

Tutorial de Introducción a Grails

Adrián Cepillo Macías

12 de marzo de 2012



Índice

Índice	1
1. Instalación de Grails	2
2. Creando nuestra aplicación	2
3. Importando datos de prueba	4
4. Configuración del Data Source	4
5. Ampliando nuestra aplicación	6
5.1. Creando la clase Author	6
5.2. Relacionando con GORM	7
5.3. Cambiemos de estilo	7

1. Instalación de Grails

Siguiendo con el tutorial anterior ahora vamos a instalar Grails de nuevo manualmente. Primero descargar la aplicación desde [aquí](#). La extraemos en el directorio que prefiramos.

Luego establecer una variable de entorno:

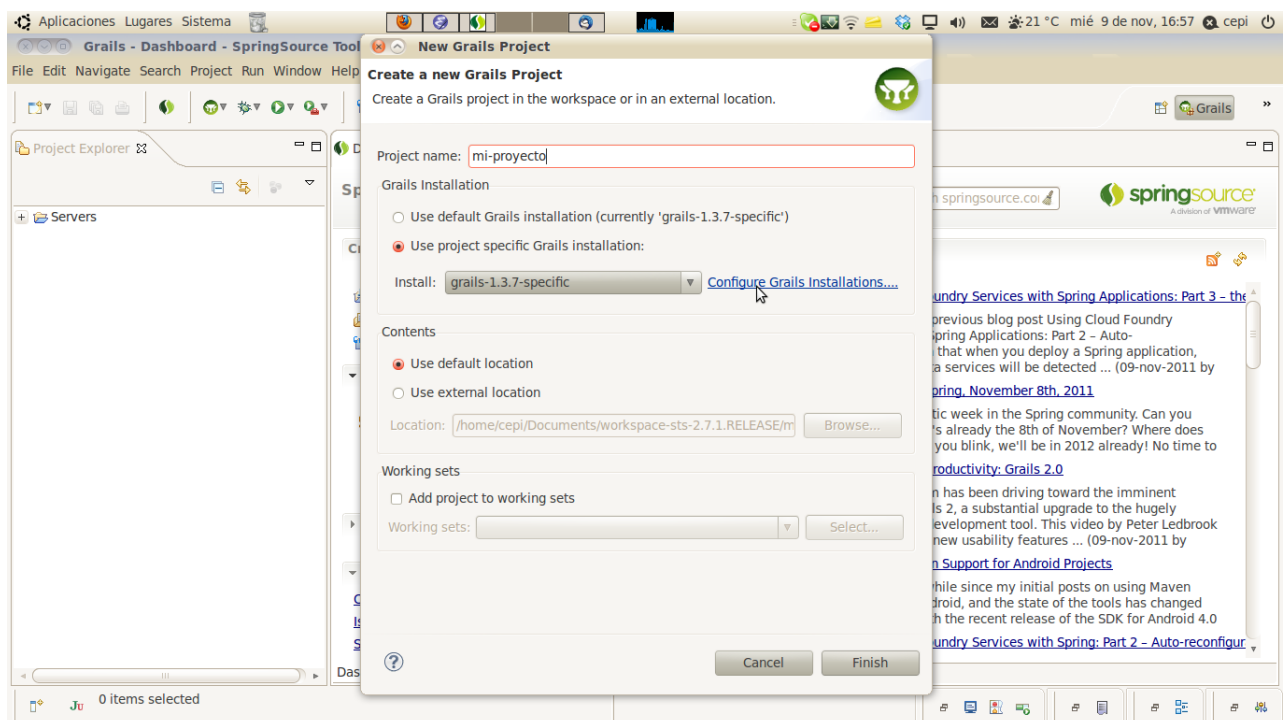
- `$ export GRAILS_HOME= /RUTA.EXTRACCION/bin`

Podemos hacer esto permanente introduciendo este comando al final del fichero `/.bashrc`.

Aún así nosotros no usaremos grails por consola, se muestra esta instalación por si se quiere usar grails independientemente de eclipse. Nosotros veremos en este tutorial algunos de los comandos que grails contiene para crear una aplicación, testarla, ... pero todo desde el prompt que nos ofrece Eclipse.

