Tutorial de Introducción a Grails

Ingeniería Web – Universidad de Cádiz

19 de marzo de 2012

Índice

1. Instalación de Grails

En este tutorial vamos a instalar Grails manualmente. Primero descargar la aplicacion desde grails.org. La extraemos en el directorio que deseemos, por ejemplo en el directorio ~/grails.

Luego hay que definir una variable de entorno:

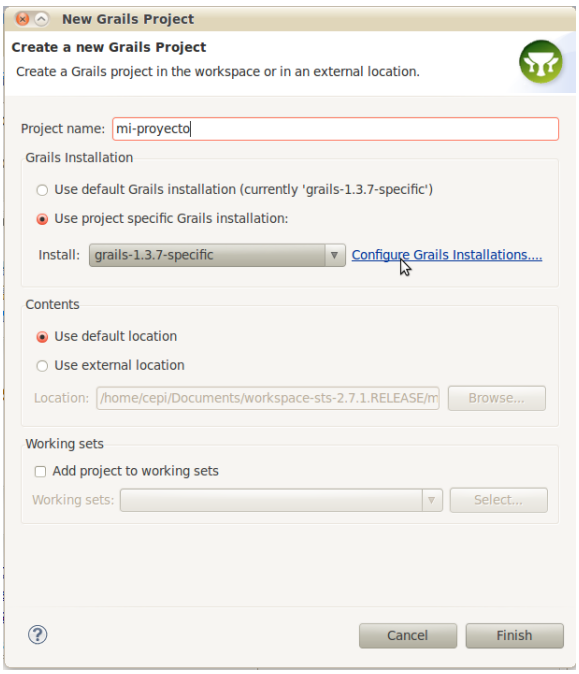
```
1 $ export GRAILS_HOME=~/grails/bin
```

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

Aunque en este tutorial no usaremos grails por consola, se muestra esta instalación por si se necesita usar grails independientemente de Eclipse STS. Veremos en este tutorial algunos de los comandos que grails contiene para generar el esqueleto de una aplicacion web, probarla, etc. Todo ello lo haremos desde el *prompt* que nos ofrece Eclipse STS.

2. Creando una aplicación

Vamos a crear de nuevo la aplicación que maneja libros. Creamos un nuevo proyecto y esta vez elegiremos una instalación específica de Grails, en lugar de la que viene incluida en Eclipse STS. Para ello debemos pulsar en *Configure Grails Installations* y añadir la versión de Grails que acabamos de descargarnos, indicando el directorio en el que se encuentra.

