Codi CMS

Descripció aspectes importants CMS

09/04/2015

Versió	Autor	Data	Revisió	Comentaris
1.0	Jesús Campos Muñoz	9-Abril -2015		Inici documentació.

Índex

1 Introducció 6

2 Canvis comuns (Editor i Visor) 7

2.1 Incloure Bootstrap en una aplicació Grails 7

2.1.1 Instal·lació de BootStrap 7

2.1.2 Bootstrap: exemple senzill d'ús 9

2.2 Incloure Spring Security CAS i Spring Security Core en una aplicació Grails .. 9

2.2.1 Arxiu BuildConfig.groovy 9

2.2.2 Modificacions al plugin 10

2.2.3 Arxiu Config.groovy 11

2.2.4 Dominis i controladors 13

2.2.5 Com fer servir els plugins de seguretat 14

2.3 Eliminar Spring Security CAS i Spring Security Core d'una aplicació Grails .. 14

2.3.1 Arxiu Config.grrovy 14

2.3.2 Arxiu BuildConfig.groovy 15

2.3.3 Controladors 15

2.4 Canviar el nom del servidor d'una aplicació Grails 16

2.5 Arxius CSS i Javascript 16

3 Aplicació Editor (gcrrhh) 18

3.1 TreeTable 18

3.1.1 Instalació del plugin TreeTable 18

3.1.2 Ús del plugin TreeTable 19

3.1.2.1 Exemple bàsic 19

3.1.2.2 Exemple complex 21

3.2 CKEditor 21

4 Aplicació Visor (rrhh) 22

5 Annexos 23

5.1 Activar pestanya *Markers* 23

5.2 Activar pestanya *Navigation* 23

6 Referències 24

6.1 Bootstrap 24

6.2 Spring Security Core 24

Imatges

Imatge 1: Esquelet aplicació Grails amb Bootstrap 8
Imatge 2: Imatge de Bootstrap d'un exemple bàsic 9
Imatge 3: Detecció de problemes del plugin Srping Security Core amb la pestanya Markers 10

Figures

Figura 1: Referenciar BootSatrap en una aplicació Grails 8
Figura 2: Codi d'exemple d'ús de Bootstrap 9
Figura 3: Instal·lació del plugin Spring Security Core 10
Figura 4: Instal·lació del plugin Spring Security CAS 10
Figura 5: Ruta de ReflectionsUtils.groovy d'Spring Security Core a la pestanya Navigator 11
Figura 6:Ruta de ReflectionsUtils.groovy d'Spring Security Core al sistema de fitxers 11
Figura 7: Import erroni a ReflectionsUtils.groovy 11
Figura 8: Import correcte a ReflectionsUtils.groovy 11
Figura 9: Instrucció incorrecta a ReflectionsUtils.groovy 11
Figura 10: Instrucció correcta a ReflectionsUtils.groovy 11
Figura 11: Configuració en comú de seguretat dels diferents entorns 12
Figura 12: Configuració de seguretat específica per entorns 12
Figura 13: Nom del servidor d'exemple 12
Figura 14: Possible error al configurar Spring Security 13
Figura 15: *Config.grrovy* eliminar plugins de seguretat 14
Figura 16: BuildConfig.grrovy eliminar plugins de seguretat 15
Figura 17: aplication.properties eliminar plugins de seguretat 15
Figura 18: Canviar el nom del servidor 16
Figura 19: Canviar el nom del servidor a Grails 2.4.3 16

1 Introducció

El gestor de continguts o CMS (Content Management System) dissenyat està format per dues aplicacions:

- L'aplicació ***gcrrhh*** o ***Editor*** (d'ara en endavant l'Editor): serveix per construir pàgines web basades en un determinat esquelet i decoració, que poden ser triats per l'usuari en funció del recursos disponibles. Aquestes pàgines es guarden en bases de dades.
- L'aplicació ***rrhh*** o ***Visor*** (d'ara en endavant el Visor): serveix per visualitzar les pàgines generades amb l'Editor.

Aquestes dues aplicacions estan programades fent servir el *framework* d'aplicacions web Grails i diferents plugins de la plataforma (com pot ser l'editor de text **CKEditor**) o *frameworks* addicionals (com és **Bootstrap**). Els plugins tenen unes determinades configuracions que cal tenir en compte per tal que es comportin com nosaltres volem, modificant el seu que tenen per defecte. A més, el *framworks* s'han d'instal·lar i configurar que poder utilitzar-los.