2. Creando nuestra aplicación

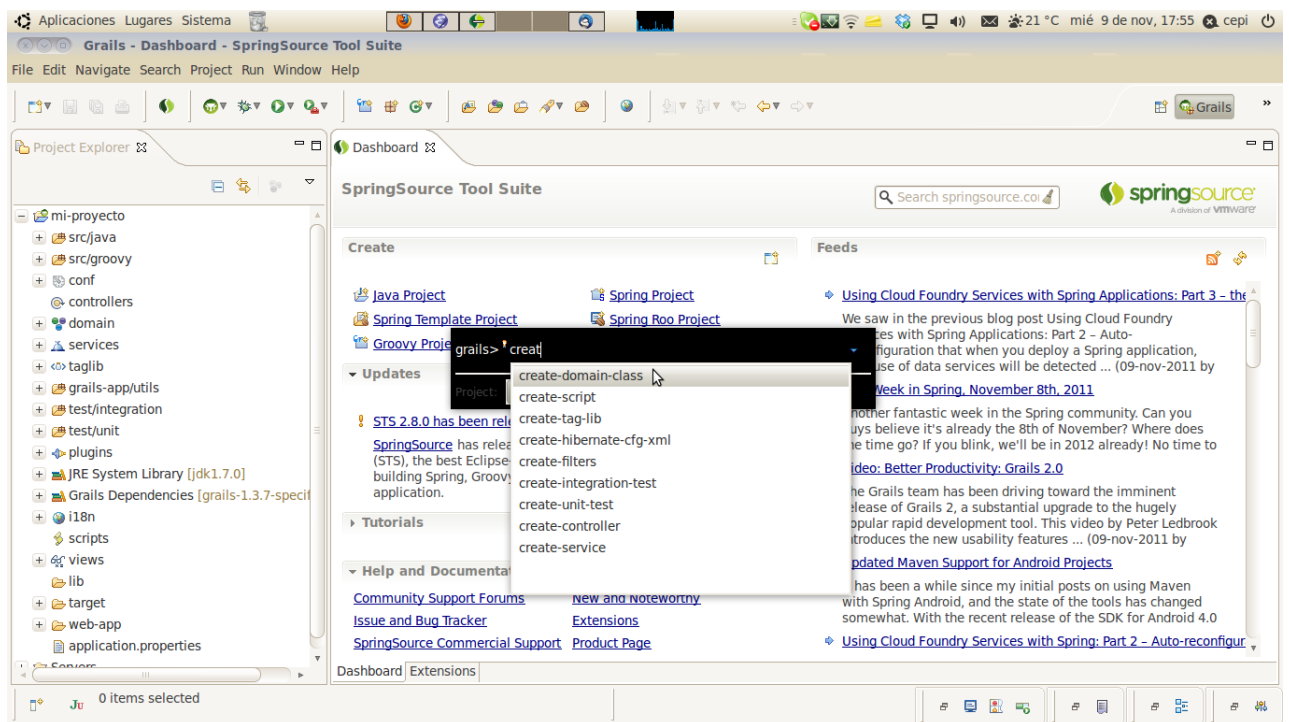
Vamos a crear de nuevo la aplicación *Book*. Creamos un nuevo proyecto, esta vez elegiremos una instalación específica. Para ello debemos clicar en *Configure Grails Installations* y añadiremos la versión que acabamos de descargarnos, indicando el directorio en el que se encuentra.



Nota: El comando de *Grails* para crear un proyecto es, `$ grails create-app mi-proyecto`. *Eclipse* no permite crear un proyecto, introduciendo este comando en el prompt.

Para conocer algunos de los comandos de *Grails* vamos a usar el prompt que nos presenta *Eclipse STS*. Podemos encontrarlo en `Navigate->Open Grails Commands Prompt`. O clicar en el simbolo de *Grails* en la barra de herramientas o pulsar `Ctrl+Alt+Shift+G` (`Cmd+Alt+Shift+G` en Mac).

Con Ctrl+Space (Cmd+Space) nos mostrará el asistente.