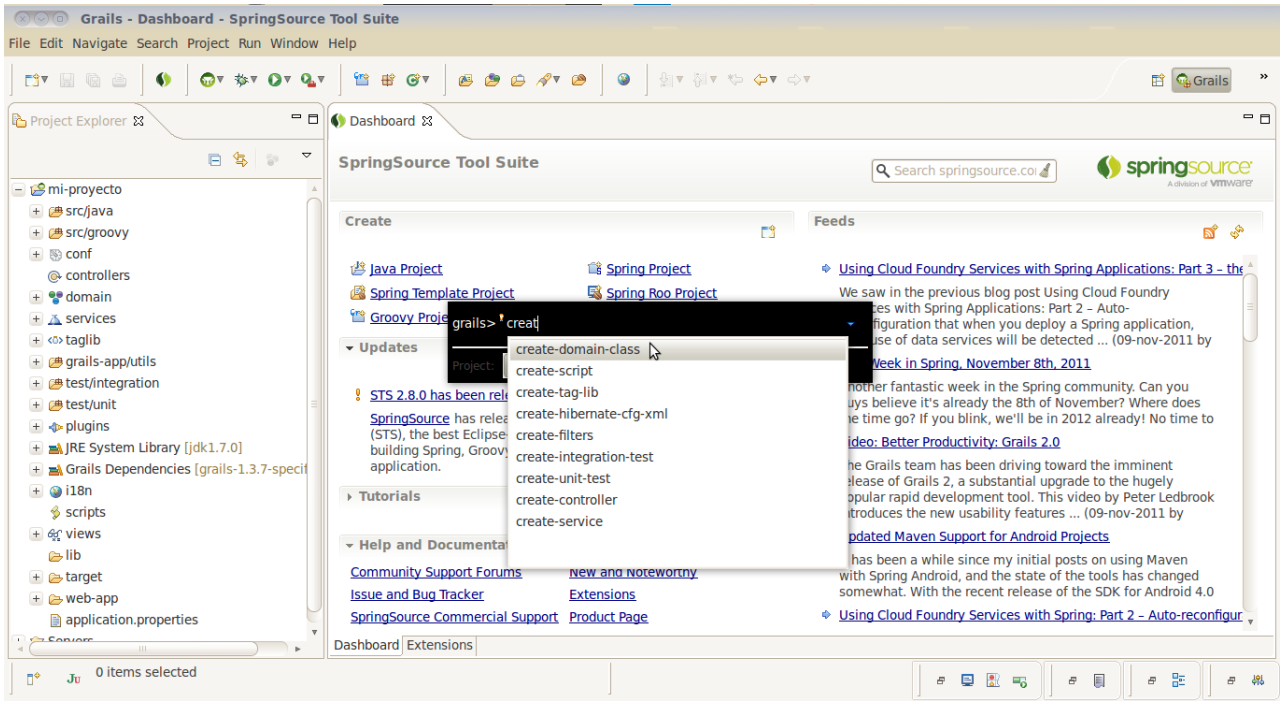


Nota: El comando de *Grails* para crear un proyecto es:

```
$ grails create-app mi-proyecto
```

Eclipse STS no permite crear un nuevo 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, pinchar 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 de comandos posibles a ejecutar.



Ahora crearemos la clase de dominio introduciendo en el prompt:

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

Modificamos el código:

```
package org.example

class Book {
  String title
}
```

Y lo mismo para el controlador:

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

```
package org.example

class BookController {
  def scaffold = Book
}
```

Ejecutamos la aplicación mediante:

```
grails> run-app
```

Hasta ahora hemos hecho lo mismo que el tutorial anterior de Eclipse STS, pero pidiéndole al scaffolding de Grails que nos genere distintos componentes de la aplicación web por nosotros.

3. Creando datos *mock* de prueba

Es probable que queramos probar la aplicación con ciertos datos *mock* (simulados) y sería muy pesado tener que introducirlos a mano cada vez que se ejecutara la aplicación. Para ello modificaremos el fichero de configuración *BootStrap.groovy*, dentro de *conf* en la jerarquía del proyecto.

```
import org.example.Book

class BootStrap {
 def init = { servletContext ->
  // Check whether the test data already exists.
   if (!Book.count()) {
     new Book(author: "John R. R. Tolkien",title: "The Simarillion").save(
         failOnError: true)
     new Book(author: "John R. R. Tolkien", title: "The Hobbit").save(
         failOnError: true)
     new Book(author: "John R. R. Tolkien", title: "The Lord of the Rings").
         save(failOnError: true)
     new Book(author: "George R. R. Martin", title: "A Song of Ice and Fire:
          1 A Game of Thrones").save(failOnError: true)
     new Book(author: "George R. R. Martin", title: "A Song of Ice and Fire:
          2 A Clash of Kings").save(failOnError: true)
     new Book(author: "George R. R. Martin", title: "A Song of Ice and Fire:
          3 A Storm of Swords").save(failOnError: true)
     new Book(author: "George R. R. Martin", title: "A Song of Ice and Fire:
          4 A Feast for Crows").save(failOnError: true)
   }
```

```
15    }
16
17    def destroy = {
18    }
19  }
```

Con esto hemos redefinido *init* en el Servlet controlador principal de la aplicación. De esta forma cada vez que se inicie la aplicación se crearán estos datos como ejemplares del modelo.

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, para cada instancia de *Book* llamamos al método *Book.save()*, el cual guarda el objeto en la base de datos. En la llamada definimos 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 por defecto que emplea Grails, HSQLDB. Esta es una base de datos guardada en memoria principal, que sirve principalmente para desarrollo y pruebas. 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','
                    update'
17              url = "jdbc:hsqldb:mem:devDB"
18          }
19      }
20      test {
```

```
        dataSource {
            dbCreate = "update"
            url = "jdbc:hsqldb:mem:testDb"
        }
    }
    production {
        dataSource {
            dbCreate = "update"
            driverClassName = "com.mysql.jdbc.Driver"
            url = "jdbc:mysql://localhost/mi-proyecto"
            username="root"
            password="root"
        }
    }
}
```

En este caso estamos usando *MySql* a través del framework de persistencia *Hibernate*. El fichero DataSource.groovy define tres orígenes de datos diferentes para una misma aplicación: desarrollo, pruebas y producción.
Grails ya dispone de drivers para JDBC, pero es necesario descomentar varias líneas del fichero *BuildConfig.groovy*.