En les webs d'aquests complements s'explica com configurar-los, però a vegades es donen aspectes per soposats que no són tan directes de veure. En aquesta guia es pretén explicar com realitzar les instal·lacions i canvis adients per tal que tots el complements quedin configurats com volem, és a dir, com estan a l'actual CMS. En alguns casos fins i tot posant exemples senzills que es poden dur a terme en una aplicació Grails desde zero per tal de veure el seu comportament.

TAMBE ES PRETENEN DONAR DETALLS TECNICS (REALIZAR EL BUSCADOR, DETALLES DE CODIGO ETC)

2 Canvis comuns (Editor i Visor)

Es detallen els canvis que s'han realitzat tant a l'Editor com al Visor. El framework de visualització per plataformes mòbils, Bootstrap, i els de seguretat, CAS i Spring Security Core es fan servir a ambdues aplicacions.

2.1 Incloure Bootstrap en una aplicació Grails

Bootstrap és un *framework* que serveix per dissenyar aplicacions de movilitat. És l'encarregat, ajudant-se d'un conjunt d'arxius Javascript i CSS, d'adaptar els elements que de la pàgina web al dispositiu mòvil.

2.1.1 Instal·lació de BootStrap

Descarregar el framework de la seva pàgina web: http://getbootstrap.com/getting-started/#download

És aconsellable descarregar la versió que es troba a : *"Bootstrap: Compiled and minified CSS, JavaScript, and fonts. No docs or original source files are included"*.

Només faran falta els següents arxius que es troben a la descompressió:

- css/bootstrap.css
- js/bootstrap.min.js
- La carpeta *fonts*, per poder fer servir el *glyphicons*.

Aquests, es copien en les següents ubicacions dintre del projecte Grails:

- web-app/css/bootstrap.css
- web-app/js/bootstrap.min.js
- web-app/fonts/

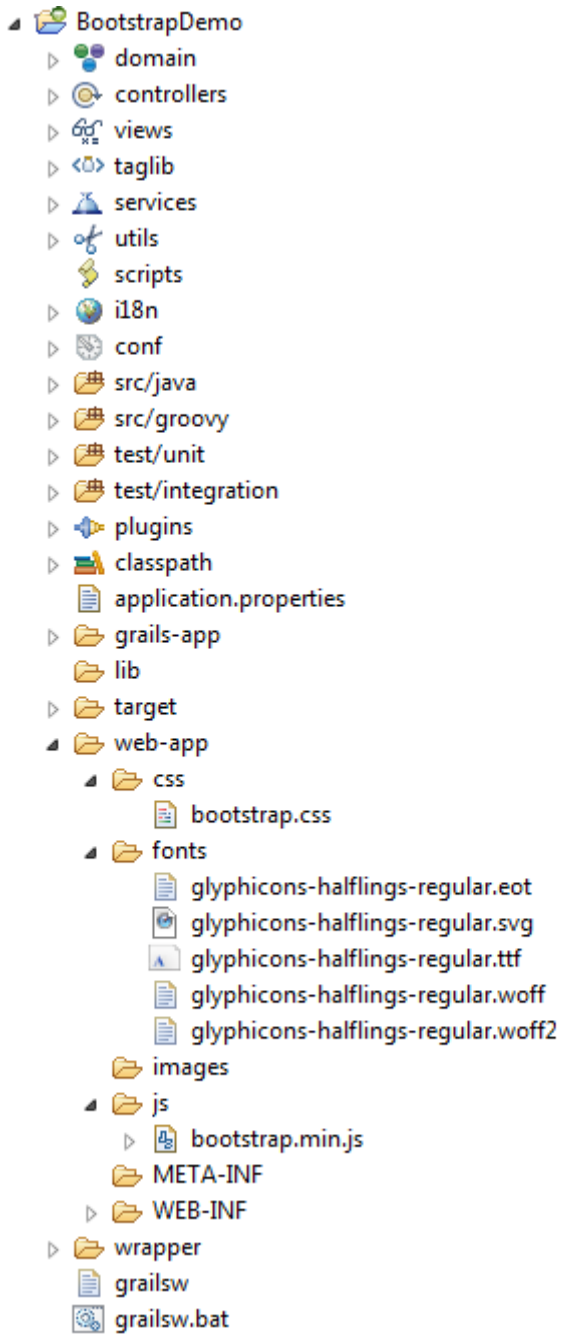
Per últim, s'han de referenciar els arxius JS i CSS a les vistes GSP. Generalment, el Bootstrap es farà servir a tota l'aplicació per la qual cosa la millor opció es fer les referències en una GSP que sigui un *layout*. Llavors al *layout* ***main.gsp*** s'ha d'insertar el següent codi:

```
<head>
   ...
   <link rel="apple-touch-icon" sizes="114x114" href="${assetPath(src:
      'apple-touch-icon-retina.png')}">
   <link rel="stylesheet"
      href="/${grails.util.Metadata.current.'app.name'}/css/bootstrap.css"
      type="text/css">
   <script type="text/javascript"
      src="/${grails.util.Metadata.current.'app.name'}/js/bootstrap.min.js"></
      script>
   <asset:stylesheet src="application.css"/>
   ...
```

```
</head>
```

