

Windows Phone 8 - Implementing Cordova plug-ins

Overview

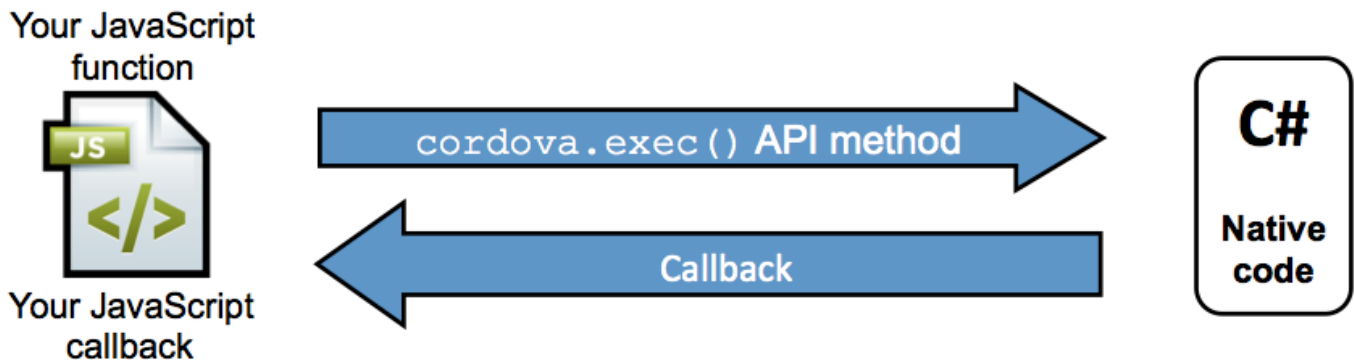
In some cases, developers of a MobileFirst application might have to use a specific third-party native library or a device function that is not yet available in Apache Cordova. With Apache Cordova, developers can create an Apache Cordova plug-in, which means that they create custom native code blocks, 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.

```

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

Implementing `cordova.exec()` in JavaScript

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

```
1  function sayHello() {  
2      var name = $("#NameInput").val();  
3      cordova.exe(sayHelloSuccess, sayHelloFailure, "SayHelloPlugin", "sayHello", [name]);  
4  }
```

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.

```
1  function sayHelloSuccess(data){  
2      WL.SimpleDialog.show(  
3          "Response from plug-in", data,  
4          [{text: "OK", handler: function() {WL.Logger.debug("Ok button pressed");}}]  
5      );  
6  }  
7  
8  function sayHelloFailure(data){  
9      WL.SimpleDialog.show(  
10         "Response from plug-in", data,  
11         [{text: "OK", handler: function() {WL.Logger.debug("Ok button pressed");}}]  
12     );  
13 }
```

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.

```
1  using System;  
2  using System.Collections.Generic;  
3  using System.Linq;  
4  using System.Text;  
5  using System.Threading.Tasks;  
6  
7  using WPCordovaClassLib.Cordova;  
8  using WPCordovaClassLib.Cordova.Commands;  
9  using WPCordovaClassLib.Cordova.JSON;  
10  
11  namespace Cordova.Extension.Commands  
12  {  
13      public class SayHelloPlugin : BaseCommand  
14      {
```

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.

```
1 public void sayHello(string options)
2     {
3         string optVal = null;
4         try {
5             optVal = JsonHelper.Deserialize<string[]>(options)[0];
6         }
7         catch (Exception) {
8             DispatchCommandResult(new PluginResult(PluginResult.Status.ERROR, "SayHelloPlugin
9         }
```

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

```
1 if (optVal == null)
2     {
3         DispatchCommandResult(new PluginResult(PluginResult.Status.ERROR, "Got null value
4     }
5     else
6     {
7         DispatchCommandResult(new PluginResult(PluginResult.Status.OK, "Hello " + optVal));
8     }
9 }
10 }
11 }
```

Sample application

Click to download

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