Ahora crearemos la clase de dominio introduciendo en el prompt:

- `grails> create-domain-class org.example.Book`

Modificamos el código:

```
1 package org.example
2
3 class Book {
4
5     String title
6 }
```

Y lo mismo para el controlador:

- `grails> create-controller org.example.Book`

```
1 package org.example
2
3 class BookController {
4
5     def scaffold = Book
6 }
```

Corremos la aplicación ejecutando:

- `grails> run-app`

3. Importando datos de prueba

Es probable que queramos probar la aplicación con ciertos datos y sería muy pesado tener que introducirlos a mano una y otra vez. Para ello podemos modificar el fichero de configuración *BootStrap.groovy*, dentro de *conf* en la jerarquía del proyecto.

```
1 import org.example.Book
2
3 class BootStrap {
4
5     def init = { servletContext ->
6         // Check whether the test data already exists.
7         if (!Book.count()) {
8             new Book(author: "John R. R. Tolkien", title: "The Simarillion").
9                 save(failOnError: true)
10            new Book(author: "John R. R. Tolkien", title: "The Hobbit").save(
11                failOnError: true)
12            new Book(author: "John R. R. Tolkien", title: "The Lord of the
13                Rings").save(failOnError: true)
14            new Book(author: "George R. R. Martin", title: "A Song of Ice and
15                Fire: 1 A Game of Thrones").save(failOnError: true)
16            new Book(author: "George R. R. Martin", title: "A Song of Ice and
17                Fire: 2 A Clash of Kings").save(failOnError: true)
18            new Book(author: "George R. R. Martin", title: "A Song of Ice and
19                Fire: 3 A Storm of Swords").save(failOnError: true)
20            new Book(author: "George R. R. Martin", title: "A Song of Ice and
                Fire: 4 A Feast for Crows").save(failOnError: true)
        }
    }
21
22     def destroy = {
23     }
24 }
```

Como podemos comprobar hemos redefinido *init* de esta forma cada vez que se inicie la aplicación se importaran estos datos a el modelo señalado.

Debemos notar la llamada a *Book.count()*. Este método devuelve el número de libros que existen ya en la base de datos. Con el *if(){...}* decimos que en caso que no existan otros datos de testeo, se introduzcan estos datos.

Por otra parte, llamamos al método *Book.save()*, el cuál guarda el objeto en la base de datos. Y establecemos la opción *failOnError: true* indicando que se lance una excepción en caso que el método falle.

4. Configuración del Data Source

Hasta ahora hemos estado trabajando con la base de datos HSQLDB en memoria, para desarrollo y testeo. Vamos a configurar una base de datos alternativa. Para ello modificamos el fichero

DataSource.groovy.

```
1  dataSource {
2      pooled = true
3      driverClassName = "org.hsqldb.jdbcDriver"
4      username = "sa"
5      password = ""
6  }
7  hibernate {
8      cache.use_second_level_cache = true
9      cache.use_query_cache = true
10     cache.provider_class = 'net.sf.ehcache.hibernate.EhCacheProvider'
11 }
12 // environment specific settings
13 environments {
14     development {
15         dataSource {
16             dbCreate = "create-drop" // one of 'create', 'create-drop
17                                     // , 'update'
18             url = "jdbc:hsqldb:mem:devDB"
19         }
20     }
21     test {
22         dataSource {
23             dbCreate = "update"
24             url = "jdbc:hsqldb:mem:testDb"
25         }
26     }
27     production {
28         dataSource {
29             dbCreate = "update"
30             driverClassName = "com.mysql.jdbc.Driver"
31             url = "jdbc:mysql://localhost/mi-proyecto"
32             username="root"
33             password="root"
34         }
35     }
36 }
```

En este caso estamos usando Mysql. Grails ya tiene el driver para JDBC pero es necesario descomentar varias líneas del fichero *BuildConfig.groovy*.

```
1  grails.project.class.dir = "target/classes"
2  grails.project.test.class.dir = "target/test-classes"
3  grails.project.test.reports.dir = "target/test-reports"
4  //grails.project.war.file = "target/${appName}-${appVersion}.war"
5  grails.project.dependency.resolution = {
6      // inherit Grails' default dependencies
7      inherits("global") {
8          // uncomment to disable ehcache
```

```

9      // excludes 'ehcache'
10    }
11    log "warn" // log level of Ivy resolver, either 'error', 'warn', '
      info', 'debug' or 'verbose'
12    repositories {
13      grailsPlugins()
14      grailsHome()
15      grailsCentral()
16
17      // uncomment the below to enable remote dependency resolution
18      // from public Maven repositories
19      mavenLocal()
20      mavenCentral()
21      mavenRepo "http://snapshots.repository.codehaus.org"
22      mavenRepo "http://repository.codehaus.org"
23      mavenRepo "http://download.java.net/maven/2/"
24      mavenRepo "http://repository.jboss.com/maven2/"
25    }
26    dependencies {
27      // specify dependencies here under either 'build', 'compile', '
      runtime', 'test' or 'provided' scopes eg.
28
29      runtime 'mysql:mysql-connector-java:5.1.13'
30    }
31  }