Figura 1: Referenciar BootSatrap en una aplicació Grails

A la següent imatge es pot veure com quedaria el projecte amb les carpetes corresponents a l'editor de codi GGTS:



Imatge 1: Esquelet aplicació Grails amb Bootstrap

2.1.2 Bootstrap: exemple senzill d'ús

Per tal de comprovar que s'han referenciat de manera correcta els arxius del *framework* Bootstrap, el següent codi es pot incloure a una vista GSP:

```
<html>
 <head>
    <meta name="layout" content="main" />
    <title>Proba Bootstrap</title>
 </head>
 <body>
    <p>
     <button type="button" class="btn btn-lg btn-default">Default</button>
     <button type="button" class="btn btn-lg btn-primary">Primary</button>
     <button type="button" class="btn btn-lg btn-success">Success</button>
     <button type="button" class="btn btn-lg btn-info">Info</button>
     <button type="button" class="btn btn-lg btn-warning">Warning</button>
     <button type="button" class="btn btn-lg btn-danger">Danger</button>
     <button type="button" class="btn btn-lg btn-link">Link</button>
    </p>
 </body>
</html>
```

Figura 2: Codi d'exemple d'ús de Bootstrap

I s'hauria de visualitzar una web molt similar a la següent:

Imatge 2: Imatge de Bootstrap d'un exemple bàsic

2.2 Incloure Spring Security CAS i Spring Security Core en una aplicació Grails

Per instal·lar el CAS, que serveix per fer autenticar-se en diferents aplicacions només amb una única identificació, i l'Spring Security Core, que serveix per obtenir les credencials, es parteix de que alguns arxius com poden ser controladors o dominis s'extreuran d'altres aplicacions.

2.2.1 Arxiu BuildConfig.groovy

Afegir en la closure **plugins** l'Spring Security Core:

```
plugins {
   ...
   compile ":spring-security-core:1.2.7.3"
   ...
}
```

Figura 3: Instal·lació del plugin Spring Security Core

Després, sobre el projecte fer botó dret → *Grails Tools* → *Refresh depencies* (Alt+G,R) per actualitzar les dependències del projecte.

Ara afegir a la closure **plugins** l'Spring Security CAS:

```
plugins {
   ...
   compile ":spring-security-core:1.2.7.3"
   compile ':spring-security-cas:1.0.5'
   ...
}
```

Figura 4: Instal·lació del plugin Spring Security CAS

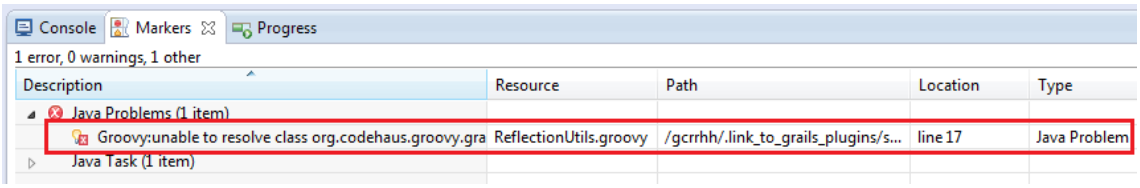
I tornar a refrescar dependències. Independentment de si s'obtenen errors o no a la a la consola del GGTS es reinicia l'entorn de desenvolupament.

