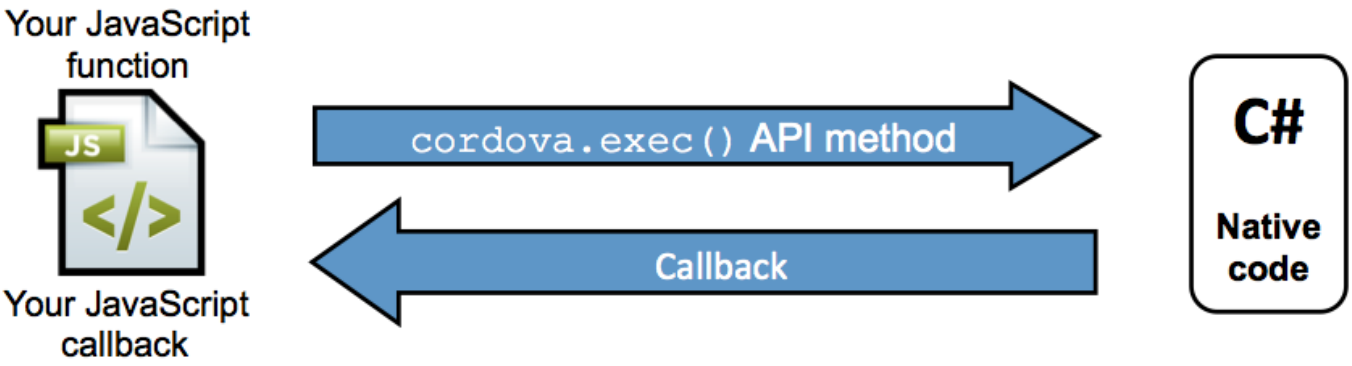
In some cases, developers of a MobileFirst application might have to use a specific third-party native library or a device function that is not yet available in Apache Cordova. With Apache Cordova, developers can create an Apache Cordova plug-in, which means that they create custom native code blocks, and call these code blocks in their applications by using JavaScript. In this tutorial, a simple Apache Cordova plug-in creation and integration for Windows Phone 8 will be demonstrated.

Note:

In Cordova-based applications, developers must check for the `deviceready` event before they use the Cordova API set. In a MobileFirst application, however, this check is done internally. Instead of implementing this check, implementation code can be placed in the `wlCommonInit()` function in `common\js\main.js`. The below code blocks are based on the sample application, provided at the bottom of this tutorial.

Plug-in creation overview:

- Declare the plug-in in the config.xml file
- Use the `cordova.exec()` API in the JavaScript code
- Create the plug-in class that will run natively in Windows Phone 8
- The plug-in performs the required action and calls a JavaScript callback method that is specified during the call to `cordova.exec()`



Declaring a plug-in

The plug-in needs to be declared in the project, so that Cordova can detect it. To declare the plug-in, add a reference to the config.xml file, located in the native folder of the Windows Phone 8 environment.

```
<feature name="sayHelloPlugin">
 <param name="wp-package" value="sayHelloPlugin" />
</feature>
```

Implementing cordova.exec() in JavaScript

From the JavaScript code of the application, use `cordova.exec()` to call the Cordova plug-in:

```
function sayHello() {
    var name = $("#NameInput").val();
    cordova.exe(sayHelloSuccess, sayHelloFailure, "SayHelloPlugin", "sayHello", [name])
;
}
```

`sayHelloSuccess` - Success callback `sayHelloFailure` - Failure callback `SayHelloPlugin` - Plug-in name as declared in config.xml `sayHello` - Action name `[name]` - Parameters array The plug-in calls the `success` and `failure` callbacks.

```
function sayHelloSuccess(data){
    WL.SimpleDialog.show(
        "Response from plug-in", data,
 [{text: "OK", handler: function() {WL.Logger.debug("Ok button pressed");}}]
    );
}

function sayHelloFailure(data){
    WL.SimpleDialog.show(
 "Response from plug-in", data,
 [{text: "OK", handler: function() {WL.Logger.debug("Ok button pressed");}}]
    );
}
```

Implementing the C# code of a Cordova plug-in

After the plug-in is declared, and the JavaScript implementation is ready, the Cordova plug-in can be implemented. For this purpose, ensure that the project is built in Eclipse and opened in the Visual Studio IDE.