```

5. Ampliando nuestra aplicación

5.1. Creando la clase Author

Vamos a añadir algunos elementos nuevos a la aplicación empezando por crear otra clase llamada *Author* que relacionar con *Book*. Se muestra primero el código de la *domain class*:

```

1  package org.example
2
3  class Author {
4      static hasMany = [books:Book]
5
6      String name
7
8      static constraints = {
9          name(blank:false)
10     }
11
12 }

```

Y por otra parte el controlador:

```

1 package org.example
2
3 class AuthorController {
4
5     def scaffold = Author
6 }

```

5.2. Relacionando con GORM

Ahora vamos a crear una relación de uno a muchos unidireccional para ello modificamos la clase *Author*, definiendo que ha escrito muchos libros:

```

1 package org.example
2
3 class Author {
4     static hasMany = [book:Book]
5
6     String name
7     String surname
8
9     static constraints = {
10         name(blank:false)
11         surname(blank:false)
12     }
13
14 }

```

En el caso de la clase *Book* no es necesario modificarla pero si se deseara crear una relación bidireccional habría que añadir la siguiente línea de código:

- static belongsTo = [author:Author]

5.3. Cambiemos de estilo

Grails tiene por defecto una hoja de estilo que se encuentra en el directorio *mi-proyecto/web-app/css/main.css* en el siguiente documento hemos modificado el color de los enlaces y los encabezados `<h1>`. Posteriormente se mostrará como crear nuevas vistas con GSP y se añadirán nuevas hojas de estilos.

```

1 html * {
2     margin: 0;
3     /*padding: 0; SELECT NOT DISPLAYED CORRECTLY IN FIREFOX */
4 }
5
6 /* GENERAL */
7
8 .spinner {
9     padding: 5px;
10    position: absolute;

```



```

11     right: 0;
12 }
13
14 body {
15     background: #fff;
16     color: #333;
17     font: 11px verdana, arial, helvetica, sans-serif;
18 }
19 #grailsLogo {
20     padding: 20px;
21 }
22
23 a:link, a:visited, a:hover {
24     color: #4169E1;
25     font-weight: bold;
26     text-decoration: none;
27 }
28
29 h1 {
30     color: #4169E1;
31     font-weight: normal;
32     font-size: 16px;
33     margin: .8em 0 .3em 0;
34 }
35
36 ul {
37     padding-left: 15px;
38 }
39
40 input, select, textarea {
41     background-color: #fcfcfc;
42     border: 1px solid #ccc;
43     font: 11px verdana, arial, helvetica, sans-serif;
44     margin: 2px 0;
45     padding: 2px 4px;
46 }
47 select {
48     padding: 2px 2px 2px 0;
49 }
50 textarea {
51     width: 250px;
52     height: 150px;
53     vertical-align: top;
54 }
55
56 input:focus, select:focus, textarea:focus {
57     border: 1px solid #b2d1ff;
58 }
59
60 .body {

```

```

61     float: left;
62     margin: 0 15px 10px 15px;
63 }
64
65 /* NAVIGATION MENU */
66
67 .nav {
68     background: #fff url(../images/skin/shadow.jpg) bottom repeat-x;
69     border: 1px solid #ccc;
70     border-style: solid none solid none;
71     margin-top: 5px;
72     padding: 7px 12px;
73 }
74
75 .menuButton {
76     font-size: 10px;
77     padding: 0 5px;
78 }
79 .menuButton a {
80     color: #333;
81     padding: 4px 6px;
82 }
83 .menuButton a.home {
84     background: url(../images/skin/house.png) center left no-repeat;
85     color: #333;
86     padding-left: 25px;
87 }
88 .menuButton a.list {
89     background: url(../images/skin/database_table.png) center left no-
90         repeat;
91     color: #333;
92     padding-left: 25px;
93 }
94 .menuButton a.create {
95     background: url(../images/skin/database_add.png) center left no-
96         repeat;
97     color: #333;
98     padding-left: 25px;
99 }
100
101 /* MESSAGES AND ERRORS */
102
103 .message {
104     background: #f3f8fc url(../images/skin/information.png) 8px 50% no-
105         repeat;
106     border: 1px solid #b2d1ff;
107     color: #006dba;
108     margin: 10px 0 5px 0;
109     padding: 5px 5px 5px 30px
110 }

