

# ¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.  
 Ese apoyo que siempre quiso tener...

## 1. Desarrollo de componentes y proyectos a medida



## 2. Auditoría de código y recomendaciones de mejora

## 3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



## 4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,  
 HTML5, CSS3, JavaScript-jQuery

Control de autenticación y  
 acceso (Spring Security)  
 UDDI

JPA-Hibernate, MyBatis  
 Motor de búsqueda empresarial (Solr)  
 ETL (Talend)

Gestor portales (Liferay)  
 Gestor de contenidos (Alfresco)  
 Aplicaciones híbridas

Web Services  
 Rest Services  
 Social SSO  
 SSO (Cas)

Dirección de Proyectos Informáticos.  
 Metodologías ágiles  
 Patrones de diseño  
 TDD

Tareas programadas (Quartz)  
 Gestor documental (Alfresco)  
 Inversión de control (Spring)

BPM (jBPM o Bonita)  
 Generación de informes (JasperReport)  
 ESB (Open ESB)


[Registrarme](#)  
[Olvidé mi contraseña](#)
» Estás en: [Inicio](#) » [Tutoriales](#) » [Hooks en Cordova: Cargar todos los plugins de forma automática](#)**Rubén Aguilera Díaz-Heredero**

Consultor tecnológico de desarrollo de proyectos informáticos.

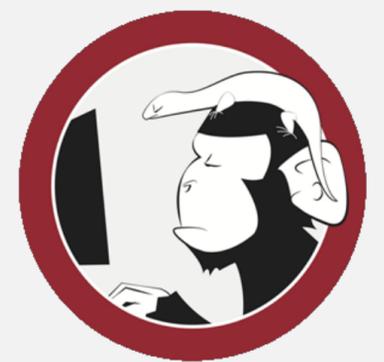
Ingeniero en Informática, especialidad en Ingeniería del Software

Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

[Ver todos los tutoriales del autor](#)

## Catálogo de servicios Autentia



## Síguenos a través de:



## Últimas Noticias

» [Curso JBoss de Red Hat](#)» [Si eres el responsable o líder técnico, considérate desafortunado. No puedes culpar a nadie por ser gris](#)» [Portales, gestores de contenidos documentales y desarrollos a medida](#)» [Comentando el libro Start-up Nation, La historia del milagro económico de Israel, de Dan Senor & Salu Singer](#)» [Screencasts de programación narrados en Español](#)[Histórico de noticias](#)

## Últimos Tutoriales

» [Generación de vistas HTML5 con el soporte de JSF2: pass through](#)» [Monta fácilmente tu proyecto con Spring Boot Starter POMs](#)Fecha de publicación del tutorial: **2014-10-31**Tutorial visitado 1 veces [Descargar en PDF](#)

# Hooks en Cordova: Cargar todos los plugins de forma automática

## 0. Índice de contenidos.

- [1. Entorno](#)
- [2. Introducción](#)
- [3. Vamos al lío](#)
- [4. Conclusiones](#)

## 1. Entorno

Este tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Mac Book Pro 15" (2,3 Ghz Intel Core i7, 16 GB DDR3)
- Sistema Operativo: Mac OS X Mavericks
- node v0.10.32
- cordova 3.4.1-0.1.0

## 2. Introducción

Los hooks son los elementos que Cordova proporciona para extender su funcionalidad en función de las necesidades que tengamos. Son pequeños scripts que podemos encontrar dentro del directorio PROJECT\_CORDOVA\_HOME/hooks, separados en directorios nombrados como la fase en la que tienen que ser ejecutados. Las distintas fases de las que podemos hacer uso son: after\_build, after\_compile, after\_docs, before\_build, ... Tenéis un listado con todas ellas en este [link](#).

Estos scripts pueden estar escritos en cualquier lenguaje, pero el más idóneo es NodeJS dado que tiene que estar presente en la máquina para poder ejecutar Cordova y es completamente multiplataforma.

En este tutorial vamos a ver cómo crear un hook para poder cargar siempre que se necesiten los plugins de un proyecto sin tener que hacerlo uno por uno de forma manual; algo muy tedioso que hace que más de un desarrollador opte por subir la carpeta "plugins" y "platforms" a su repositorio de código con tal de no estar haciéndolo cada vez de forma manual.

## 3. Vamos al lío

Para poner un ejemplo más o menos real vamos a hacer un clone del repositorio de código que utilicé para la charla sobre Cordova en el Google Developer Group de Madrid, que podéis ver en este [enlace](#).

