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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versió** | **Autor** | **Data** | **Revisió** | **Comentaris** |
| 1.0 | Jesús Campos Muñoz | 09-04-2015 |  | Inici documentació. |
| 1.1 | Jesús Campos Muñoz | 20-04-2015 |  | Inclusió al manual de les aplicacions RPPAS, RPPDI i RPBEC. |

**Índex**

[1 Introducció 10](#_Toc417044472)

[2 Canvis comuns (Editor i Visor) 11](#_Toc417044473)

[2.1 Incloure Bootstrap en una aplicació Grails 11](#_Toc417044474)

[2.1.1 Instal·lació de BootStrap 11](#_Toc417044475)

[2.1.2 Bootstrap: exemple senzill d’ús 13](#_Toc417044476)

[2.2 Incloure Spring Security CAS i Spring Security Core en una aplicació Grails 13](#_Toc417044477)

[2.2.1 Arxiu BuildConfig.groovy 13](#_Toc417044478)

[2.2.2 Modificacions al plugin 14](#_Toc417044479)

[2.2.3 Arxiu Config.groovy 15](#_Toc417044480)

[2.2.4 Dominis i controladors 17](#_Toc417044481)

[2.2.5 Com fer servir els plugins de seguretat 18](#_Toc417044482)

[2.3 Eliminar Spring Security CAS i Spring Security Core d'una aplicació Grails 18](#_Toc417044483)

[2.3.1 Arxiu Config.grrovy 18](#_Toc417044484)

[2.3.2 Arxiu BuildConfig.groovy 19](#_Toc417044485)

[2.3.3 Controladors 19](#_Toc417044486)

[2.3.4 Desfer els canvis (re-instal·lar CAS) 20](#_Toc417044487)

[2.4 Canviar el nom del servidor d’una aplicació Grails 20](#_Toc417044488)

[2.5 Arxius CSS i Javascript 21](#_Toc417044489)

[3 Aplicació Editor (gcrrhh) 22](#_Toc417044490)

[3.1 TreeTable 22](#_Toc417044491)

[3.1.1 Instal·lació del plugin TreeTable 22](#_Toc417044492)

[3.1.2 Ús del plugin TreeTable 23](#_Toc417044493)

[3.1.2.1 Exemple bàsic 23](#_Toc417044494)

[3.1.2.2 Exemple complex 25](#_Toc417044495)

[3.2 CKEditor 26](#_Toc417044496)

[3.2.1 Instal·lació i configuració del plugin CKEditor 27](#_Toc417044497)

[3.2.2 Arxiu de configuració config.js 28](#_Toc417044498)

[3.2.2.1 Personalitzar l'idioma 31](#_Toc417044499)

[3.2.2.2 Personalitzar la barra d'eines 31](#_Toc417044500)

[3.2.2.3 Botó de indentació 32](#_Toc417044501)

[3.2.2.4 Fulles d'estils aplicades a l’àrea d’edició 32](#_Toc417044502)

[3.2.2.5 Indentació del codi HTML a l'enviar les dades al controlador 33](#_Toc417044503)

[3.2.2.6 Evitar que els estils i atributs desapareguin 36](#_Toc417044504)

[3.2.3 Arxiu Config.groovy 36](#_Toc417044505)

[3.2.4 Pujar arxius al servidor amb CKEditor 37](#_Toc417044506)

[3.2.4.1 Modificació al plugin de la ruta de pujada d’arxius al servidor 41](#_Toc417044507)

[3.2.5 Codemirror per a rich editor 42](#_Toc417044508)

[3.2.6 Codemirror per a non-rich editor 43](#_Toc417044509)

[3.2.6.1 Instal·lació de Codemirror per a non-rich editor 43](#_Toc417044510)

[3.2.6.2 Exemple bàsic de Codemirror per a non-rich editor 44](#_Toc417044511)

[3.2.6.3 Ús de Codemirror per a non-rich editor en l’aplicació Editor 46](#_Toc417044512)

[3.2.7 Solució de problemes 47](#_Toc417044513)

