|  |
| --- |
| Codi CMS |
| Descripció aspectes importants CMS |
|  |
|  |
| 09/04/2015 |

**Índex**

[1 Introducció 3](#_Toc416344341)

[2 Canvis comuns (Visor i Editor) 4](#_Toc416344342)

[2.1 Incloure BootStrap en una aplicació Grails 4](#_Toc416344343)

[2.1.1 Instal·lació de BootStrap 4](#_Toc416344344)

[2.2 Incloure CAS + Spring Security Core en una aplicació Grails 6](#_Toc416344345)

[2.3 Canviar el nom del servidor d’una aplicació Grails 6](#_Toc416344346)

[3 Aplicació Editor (gcrrhh) 7](#_Toc416344347)

[3.1 TreeTable 7](#_Toc416344348)

[3.2 CKEditor 7](#_Toc416344349)

[4 Aplicació Visor (rrhh) 8](#_Toc416344350)

**Imatges**

[Imatge 1: Esquelet aplicació Grails amb Bootstrap 5](#_Toc416348508)

[Imatge 2: Imatge de Bootstrap d'un exemple bàsic 6](#_Toc416348509)

**Figures**

[Figura 1: Referenciar BootSatrap en una aplicació Grails 5](#_Toc416348512)

[Figura 2: Codi d'exemple d'ús de Bootstrap 6](#_Toc416348513)

[Figura 3: Instal·lació del plugin Spring Security Core 7](#_Toc416348514)

[Figura 4: Instal·lació del plugin Spring Security CAS 7](#_Toc416348515)

# 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)

# 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.

## 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.

### 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

### 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

## Incloure CAS + 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.

### 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** el 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

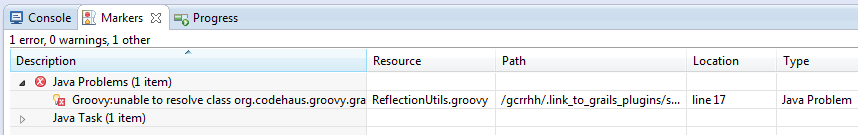
I tornar a refrescar dependències. Independentment de si s’obtenen errors o no a la a la consola del GGTS es reinicia l’entorn de desenvolupament.

### 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, peró no ho és tant detectar on.

Una opció és mirar a la pestanya de *Markers* (si 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*) 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 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 |

Per accedir desde l’entorn de desenvolupament s’ha de fer a través de la pestanya *Navigator* (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*).

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 |

Llavors, dintre de l’arxiu ReflectionsUtils substituir:

|  |
| --- |
| import org.codehaus.groovy.grails.commons.ApplicationHolder |

per:

|  |
| --- |
| import grails.util.Holders |

Substituir:

|  |
| --- |
| application = ApplicationHolder.application |

per:

|  |
| --- |
| application = Holders.grailsApplication |

### 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 5: 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 6: 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 7: 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*. Aixó 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 7):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | grails.plugins.springsecurity.cas.loginUri = '/login'  grails.plugins.springsecurity.cas.serviceUrl =  "https://localhost:8090/${grails.util.Metadata.current.'app.name'}/j\_spring\_cas\_security\_c  heck"  grails.plugins.springsecurity.cas.serverUrlPrefix = 'https://cas.upc.edu/'  grails.plugins.springsecurity.cas.proxyCallbackUrl =  "https://localhost:8090/${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=https://localhost:8090/${grails.util.Metadata.current.'app  .name'}" |

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

### Altres canvis

Copiar de otras aplicaciones.

## 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'}"  ...  }  ...  } |

En el cas de Grails 2.3.4 aixó no és suficient, a més a l’arxiu **BuildConfig.groovy** s’ha de comentar les egüents 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],  ...  ] |

# Aplicació Editor (gcrrhh)

El nom genèric de l’aplicació al servidor és gcrrhh i és coneguda com a l’Editor

## TreeTable

Plugin per visualitzar el contingut de l’Editor de manera ordenada

## CKEditor

Plugin que serveix per editar el contingut en mode [WYSIWYG](http://ca.wikipedia.org/wiki/WYSIWYG), és a dir, el que és veu a la pantalla d’edició és el que veurà l’usuari a la web final

# Aplicació Visor (rrhh)