CREACIÓN DE PLUGINS CON APACHE CORDOVA

Germán Caballero Rodríguez

Creación de un plugin

- Bien lo que vamos a ver en esta parte del curso es cómo crear un plugin que podamos reutilizar en cualquier de nuestros desarrollo con PhoneGap/Cordova.
- Los plugins son la mejor forma de extender funcionalidad en PhoneGap permitiendo la ejecución de código nativo invocado desde Javascript.
- Nos vamos a centrar en cómo crear un plugin para dispositivos Android

Creación de un plugin

- Lo primero que vamos a hacer es crear nuestro plugin.
- Para ello vamos a crear una estructura de carpetas que almacenen todos los componentes necesarios para la implementación del plugin.
- Por tanto en cualquier parte que decidamos de nuestro sistema de ficheros vamos a crear la carpeta “**cordova-plugin-pruebaplugin**”.

Creación de un plugin

- La forma en la que PhoneGap se comunica con la parte nativa es a través de esta llamada javascript:

```
cordova.exec(successCallback,
failureCallback, class, method, [arguments]);
```

Creación de un plugin

- **Estos son los argumentos:**
- **successCallback**: será la función que se quiera ejecutar cuando el resultado de la invocación sea satisfactorio.
- **failureCallback**: será la función a ejecutar cuando el resultado de la invocación no sea satisfactorio.
- **class**: será el nombre de la clase de nuestro código nativo, sin tener en cuenta el nombre del paquete.
- **method**: será el nombre de la acción que se va a tener en cuenta en el método “execute” de la clase anterior que queremos invocar.
- **[arguments]:** será un array, generalmente en formato JSON, donde se le pasan todos los parámetros de entrada al método invocado.

Creación de un plugin

Por tanto lo primero que vamos a hacer es crear nuestro **fichero js** con la llamada al código nativo.

Para ello dentro de la carpeta "**cordova-plugin-pruebaplugin**" creamos otra carpeta llamada "www" y dentro creamos el fichero **"pruebaplugin.js"** con el siguiente contenido:

Creación de un plugin

Por tanto lo primero que vamos a hacer es crear nuestro **fichero js** con la llamada al código nativo.

Para ello dentro de la carpeta "**cordova-plugin-pruebaplugin**" creamos otra carpeta llamada "www" y dentro creamos el fichero **"pruebaplugin.js"** con el siguiente contenido:

Creación de un plugin

```
var pruebaplugin = {

    funcionTest: function(successCallback, errorCallback,
    cantidad){
      cordova.exec(successCallback,
          errorCallback,
          "PruebaPlugin",
          "accionTest",
          [{
            "cantidad":cantidad
          }]
      );

    }
}

module.exports = pruebaplugin;
```

Creación de un plugin

- En la llamada estamos indicando que vamos a tener una clase llamada “PruebaMail” con la acción “accionTest” que será el encargado de recoger los argumentos y realizar el envio del mensaje.
- Además fijaos en la forma de pasar los argumentos para que se recojan en formato JSON.
- Por tanto el siguiente paso lógico es crear la clase especificada.

Creación de un plugin

- Para ello dentro de la carpeta "**cordova-plugin-pruebaplugin**" vamos a crear otra llamada **"src"** y dentro de esta otra llamada **"android"** para distinguir las plataformas por si queremos extender la funcionalidad de este plugin a otras plataformas soportadas como IOS o Windows Phone.
- Dentro de la carpeta "android" vamos a crear el fichero **PruebaPlugin.java**.
- Con contenido de la siguientes diapositivas

Creación de un plugin

```
package com.pronoide.plugin.pruebaplugin;

import org.apache.cordova.CallbackContext;
import org.apache.cordova.CordovaPlugin;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class PruebaPlugin extends CordovaPlugin {
```

Creación de un plugin

```
public class PruebaPlugin extends CordovaPlugin {

    public static final String ACTION = "accionTest";

    public boolean execute(String action, JSONArray jsonArgs,
    CallbackContext callbackContext) throws JSONException {
        try {
            if (ACTION.equals(action)) {
                JSONObject args = jsonArgs.getJSONObject(0);
                String cantidad = args.getString("cantidad");
                callbackContext.success("Todo Ok desde el nativo
                Android. Recibido en 'cantidad': " + cantidad);
                return true;
            } else {
                callbackContext.error("La accion no existe, error desde
                Android nativo");
                return false;
            }
        } catch (Exception e) {
            callbackContext.error(e.getMessage());
            return false;
        }
    }
}
```

Creación de un plugin

- Como vemos no es más que una clase Java que extiende de la clase abstracta CordovaPlugin que hace que tenga que implementar el método “execute”.
- En este método distinguimos por el nombre de la acción que le pasamos, dependiendo del resultado de la acción devolvemos true o false y llamaremos al correspondiente callback.
- También tenemos que fijarnos en cómo recupera los argumentos de entrada en JSON a sus respectivas variables String.