2.2.2 Modificacions al plugin

Un cop reiniciat l'entorn, si s'està fent servir la versió 2.4.3 de Grails o per sobre s'obtindrà un error al codi del plugin que s'ha descarregat.

És fàcil veure que hi ha un problema al projecte perquè apareix una X vermella, però no ho és tant detectar on.

Una opció és mirar a la pestanya de *Markers* (si no està activada consultar **5.1 Activar pestanya *Markers***) on s'ha produït l'error, com es mostra a la següent imatge:



Imatge 3: Detecció de problemes del plugin Srping Security Core amb la pestanya Markers

A la pestanya *Markers* s'ens mostra l'arxiu conflictiu, indicant la seva ruta per poder localitzar-lo i en quina aplicació està causant el problema. Peró de manera molt senzilla, fent doble click sobre *Groovy unable to resolve...*, d'aquesta manera s'obrirà a l'editor l'arxiu problemàtic.

Una altra és anar directament a l'arxiu en concret ja que és un error detectat (arxiu ***ReflectionsUtils.groovy***) entre la versió de Grails 2.4.3 i superiors i Spring Security Core. La ruta de l'arxiu és:

```
/gcrrhh/.link_to_grails_plugins/spring-security-core-
```

```
1.2.7.3/src/groovy/org/codehaus/groovy/grails/plugins/springsecurity/ReflectionUtils.groovy
```

Figura 5: Ruta de ReflectionsUtils.groovy d'Spring Security Core a la pestanya Navigator

Per accedir desde l'entorn de desenvolupament s'ha de fer a través de la pestanya *Navigator* (si no està activada consultar **5.2 Activar pestanya *Navigation***).

Per accedir desde el sistema de fitxers:

```
dir_workspace\dir_aplicacio\target\work\plugins\spring-security-core-1.2.7.3\src\groovy\org\codehaus\groovy\grails\plugins\springsecurity\ReflectionUtils.groovy
```

Figura 6:Ruta de ReflectionsUtils.groovy d'Spring Security Core al sistema de fitxers

Llavors, dintre de l'arxiu ReflectionsUtils substituir:

```
import org.codehaus.groovy.grails.commons.ApplicationHolder
```

Figura 7: Import erroni a ReflectionsUtils.groovy

per:

```
import grails.util.Holders
```

Figura 8: Import correcte a ReflectionsUtils.groovy

Substituir:

```
application = ApplicationHolder.application
```

Figura 9: Instrucció incorrecta a ReflectionsUtils.groovy

per:

```
application = Holders.grailsApplication
```

Figura 10: Instrucció correcta a ReflectionsUtils.groovy

2.2.3 Arxiu Config.groovy

Al final d'aquest arxiu s'han d'incloure les següents línies:

```
// Added by the Spring Security Core plugin:
grails.plugins.springsecurity.userLookup.userDomainClassName='Security.SecUser'
grails.plugins.springsecurity.userLookup.authorityJoinClassName= 'Security.SecUserSecRole'
grails.plugins.springsecurity.authority.className = 'Security.SecRole'
grails.plugins.springsecurity.securityConfigType = 'Annotation'
grails.plugins.springsecurity.rejectIfNoRule = false
grails.plugins.springsecurity.cas.active = true
grails.plugins.springsecurity.cas.sendRenew = false
grails.plugins.springsecurity.cas.serverUrlEncoding = 'UTF-8'
grails.plugins.springsecurity.cas.key = 'grails-spring-security-cas'
grails.plugins.springsecurity.cas.artifactParameter = 'ticket'
grails.plugins.springsecurity.cas.serviceParameter = 'service'
```

```
grails.plugins.springsecurity.cas.filterProcessesUrl = '/j_spring_cas_security_check'
```

Figura 11: Configuració en comú de seguretat dels diferents entorns

Després, per a cada entorn (dev, test i prod) s'han de configurar les següents línies:

```
environments {
    ...
    dev { //Per exemple
grails.plugins.springsecurity.cas.loginUri = '/login'
grails.plugins.springsecurity.cas.serviceUrl =
"http://localhost:8099/${grails.util.Metadata.current.'app.name'}/j_spring_cas_security_check"
grails.plugins.springsecurity.cas.serverUrlPrefix = 'https://cas.upc.edu/'
grails.plugins.springsecurity.cas.proxyCallbackUrl =
"http://localhost:8099/${grails.util.Metadata.current.'app.name'}/secure/receptor"
grails.plugins.springsecurity.cas.proxyReceptorUrl = '/secure/receptor'
grails.plugins.springsecurity.cas.useSingleSignOut = 'true'
grails.plugins.springsecurity.logout.afterLogoutUrl =
"https://cas.upc.edu/logout?url=http://localhost:8099/${grails.util.Metadata.current.'app.name'}"
        ...
    }
    ...
}
```