```
grails.project.class.dir = "target/classes"
grails.project.test.class.dir = "target/test-classes"
grails.project.test.reports.dir = "target/test-reports"
//grails.project.war.file = "target/${appName}-${appVersion}.war"
grails.project.dependency.resolution = {
    // inherit Grails' default dependencies
    inherits("global") {
        // uncomment to disable ehcache
        // excludes 'ehcache'
    }
    log "warn" // log level of Ivy resolver, either 'error', 'warn', 'info',
        'debug' or 'verbose'
    repositories {
        grailsPlugins()
        grailsHome()
        grailsCentral()

        // uncomment the below to enable remote dependency resolution
        // from public Maven repositories
        mavenLocal()
        mavenCentral()
        mavenRepo "http://snapshots.repository.codehaus.org"
        mavenRepo "http://repository.codehaus.org"
        mavenRepo "http://download.java.net/maven/2/"
        mavenRepo "http://repository.jboss.com/maven2/"
    }
    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 de dominio llamada *Author* que se relacione 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    String name
6    static constraints = {
7      name(blank:false)
8    }
9  }
```

Y por otra parte el controlador:

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

5.2. Relaciones entre clases con GORM

Ahora vamos a crear una relación 1:N unidireccional entre Author y Book. Para ello modificamos la clase *Author*:

```
1  package org.example
2
3  class Author {
4    static hasMany = [book:Book]
5    String name
6    String surname
7    static constraints = {
```

```
8       name(blank:false)
9       surname(blank:false)
10    }
11  }
```

En el caso de la clase *Book* no es necesario modificarla. Si la navegabilidad entre Book y Author fuera bidireccional, habría que añadir la siguiente línea de código a la clase Book:

```
1  static belongsTo = [author:Author]
```

5.3. Cambio de estilo en la vista

Grails genera una hoja de estilo por defecto 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>.

```
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  }
```

```
.menuButton a.home {
    background: url(../images/skin/house.png) center left no-repeat;
    color: #333;
    padding-left: 25px;
}
.menuButton a.list {
    background: url(../images/skin/database_table.png) center left no-repeat
        ;
    color: #333;
    padding-left: 25px;
}
.menuButton a.create {
    background: url(../images/skin/database_add.png) center left no-repeat;
    color: #333;
    padding-left: 25px;
}

/* MESSAGES AND ERRORS */

.message {
    background: #f3f8fc url(../images/skin/information.png) 8px 50% no-
        repeat;
    border: 1px solid #b2d1ff;
    color: #006dba;
    margin: 10px 0 5px 0;
    padding: 5px 5px 5px 30px
}

div.errors {
    background: #fff3f3;
    border: 1px solid red;
    color: #cc0000;
    margin: 10px 0 5px 0;
    padding: 5px 0 5px 0;
}
div.errors ul {
    list-style: none;
    padding: 0;
}
div.errors li {
  background: url(../images/skin/exclamation.png) 8px 0% no-repeat;
    line-height: 16px;
    padding-left: 30px;
}

td.errors select {
    border: 1px solid red;
}
td.errors input {
    border: 1px solid red;
}
```

```
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;
```

```
}

/* LIST */

.list table {
    border-collapse: collapse;
}
.list th, .list td {
    border-left: 1px solid #ddd;
}
.list th:hover, .list tr:hover {
    background: #b2d1ff;
}

/* PAGINATION */

.paginateButtons {
    background: #fff url(../images/skin/shadow.jpg) bottom repeat-x;
    border: 1px solid #ccc;
    border-top: 0;
    color: #666;
    font-size: 10px;
    overflow: hidden;
    padding: 10px 3px;
}
.paginateButtons a {
    background: #fff;
    border: 1px solid #ccc;
    border-color: #ccc #aaa #aaa #ccc;
    color: #666;
    margin: 0 3px;
    padding: 2px 6px;
}
.paginateButtons span {
    padding: 2px 3px;
}

/* DIALOG */

.dialog table {
    padding: 5px 0;
}

.prop {
    padding: 5px;
}
.prop .name {
    text-align: left;
    width: 15%;
    white-space: nowrap;
}
```

```
.prop .value {
    text-align: left;
    width: 85%;
}

/* ACTION BUTTONS */

.buttons {
    background: #fff url(../images/skin/shadow.jpg) bottom repeat-x;
    border: 1px solid #ccc;
    color: #666;
    font-size: 10px;
    margin-top: 5px;
    overflow: hidden;
    padding: 0;
}

.buttons input {
    background: #fff;
    border: 0;
    color: #333;
    cursor: pointer;
    font-size: 10px;
    font-weight: bold;
    margin-left: 3px;
    overflow: visible;
    padding: 2px 6px;
}
.buttons input.delete {
    background: transparent url(../images/skin/database_delete.png) 5px 50%
        no-repeat;
    padding-left: 28px;
}
.buttons input.edit {
    background: transparent url(../images/skin/database_edit.png) 5px 50% no
        -repeat;
    padding-left: 28px;
}
.buttons input.save {
    background: transparent url(../images/skin/database_save.png) 5px 50% no
        -repeat;
    padding-left: 28px;
}
```