

# Android - Implementing Apache Cordova plugin

## 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.

**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, you can place implementation code in the `wlCommonInit()` function in `common\js\main.js`.

Jump to:

- Creating a plug-in
- Declaring a plug-in
- Implementing `cordova.exec()` in JavaScript
- Implementing the Java code of a Cordova plug-in
- Sample application

## Creating a plug-in

1. Declare the plug-in in the `config.xml` file.
2. Use the `cordova.exec()` API in the JavaScript code.
3. Create the plug-in class that will run natively in Android. 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

You must declare the plug-in 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\res\xml` folder in the Android environment.

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

## Implementing `cordova.exec()` in JavaScript

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

```
1 function sayHello() {
2     var name = $("#NameInput").val();
3     cordova.exec(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 Java code of a Cordova plug-in

After you have declared the plug-in and the JavaScript implementation is ready, you can implement the Cordova plug-in.

1. Add a new Java class file.
2. Extend the `org.apache.cordova.CordovaPlugin` class and add the required import statements.

```
1 public class SayHelloPlugin extends CordovaPlugin {
```

3. Implement an `execute` method.

- The arguments contain information that is required by a plug-in, such as action, arguments array, and callback context.

```
1 public boolean execute(String action, JSONArray args, CallbackContext callbackContext)
2     throws JSONException {
```

4. If the supplied action is `sayHello`, retrieve the first argument from the `args` array, prepare a `responseText` string and, by using the `callbackContext` argument, call the success callback with this `responseText` string as the argument.

```

1  if (action.equals("sayHello")){
2      try {
3          String responseText = "Hello " + args.getString(0);
4          callbackContext.success(responseText);
5      } catch (JSONException e){
6          callbackContext.error("Failed to parse parameters");
7      }
8      return true;
9  }

```

5. Returning `false` means that the action that is supplied from JavaScript was not recognized.

```

1      return false;
2  }

```

## Sample application

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/ApacheCordovaPluginsProject.zip>)  
the Studio project.