[3.2.7.1 Generació de l’arxiu WAR 47](#_Toc417044514)

[3.2.7.2 CKEditor esborra els estils i atributs dels elements HTML 47](#_Toc417044515)

[3.2.7.3 CKEditor/Text Area afegeix espais i tabulacions 48](#_Toc417044516)

[3.2.7.4 L’editor del CKEditor no apareix al navegador 49](#_Toc417044517)

[3.2.8 Més opcions per l’arxiu config.js 49](#_Toc417044518)

[3.2.9 Creació d’un botó de la barra d’eines 49](#_Toc417044519)

[4 Aplicació Visor (rrhh) 50](#_Toc417044520)

[4.1 Mapejat del domini 50](#_Toc417044521)

[4.2 Controlador de visualització 52](#_Toc417044522)

[4.2.1 Redirecció 53](#_Toc417044523)

[4.2.2 Mètode render 54](#_Toc417044524)

[4.3 Controladors de cerca 54](#_Toc417044525)

[4.4 TAGs del CMS 55](#_Toc417044526)

[5 Annexos 57](#_Toc417044527)

[5.1 Activar pestanya *Markers* 57](#_Toc417044528)

[5.2 Activar pestanya *Navigator* 57](#_Toc417044529)

[5.3 Crear un Taglib personalitzat 57](#_Toc417044530)

[5.4 Configuració ràpida del plugin CKEditor per a l’aplicació Editor 57](#_Toc417044531)

[5.4.1 Instal·lació i configuració 57](#_Toc417044532)

[5.4.1.1 Arxiu BuildConfig.groovy 57](#_Toc417044533)

[5.4.1.2 Arxiu Config.groovy 58](#_Toc417044534)

[5.4.1.3 Arxiu config.js (ckeditorconfig.js) 58](#_Toc417044535)

[5.4.1.4 Pàgines GSP 60](#_Toc417044536)

[5.4.1.5 Controladors 61](#_Toc417044537)

[5.4.2 Modificacions 62](#_Toc417044538)

[5.5 Arxiu *config.js* (CKEditor) de l’aplicació Editor (gcrrhh) 62](#_Toc417044539)

[5.6 CKEditor: configuració del plugin al *Config.groovy* del gcrrhh 64](#_Toc417044540)

[5.7 Taglib's per renderitzar HTML des del controlador en una vista GSP 65](#_Toc417044541)

[6 Referències 67](#_Toc417044542)

[6.1 Bootstrap 67](#_Toc417044543)

[6.2 Spring Security Core 67](#_Toc417044544)

**Imatges**

[Imatge 1: Esquelet aplicació Grails amb Bootstrap 11](#_Toc417044379)

[Imatge 2: Imatge de Bootstrap d'un exemple bàsic 12](#_Toc417044380)

[Imatge 3: Detecció de problemes del plugin Srping Security Core amb la pestanya Markers 13](#_Toc417044381)

[Imatge 4: Esquelet d’una aplicació Grails amb TreeTable 21](#_Toc417044382)

[Imatge 5: Exemple sencill amb TreeTable. Vista 1 23](#_Toc417044383)

[Imatge 6: Exemple sencill amb TreeTable. Vista 2 23](#_Toc417044384)

[Imatge 7: Aparença del CKEditor amb configuració ràpida 27](#_Toc417044385)

[Imatge 8: Fulles d'estils a fer servir en l'àrea d'edició del CKEditor 32](#_Toc417044386)

[Imatge 9: indentació codi HTML del CKEditor a l'enviar dades al controlador 33](#_Toc417044387)

[Imatge 10: Espais/tabulacions afegides en l'editor de codi HTML/GSP 47](#_Toc417044388)

[Imatge 11: Taules de Continguts i Templates 49](#_Toc417044389)

**Figures**

[Figura 1: Referenciar Bootstrap en una aplicació Grails 12](#_Toc417044390)

[Figura 2: Codi d'exemple d'ús de Bootstrap 13](#_Toc417044391)

[Figura 3: Instal·lació del plugin Spring Security Core 14](#_Toc417044392)