Figura 12: Configuració de seguretat específica per entorns

Per fer-ho, s'ha suposat que a desenvolupament es té la següent configuració:

```
environments {
    ...
    dev {
        ...
        grails.serverURL = http://localhost:8099/${grails.util.Metadata.current.'app.name'}
        ...
    }
    ...
}
```

Figura 13: Nom del servidor d'exemple

On /${grails.util.Metadata.current.'app.name'} indica el nom de l'aplicació de manera genèrica en comptes de posar *rrhh* o *gcrrhh*. Això és útil per quan una aplicació no te un nom definitiu.

S'ha d'anar amb compte l'entorn que e configura, ja que habitualment els entorns de test i producció acostumen a tenir una connexió de tipus https, mentres que a

desenvolupament és http. Es podria donar el següent cas (suposant la configuració de la Figura 13: Nom del servidor d'exemple):

```
1  grails.plugins.springsecurity.cas.loginUri = '/login'
2  grails.plugins.springsecurity.cas.serviceUrl =
3  "https://localhost:8090/${grails.util.Metadata.current.'app.name'}/j_spring_cas_security_c
4  heck"
5  grails.plugins.springsecurity.cas.serverUrlPrefix = 'https://cas.upc.edu/'
6  grails.plugins.springsecurity.cas.proxyCallbackUrl =
7  "https://localhost:8090/${grails.util.Metadata.current.'app.name'}/secure/receptor"
8  grails.plugins.springsecurity.cas.proxyReceptorUrl = '/secure/receptor'
9  grails.plugins.springsecurity.cas.useSingleSignOut = 'true'
10 grails.plugins.springsecurity.logout.afterLogoutUrl
11 ="https://cas.upc.edu/logout?url=https://localhost:8090/${grails.util.Metadata.current.'app
12 .name'}"
```

Figura 14: Possible error al configurar Spring Security

A les línies 2-3, 6-7 i 10-11 hi han erros. Estan intentant accedir mitjançant https quan la configuració és http.

2.2.4 Dominis i controladors

Ara s'haurien de crear els dominis del package **Security** (*SecRole, SecUser, SecUserSecRole*) i els controladors del package **Security** (*Login, Logout*).

Llavors, per no haver de crear i configurar desde zero els dominis i els controlador es copien d'altre projecte tenint cura de el package Security quedi com al projecte del que es copien.

Una vegada s'han copiat els dominis cal modificar a la closure *static mapping* la variable *table* que hi ha dintre de cada un d'ells per tal d'apuntar a les taules correctes.

Com que la inclusió de determinades parts de la seguretat es fa per herència en altres aplicacions, extenent de la classe ***BaseCtronler.groovy***, la solució és copiar els arxius necessaris d'altres aplicacions com pugin ser Tempus (RLG), Salut Cardiovascular (SAL), etc ...

Primer de tot s'agafa el package **utils** (src/groovy/utils) d'altre projecte i es copia al nostre en la mateixa ubicació. Aquest package té cinc arxiu: **BaseControler.*groovy*, Crypt.*groovy*, Database.*groovy*, FileUploader.*groovy*, Validation.*groovy***.

2.2.5 Com fer servir els plugins de seguretat

Fins ara s'ha explicat quins són els plugin necessaris per tenir seguretat a la nostra aplicació i com configurar-los.

Ara s'explicarà com fer-los servir per tal de prohibir la visualització de vistes i/o controladors a determinats perfils.

Per tal que un controlador tingui en compte el rol de l'usuari:

- Ha d'extendre/heretar de ***BaseControler.groovy***
- Ha d'importar:
  - utils.BaseControler
  - grails.plugins.springsecurity.Secured
- Abans del nom de la classe ha de tenir la sentència: `@Secured(['ROLE'])`, on 'ROLE' ha de ser el rol d'usuari a vigilar (ROLE_ADMIN, ROLE_USER, etc...)
- En l'acció en que accedim per primera vegada al controlador s'ha d'executar la instrucció: this.populateUserName().