```

```

108
109 div.errors {
110     background: #fff3f3;
111     border: 1px solid red;
112     color: #cc0000;
113     margin: 10px 0 5px 0;
114     padding: 5px 0 5px 0;
115 }
116 div.errors ul {
117     list-style: none;
118     padding: 0;
119 }
120 div.errors li {
121     background: url(../images/skin/exclamation.png) 8px 0% no-
122         repeat;
123     line-height: 16px;
124     padding-left: 30px;
125 }
126 td.errors select {
127     border: 1px solid red;
128 }
129 td.errors input {
130     border: 1px solid red;
131 }
132 td.errors textarea {
133     border: 1px solid red;
134 }
135
136 /* TABLES */
137
138 table {
139     border: 1px solid #ccc;
140     width: 100%
141 }
142 tr {
143     border: 0;
144 }
145 td, th {
146     font: 11px verdana, arial, helvetica, sans-serif;
147     line-height: 12px;
148     padding: 5px 6px;
149     text-align: left;
150     vertical-align: top;
151 }
152 th {
153     background: #fff url(../images/skin/shadow.jpg);
154     color: #666;
155     font-size: 11px;
156     font-weight: bold;

```

```

157     line-height: 17px;
158     padding: 2px 6px;
159 }
160 th a:link, th a:visited, th a:hover {
161     color: #333;
162     display: block;
163     font-size: 10px;
164     text-decoration: none;
165     width: 100%;
166 }
167 th.asc a, th.desc a {
168     background-position: right;
169     background-repeat: no-repeat;
170 }
171 th.asc a {
172     background-image: url(../images/skin/sorted_asc.gif);
173 }
174 th.desc a {
175     background-image: url(../images/skin/sorted_desc.gif);
176 }
177
178 .odd {
179     background: #f7f7f7;
180 }
181 .even {
182     background: #fff;
183 }
184
185 /* LIST */
186
187 .list table {
188     border-collapse: collapse;
189 }
190 .list th, .list td {
191     border-left: 1px solid #ddd;
192 }
193 .list th:hover, .list tr:hover {
194     background: #b2d1ff;
195 }
196
197 /* PAGINATION */
198
199 .paginateButtons {
200     background: #fff url(../images/skin/shadow.jpg) bottom repeat-x;
201     border: 1px solid #ccc;
202     border-top: 0;
203     color: #666;
204     font-size: 10px;
205     overflow: hidden;
206     padding: 10px 3px;

```

```

207 }
208 .paginateButtons a {
209     background: #fff;
210     border: 1px solid #ccc;
211     border-color: #ccc #aaa #aaa #ccc;
212     color: #666;
213     margin: 0 3px;
214     padding: 2px 6px;
215 }
216 .paginateButtons span {
217     padding: 2px 3px;
218 }
219
220 /* DIALOG */
221
222 .dialog table {
223     padding: 5px 0;
224 }
225
226 .prop {
227     padding: 5px;
228 }
229 .prop .name {
230     text-align: left;
231     width: 15%;
232     white-space: nowrap;
233 }
234 .prop .value {
235     text-align: left;
236     width: 85%;
237 }
238
239 /* ACTION BUTTONS */
240
241 .buttons {
242     background: #fff url(../images/skin/shadow.jpg) bottom repeat-x;
243     border: 1px solid #ccc;
244     color: #666;
245     font-size: 10px;
246     margin-top: 5px;
247     overflow: hidden;
248     padding: 0;
249 }
250
251 .buttons input {
252     background: #fff;
253     border: 0;
254     color: #333;
255     cursor: pointer;
256     font-size: 10px;

```

```
257     font-weight: bold;
258     margin-left: 3px;
259     overflow: visible;
260     padding: 2px 6px;
261 }
262 .buttons input.delete {
263     background: transparent url(../images/skin/database_delete.png) 5px
264         50% no-repeat;
265     padding-left: 28px;
266 }
267 .buttons input.edit {
268     background: transparent url(../images/skin/database_edit.png) 5px
269         50% no-repeat;
270     padding-left: 28px;
271 }
272 .buttons input.save {
273     background: transparent url(../images/skin/database_save.png) 5px
274         50% no-repeat;
275     padding-left: 28px;
276 }
```