Step 1

- Create a new C# class
- Add the new class to your project namespace and add the required import statements.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using WPCordovaClassLib.Cordova;
using WPCordovaClassLib.Cordova.Commands;
using WPCordovaClassLib.Cordova.JSON;

namespace Cordova.Extension.Commands
{
    public class SayHelloPlugin : BaseCommand
    {
```

Step 2

Implement the `SayHelloPlugin` class and the `sayHello` method.

- The JavaScript wrapper calls the sayHello method and passes a single parameter. It returns a string back to JavaScript.

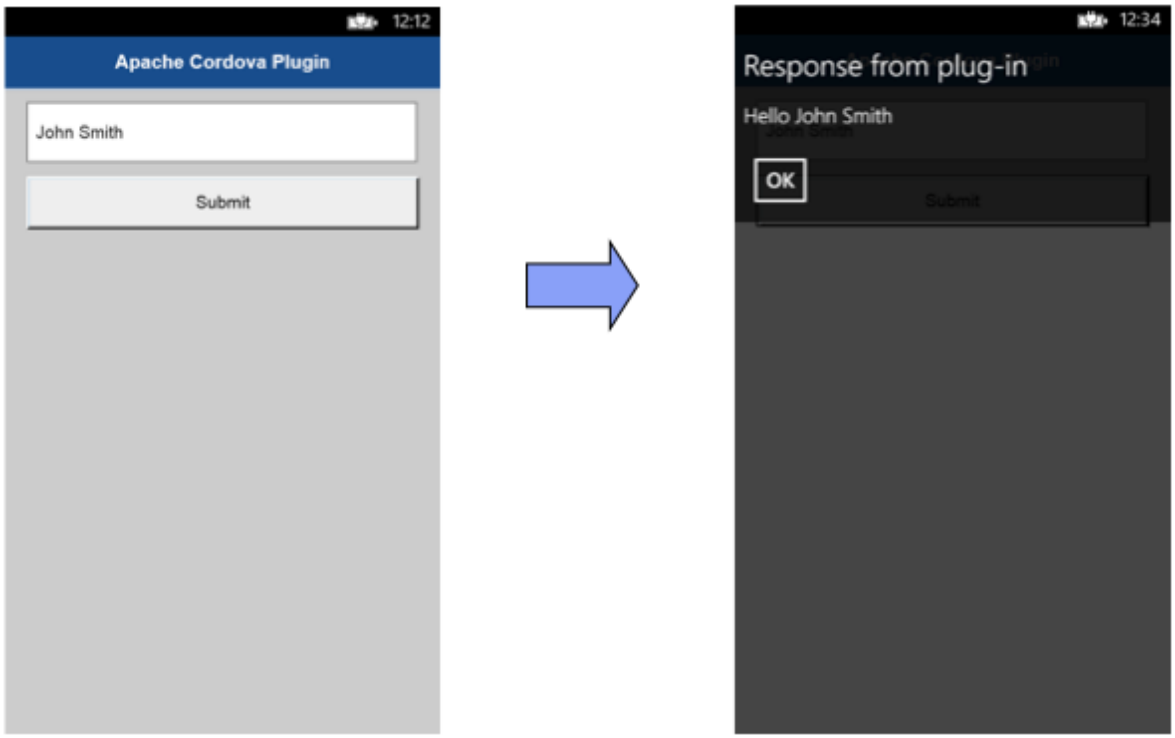
```
public void sayHello(string options)
        {
            string optVal = null;
            try {
                optVal = JsonHelper.Deserialize<string[]>(options)[0];
            }
            catch (Exception) {
                DispatchCommandResult(new PluginResult(PluginResult.Status.ERROR, "SayHelloPlugin
signaled an error"));
            }
```

- The DispatchCommandResult method returns the result to JavaScript, whether success or failure.

```
if (optVal == null)
            {
                DispatchCommandResult(new PluginResult(PluginResult.Status.ERROR, "Got null value
as input"));
            }
            else
            {
                DispatchCommandResult(new PluginResult(PluginResult.Status.OK, "Hello " + optVal));
            }
        }
    }
}
```

Sample application

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


12:12
Apache Cordova Plugin
John Smith
Submit
12:34
Response from plug-in
Hello John Smith
OK
Submit