2.3 Eliminar Spring Security CAS i Spring Security Core d'una aplicació Grails

Una vegada l'aplicació està ben configurada amb els plugins de seguretat pot ser necessari per segons quin tipus de proves eliminar-los, com per exemple a les proves de càrrega o estrés per tal d'accedir-hi sense indentificar-se.

Com que desfer tot els passos de l'apartat anterior (**2.2 Incloure Spring Security CAS i Spring Security Core en una aplicació Grails**) pot ser una mica tediós i no totes les aplicacion estan fetes de la mateixa manera, en aquest apartat es descriu d'una manera ràpida i genèrica.

2.3.1 Arxiu Config.grrovy

En els respectius entorns comentar totes les línies que comencin per: *grails.plugins.springsecurity.cas*.

Comentar les línies del final de l'arxiu:

```
//// Added by the Spring Security Core plugin:
//grails.plugins.springsecurity.userLookup.userDomainClassName =
 'Security.SecUser'
//grails.plugins.springsecurity.userLookup.authorityJoinClassName =
 'Security.SecUserSecRole'
//grails.plugins.springsecurity.authority.className = 'Security.SecRole'
//grails.plugins.springsecurity.securityConfigType = 'Annotation'
//grails.plugins.springsecurity.rejectIfNoRule = false
```

Figura 15: *Config.grrovy* eliminar plugins de seguretat

2.3.2 Arxiu BuildConfig.groovy

Comentar a la closure **plugins** les següents línies:

```
plugins {
   ...
//     compile ":spring-security-core:1.2.7.3"
//     compile ':spring-security-cas:1.0.5'
   ...
}
```

Figura 16: BuildConfig.grrovy eliminar plugins de seguretat

En el cas d'aplicacions realitzades amb la versión de Grails 2.1.1 s'ha de desinstal·lar mitjançant el **Plugin Manager** (botó dret sobre plugins a l'arbre del projecte en el GGTS, *Open Grails Plugin Manager*, seleccionar el plugin i desinstlar-ho) o bé comentar les següents línies a l'arxiu ***aplications.properties***:

```
#Grails Metadata file
#Wed Jan 22 12:22:56 CET 2014
app.grails.version=2.1.1
...
#plugins.spring-security-cas=1.0.5
#plugins.spring-security-core=1.2.7.3
```

Figura 17: aplication.properties eliminar plugins de seguretat

2.3.3 Controladors

S'ha d'eliminar la següent sentència en tots el controladors on estigui declarada: `@Secured(['ROLE'])`, on 'ROLE' és el rol d'usuari a vigilar (ROLE_ADMIN, ROLE_USER, etc...).

Eliminar import grails.plugins.springsecurity.Secured.

A partir d'ara hi ha dues opcions:

- 1ª opció: Evitar que els controladors heretin de ***BaseControler.groovy*** i eliminar les sentències:
  - this.populateUserName()
  - this.doSecurityCheck()
- 2ª opció: Modificar ***BaseControler.groovy*** per tal de no fer més modificacions als controladors. S'ha de fer un hardcode de les dades degut a que es necessita el PERNR de l'usuari i el seu nom per fer-lo servir (s'han de fer servir uns que ja existeixin a la base de dades). A /src/groovy/utils/BaseControler.groovy s'han de canviar els següents métodes:
  - def populateUserName() queda de la següent manera:

```
def populateUserName()
```

```
{
        session["user"] =   "jesus.campos"
}
```

- Afegir a l'inici del doSecurityCheck():

  session.user="jesus.campos"

2.4 Canviar el nom del servidor d'una aplicació Grails

Habitualment per canviar el nom del servidor de *localhost* a *pc24094.upc.es*, per exemple, s'ha de modificar la closure ***environments*** de l'entorn corresponent a l'arxiu **Config.groovy** de la següent manera:

```
environments {
      ...
      test {
         ...
         grails.serverURL = "http://pc24094.upc.es:8099/
         ${grails.util.Metadata.current.'app.name'}"
         ...
      }
      ...
}
```

Figura 18: Canviar el nom del servidor

En el cas de Grails 2.4.3 això no és suficient, a més a l'arxiu **BuildConfig.groovy** s'ha de comentar les següents línies dintre de la closure ***grails.project.fork***:

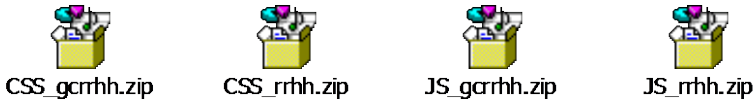
```
grails.project.fork = [
       ...
       // configure settings for the run-app JVM
//     run: [maxMemory: 768, minMemory: 64, debug: false, maxPerm: 256,
forkReserve:false],
       ...
]
```

Figura 19: Canviar el nom del servidor a Grails 2.4.3

2.5 Arxius CSS i Javascript

Tant pel que fa a l'Editor com al Visor es fan servir diferents arxius CSS, alguns dels qual son personalitzacions dels CSS del *framework* Bootstrap. També es fan servir arxius Javascript.

Als següents arxius compressos hi ha un recull dels més importants, inclosa la configuració del plugin Ckeditor.


CSS_gcrrhh.zip
CSS_rrhh.zip
JS_gcrrhh.zip
JS_rrhh.zip

3 Aplicació Editor (gcrrhh)

El nom genèric de l'aplicació al servidor és gcrrhh i és coneguda com a l'Editor. Serveix per construir pàgines web amb un determinat format aplicacble per part de l'usuari, els Templates. Els Templates serveixen per modifcar el marc de la pàgina, és a dir, capceleres i peus de pàgina de pàgina que es veuran, etc. També existeixen un formats anomenats Plantilles, els quals faciliten la tasca de la creació del contingut i permeten seguir un determinat patró de disseny al temps que estalvia a l'editor de contingut (referint-se al dissenyador) el codi CSS i JS per realitzar element amb moviment (desplegables).

3.1 TreeTable

Plugin per visualitzar el contingut de l'Editor de manera ordenada.

3.1.1 Instalació del plugin TreeTable

Descarregar l'arxiu comprimit (opció **"Download Now"**) del següent enllaç: https://plugins.jquery.com/treetable/

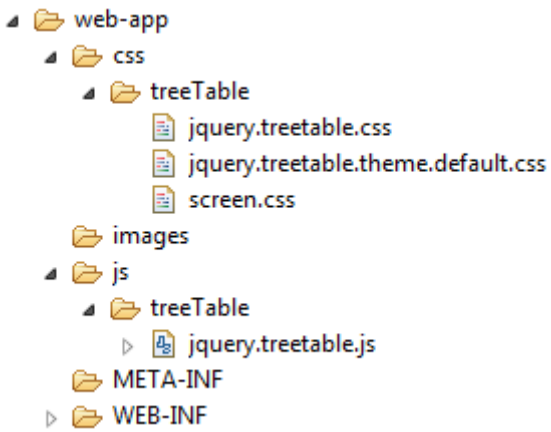
Només faran falta els següents arxius que es troben a la descompressió:

- El arxius continguts a la carpeta css.
- L'arxiu jquery.treetable.js de l'arrel.

Aquests, es copien en les següents ubicacions dintre del projecte Grails:

- El arxius continguts a la carpeta css a la carpeta web-app/css/treeTable
- web-app/css/treeTable/jquery.treetable.js

A la següent imatge es pot veure com quedaria el projecte amb les carpetes corresponents a l'editor de codi GGTS:



Imatge 4: Esquelet aplicació Grails amb TreeTable

Per últim, s'han de referenciar els arxius JS i CSS a les vistes GSP. Generalment, el TreeTable es farà servir només per visualitzar tots els continguts que es poden editar, però no quan s'estigui editant una pàgina. Per referenciar el plugin TreeTable en un la un *layout* (a l'aplicació es troba al ***main.gsp***) s'ha d'insertar el següent codi:

```
<head>
      ...
      <meta name="layout" content="main" />
      <link
          href="/${grails.util.Metadata.current.'app.name'}/css/treeTable/jque
          ry.treetable.css" rel="stylesheet" type="text/css" />
      <link
          href="/${grails.util.Metadata.current.'app.name'}/css/treeTable/jque
          ry.treetable.theme.default.css" rel="stylesheet" type="text/css" />
      <script
          src="/${grails.util.Metadata.current.'app.name'}/js/treeTable/jquery
          .treetable.js">
      </script>
      ...
```

Figura 20: Referenciar TreeTable en una aplicació Grails

3.1.2 Ús del plugin TreeTable

Per mostrar el comportament del plugin es mostren dos exemples:

- El bàsic és per veure simplement el seu funcionament.
- L'exemple complex serveix per veure com està implementat realment en l'aplicació de l'Editor. Aquest descriu detalladament la configuració a l'aplicació.