Entonces para hacer el clone ejecutamos desde el terminal:

[view plain](#) [print](#) ?01. 

```
git clone https://github.com/raguilera82/GDG-demo.git
```

Si le echáis un vistazo al código veréis que a priori no es un proyecto Cordova, dado que es buena práctica subir al repositorio de código solo los fuentes que no puedan volver a ser generados en el cliente. Esto cobra especial relevancia

cuando nuestro desarrollo es multiplataforma, ya que, ¿para qué quiere el desarrollador de la aplicación de Windows Phone en su Windows 8 el código de la aplicación de iOS?

Entonces lo primero que vamos a hacer es construir el proyecto de Cordova que tenga en cuenta el código que hay dentro de la carpeta "app". Esto lo conseguimos ejecutando el comando de create con la variante --link-to, que hace un enlace simbólico entre la carpeta "www" del proyecto de Cordova y la carpeta "app" de nuestro proyecto web.

```
view plain print ?
01. cordova create cordova-apps com.autentia.gdgdemo GDGdemo --link-to=app
```

**Nota:** Esto es solo si estamos en fase de desarrollo, para producción haríamos el link con la carpeta "dist" que contiene el código de producción de la aplicación web.

Ahora si entramos en la carpeta "cordova-apps" veremos que están los directorios típicos de una aplicación Cordova y concretamente el directorio "hooks" del que estamos hablando.

Si véis el README de mi proyecto, veréis que el siguiente paso es instalar uno por uno todos los plugins necesarios; que para el caso de esta demo son unos pocos. Este proceso lo podemos simplificar creando un hook, que se ejecute siempre que se añada una nueva plataforma al proyecto de Cordova.

Para hacer esto solo tenemos que crear un directorio dentro del directorio "hooks" con el nombre de la fase en la que se quiera ejecutar, en nuestro caso va a ser "after\_platform\_add" y dentro crearemos el fichero install\_all\_plugins.js con el siguiente contenido:

```
view plain print ?
01.  #!/usr/bin/env node
02.
03.  var pluginlist = [
04.    "org.apache.cordova.geolocation",
05.    "org.apache.cordova.camera",
06.    "org.apache.cordova.vibration",
07.    "org.apache.cordova.device",
08.    "org.apache.cordova.network-information",
09.    "org.apache.cordova.dialogs",
10.    "org.apache.cordova.device-motion"
11.  ];
12.
13.  var fs = require('fs');
14.  var path = require('path');
15.  var sys = require('sys')
16.  var exec = require('child_process').exec;
17.
18.  function puts(error, stdout, stderr) {
19.    sys.puts(stdout)
20.  }
21.
22.  pluginlist.forEach(function(plug) {
23.    exec("cordova plugin add " + plug, puts);
24.  });
```

Como véis el hook es muy sencillo, nosotros solo tenemos que rellenar una vez la lista de plugins que se tienen que cargar cada vez que una plataforma se añade al proyecto de Cordova.

Por último guardamos el fichero que es de vital importancia que tenga permisos de ejecución.

```
view plain print ?
01. chmod +x install_all_plugins.js
```

Y ahora cuando añadamos cualquier plataforma, el código del plugin se ejecutará y automáticamente nos cargará los plugins que hayamos especificado en la lista.

```
view plain print ?
01. cordova platform add android ios
```

Por último decir que el directorio "hooks" sí que se tiene que subir al repositorio de código. Si yo lo subiera a este repositorio no tendría que indicar como instalar los plugins necesarios, ya que todo se haría de forma automática. No lo voy a hacer para que podáis practicar con este tutorial ;-)

## 4. Conclusiones

Este es solo un ejemplo de la potencia que tienen los hooks en Cordova, la lista de fases de ejecución es muy amplia y el límite lo pone la imaginación.

Cualquier duda o sugerencia en la zona de comentarios.

Saludos.

## A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

★★★★★

Por favor, vota +1 o compártelo si te pareció interesante

[+](#) Share | [f](#) [t](#) [in](#) [e](#) [★](#) [g+](#) [0](#) [g+](#) [0](#)

» [\[S.O.L.I.D.\] Dependency inversion principle / Principio de inversión de dependencias](#)

» [\[S.O.L.I.D.\] Interface Segregation Principle / Principio de segregación de interfaz](#)

» [\[S.O.L.I.D.\] Liskov substitution](#)

## Últimos Tutoriales del Autor

» [Mockear la capa back con Dyson](#)

» [Automatizando los Smoke Test con TestLink y Jenkins](#)

» [Smoke Test implementados con TestNG y Selenium](#)

» [GitLab: Crear y gestionar nuestro servidor propio de Git](#)

» [Crear servidor propio de Git en CentOS 6.5](#)

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

» **Regístrate** y accede a esta y otras ventajas «



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

**PUSH THIS** | [Page Pushers](#) | [Community](#) | [Help?](#)

0 people brought clicks to this page

no clicks

+ + + + + + + +

powered by [karmacrazy](#)