[Figura 4: Instal·lació del plugin Spring Security CAS 14](#_Toc417044393)

[Figura 5: Ruta de ReflectionsUtils.groovy d'Spring Security Core a la pestanya Navigator 15](#_Toc417044394)

[Figura 6: Ruta de ReflectionsUtils.groovy d'Spring Security Core al sistema de fitxers 15](#_Toc417044395)

[Figura 7: Import erroni a ReflectionsUtils.groovy 15](#_Toc417044396)

[Figura 8: Import correcte a ReflectionsUtils.groovy 15](#_Toc417044397)

[Figura 9: Instrucció incorrecta a ReflectionsUtils.groovy 15](#_Toc417044398)

[Figura 10: Instrucció correcta a ReflectionsUtils.groovy 15](#_Toc417044399)

[Figura 11: Configuració de seguretat en comú dels diferents entorns 15](#_Toc417044400)

[Figura 12: Configuració de seguretat específica per entorns 16](#_Toc417044401)

[Figura 13: Nom del servidor d'exemple 16](#_Toc417044402)

[Figura 14: Possible error al configurar Spring Security 17](#_Toc417044403)

[Figura 15: *Config.grrovy* eliminar plugins de seguretat 18](#_Toc417044404)

[Figura 16: BuildConfig.grrovy eliminar plugins de seguretat 19](#_Toc417044405)

[Figura 17: aplication.properties eliminar plugins de seguretat 19](#_Toc417044406)

[Figura 18: Modificació de l'arxiu BaseControler.groovy per desactivar el CAS 20](#_Toc417044407)

[Figura 19: Re-instal·lació del plugin Spring Security CAS i Spring Security Core 20](#_Toc417044408)

[Figura 20: Canviar el nom del servidor 21](#_Toc417044409)

[Figura 21: Canviar el nom del servidor a Grails 2.4.3 21](#_Toc417044410)

[Figura 22: Referenciar TreeTable en una aplicació Grails 23](#_Toc417044411)

[Figura 23: Codi exemple senzill amb TreeTable 24](#_Toc417044412)

[Figura 24: Crida al template que construeix el TreeTable 25](#_Toc417044413)

[Figura 25: Codi Javascript per executar TreeTable 26](#_Toc417044414)

[Figura 26: Instal·lació del plugin CKEditor 27](#_Toc417044415)

[Figura 27: Codi GSP per incloure CKEditor 28](#_Toc417044416)

[Figura 28: Tractar el text escrit al CKEditor 28](#_Toc417044417)

[Figura 29: Ruta arxiu de configuració config.js del CKEditor a la pestanya Navigator 28](#_Toc417044418)

[Figura 30: Ruta arxiu de configuració *config.js* del CKEditor al sistema de fitxers 29](#_Toc417044419)

[Figura 31: Ruta arxiu *config.js* al sistema de fitxers amb versions antigues de Grails 29](#_Toc417044420)

[Figura 32: Arxiu *config.js* original 29](#_Toc417044421)

[Figura 33: Arxiu *config.js* modificat 29](#_Toc417044422)

[Figura 34: Aparença del CKEditor amb l’arxiu *config.js* modificat 30](#_Toc417044423)

[Figura 35: Reubicació de l'arxiu *config.js* 30](#_Toc417044424)

[Figura 36: Canviar idioma CKEditor a l'arxiu *config.js* 31](#_Toc417044425)

[Figura 37: Personalitzar la barra d’eines de CKEditor a l'arxiu *config.js* 32](#_Toc417044426)

[Figura 38: Canviar idioma CKEditor a l'arxiu *config.js* 32](#_Toc417044427)

[Figura 39: Canviar idioma CKEditor a l'arxiu *config.js* 32](#_Toc417044428)

[Figura 40: Canviar idioma CKEditor a l'arxiu *config.js* 33](#_Toc417044429)

[Figura 41: Formatat de les dades per defecte de la versió 4.4.1.0 del CKEditor 34](#_Toc417044430)

[Figura 42: Formatat de les dades per defecte de la versió 3.6.6.1.0 del CKEditor 35](#_Toc417044431)