3.1.2.1 Exemple bàsic

Per veure la funcionalitat del plugin només cal copiar el següent codi a una pàgina GSP que tingui les referències de l'apartat anterior:

```
<table id="example-basic-expandable">
      <tr class="blank" data-tt-id="blank00">
            <td>
                  <span class="blank">File 1</span>
            </td>
            <td>File #1</td>
            <td></td>
      </tr>
      <tr class="branch" data-tt-id="00">
            <td>
                  <span class="file">File 2</span>
            </td>
            <td>File #2</td>
            <td></td>
      </tr>
      <tr class="branch" data-tt-id="01">
            <td>
                  <span class="folder">Folder 1</span>
```

```
            </td><td>Folder #1</td>
            <td>--</td>
      </tr>
      <tr class="branch" data-tt-id="02" data-tt-parent-id="01">
            <td>
                  <span class="folder">Folder 2</span>
            </td><td>Folder #2</td>
            <td>--</td>
      </tr>
      <tr class="branch" data-tt-id="03" data-tt-parent-id="02">
            <td>
                  <span class="folder">Folder 3</span>
            </td><td>Folder #3</td>
            <td>--</td>
      </tr>
      <tr class="branch" data-tt-id="04" data-tt-parent-id="02">
            <td>
                  <span class="file">File 3</span>
            </td><td>File #3</td>
            <td>--</td>
      </tr>
</table>

<script>
      $("#example-basic-expandable").treetable({ expandable: true });
</script>
```

Figura 21: Codi exemple senzill amb TreeTable

A l'executar el codi anterior s'obté al navegador web:

Figura 22: Exemple sencill amb TreeTable. Vista 1

Expandint el node *Folder 1* que té fills:

Figura 23: Exemple sencill amb TreeTable. Vista 2

A partir d'ara es descriu el codi. Primer de tot cal dir que el plugin es basa en incloure-ho tot en una taula. L'atribut data-tt-id de cada tag *tr* serveix d'atribut id, és a dir, ha de ser únic. Aquest atribut és útil en el cas de voler fer que un element contingui d'altres

en el seu interior. Com pot ser el cas del *Folder 2*, que té dintre *Folder 3* i *File 3*, llavors *Folder 2* seria l'element pare i elas altres els fills. Per exemple, si es vol que un element estigui dintre d'altre, l'element fill ha de tenir l'atribut data-tt-parent-id on s'indicarà el valor de l'atribut data-tt-id de l'element pare. En aquest exemple el valor data-tt-parent-id del elements és *02*, que coincideix amb el valor de data-tt-id del *Folder 2*.

En aquest cas la fila es divideix en tres columnes, això pot variar. La imatge del triangle indicant que l'element té fills apareix automàticament

3.1.2.2 Exemple complex

3.2 CKEditor

Plugin que serveix per editar el contingut en mode WYSIWYG, és a dir, el que és veu a la pantalla d'edició és el que veurà l'usuari a la web final

4 Aplicació Visor (rrhh)

5 Annexos

5.1 Activar pestanya *Markers*

Si la pestanya *Markers* no està activada cal anar a *Window* → *Show View* → *Other...* i escriure *Markers*, acte seguit apareixerà la paraula *Markers* a sota de la carpeta *General.*

5.2 Activar pestanya *Navigation*

Si no està activada cal anar a *Window* → *Show View* → *Other...* i escriure *Navigator*, acte seguit apareixerà la paraula *Navigator* a sota de la carpeta *General.*

6 Referències

6.1 Bootstrap

Manual de Bootstrap: https://librosweb.es/libro/bootstrap_3/

6.2 Spring Security Core

Explicació de com crear els dominis i els controladors que a l'apartat **2.2 Incloure Spring Security CAS i Spring Security Core en una aplicació Grails** s'han suggerit que era millor copiar-los d'altres aplicacions: http://spring.io/blog/2010/08/11/simplified-spring-security-with-grails/