Creación de un plugin

- Si quisieramos se podrían añadir nuevas clases con otras dependencias a librerías externas de java.
- En este punto ya tenemos todo el código de nuestro plugin, pero ahora viene la parte importante de cómo indicar al proyecto que vaya hacer uso de nuestro plugin la forma en la que tiene que incluir estos fuentes.

Creación de un plugin

- PhoneGap/Cordova lo consigue a través de la definición del fichero **plugin.xml**.
- Este fichero contiene la **información** de nuestro plugin como el nombre, las plataformas que soporta y sobre todo la información de distribución de los fuentes del plugin el proyecto que lo vaya a utilizar.
- En nuestro caso vamos a crear el fichero plugin.xml el directorio raíz del plugin con el siguiente contenido:

Creación de un plugin

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://www.phonegap.com/ns/plugins/1.0"
        id="com.pronoide.plugin.pruebaplugin"
        version="0.1.0">

    <name>PruebaPlugin</name>
    <description>PruebaPlugin Plugin</description>
    <license>MIT</license>
    <keywords>phonegap,mail</keywords>

    <js-module src="www/pruebaplugin.js" name="PruebaPlugin">
        <clobbers target="pruebaplugin" />
    </js-module>
    <!-- android -->
    <platform name="android">
        <config-file target="res/xml/config.xml" parent="/*">
            <feature name="PruebaPlugin">
                <param name="android-package" value=
                "com.pronoide.plugin.pruebaplugin.PruebaPlugin"/>
            </feature>
        </config-file>
        <source-file src="src/android/PruebaPlugin.java" target-dir=
        "src/com/pronoide/plugin/pruebaplugin" />
     </platform>
</plugin>
```

Creación de un plugin

- En caso de querer extender esta funcionalidad a otras plataformas tendríamos que hacer la definición de fuentes de forma similar.
- En este punto ya tenemos el código de nuestro plugin listo para ser probado en cualquier proyecto Cordova/PhoneGap.
- Para hacer uso de nuestro plugin vamos a crear un nuevo proyecto Cordova/PhoneGap.

cordova create TestPlugin com.pronoide.TestPlugin “Probador del Plugin”

Creación de un plugin

- Esto creará la estructura por defecto de un proyecto Cordova.
- Entrando en la carpeta “TestPlugin” podemos añadir nuestro plugin ejecutando:

cordova plugin add ..\cordova-plugin-pruebaplugin o la ruta absoluta del plugin

- c**ordova platform add android**

Creación de un plugin

- Ahora para probar el plugin vamos primero a editar el fichero “www/index.html” para añadir un div que permita mostrar el estado de la invocación del método. Quedando el código de esta forma:

Creación de un plugin

```
        <meta name="viewport" content="user-scalable=no,
        initial-scale=1, maximum-scale=1, minimum-scale=1,
        width=device-width">
        <link rel="stylesheet" type="text/css" href="css/ind
        <title>Test Plugin</title>
    </head>
    <body>
        <div class="app">
            <h1>Test Plugin</h1>
            <div id="deviceready" class="blink">
                <p class="event listening">Conectando</p>
                <p class="event received">Dispositivo OK</p>
                <div id="estado">ESTADO</div>
            </div>
        </div>
        <script type="text/javascript" src="cordova.js"></sc
        <script type="text/javascript" src="js/index.js"></s
    </body>
</html>
```

Creación de un plugin

- Ahora añadimos el código en el fichero **“www/index.js”** que hace la invocación al método teniendo en cuenta que le tendréis que pasar una cadena de texto.
- El código de este fichero quedaría de la siguiente forma:

Creación de un plugin

```
    // The scope of 'this' is the event. In order to call the
    'receivedEvent'
    // function, we must explicitly call 'app.receivedEvent(...);'
    onDeviceReady: function() {
        app.receivedEvent('deviceready');
        pruebaplugin.funcionTest (app.testSuccess, app.testError, "987");
    },
    testSuccess : function(mensajeDevuelto) {
        var estado = document.getElementById('estado');
        estado.innerHTML = '<p>Mensaje enviado</p><p>' + mensajeDevuelto
        + '</p>';
    },
    testError : function(error) {
        var estado = document.getElementById('estado');
        estado.innerHTML = '<p>Mensaje NO enviado:' + error + '</p>';
    },
    // Update DOM on a Received Event
    receivedEvent: function(id) {
        var parentElement = document.getElementById(id);
        var listeningElement = parentElement.querySelector('.listening');
```

Creación de un plugin

- Dentro de la función “onDeviceReady” realizamos la llamada a nuestra función. En caso de que la invocación sea errónea se ejecutará el método “**testError**” mostrando en pantalla el error que se ha producido.
- Si la invocación se hace correctamente en la pantalla se mostrará un mensaje enviado desde código nativo.
- Para distribuir nuestro plugin simplemente subimos los fuentes a una cuenta de GitHub.
- Si queréis probarlo en vuestro proyecto Cordova/PhoneGap solo tendréis que ejecutar:

cordova plugin add https://github.com/cuentagithub/cordova-plugin-testplugin.git