[Figura 43: Canviar el format de tractament del text HTML del CKEditor a l'arxiu *config.js* 36](#_Toc417044432)

[Figura 44: Configuració del CKEditor a l'arxiu Config.groovy 37](#_Toc417044433)

[Figura 45: Config.groovy bàsic per pujar arxius al servidor amb CKEditor 38](#_Toc417044434)

[Figura 46: Carpeta del servidor 39](#_Toc417044435)

[Figura 47: Arxiu Config.groovy per pujar arxius al servidor aplicació gcrrhh 40](#_Toc417044436)

[Figura 48: Codi GSP per incloure l'espai en la ruta del servidor 40](#_Toc417044437)

[Figura 49: Ubicació PathUtils.groovy del CKEditor amb la pestanya Navigator 41](#_Toc417044438)

[Figura 50: Ubicació PathUtils.groovy del CKEditor en el sistema de fitxers 41](#_Toc417044439)

[Figura 51: Modificació al fitxer PathUtils.groovy per eliminar la capeta *File/Image* 41](#_Toc417044440)

[Figura 52: Arxiu config.js per fer servir Codemirror al CKEditor 43](#_Toc417044441)

[Figura 53: Afegir llibreries Codemirror non-rich editor 44](#_Toc417044442)

[Figura 54: Codi Javascript per a Codemirror non-rich editor 44](#_Toc417044443)

[Figura 55: Afegir un *theme* a Codemirror non-rich editor 44](#_Toc417044444)

[Figura 56: Exemple bàsic de Codemirror per a non-rich editor 45](#_Toc417044445)

[Figura 57: Exemple bàsic de Codemirror per a non-rich editor en un Text Area 46](#_Toc417044446)

[Figura 58: Codemirror en non-rich editor per l'aplicació gcrrhh 47](#_Toc417044447)

[Figura 59: Arxiu *BuildConfig.groovy* per generar el WAR amb un projecte amb CKEditor 47](#_Toc417044448)

[Figura 60: Exemple codi HTML/GSP amb estils i atributs que es perden per culpa de CKEditor 47](#_Toc417044449)

[Figura 61: Arxiu *config.js* per evitar que es perden els estils i atributs per culpa de CKEditor 48](#_Toc417044450)

[Figura 62: Codi GSP que provoca espais/tabulacions afegides en l'editor de codi HTML/GSP 48](#_Toc417044451)

[Figura 63: Taula de Continguts mapejada a l'aplicació Visor 51](#_Toc417044452)

[Figura 64: Taula de Templates mapejada a l'aplicació Visor 52](#_Toc417044453)

[Figura 65: Consulta del contingut d'un Template 52](#_Toc417044454)

[Figura 66: Redirecció URL 53](#_Toc417044455)

[Figura 67: Redirecció al VisulController.groovy en el cas{nom\_servidor}/rrhh 53](#_Toc417044456)

[Figura 68: Determinació del perfil de l'usuari 54](#_Toc417044457)

[Figura 69: Renderització de la pàginaweb amb VisualController.groovy 54](#_Toc417044458)

[Figura 70: Consulta SQL SearcherController.groovy 55](#_Toc417044459)

[Figura 71: Consulta SQL ActualitzionsController.groovy 55](#_Toc417044460)

[Figura 72: Instal·lació del plugin CKEditor 57](#_Toc417044461)

[Figura 73: Arxiu Config.groovy 58](#_Toc417044462)

[Figura 74: Arxiu ckeditorconfig.js 60](#_Toc417044463)

[Figura 75: Afegir CKeditor de manera ràpida a una vista GSP 61](#_Toc417044464)

[Figura 76: Afegir libreria <ckeditor:resources> depenent de si es fa servir CKEditor o editor de codi HTML/GSP 61](#_Toc417044465)

[Figura 77: Codi GSP que provoca espais/tabulacions afegides en l'editor de codi HTML/GSP 61](#_Toc417044466)

[Figura 78: Arxiu config.js (CKEditor) de l’aplicació Editor (gcrrhh) 64](#_Toc417044467)

[Figura 79: CKEditor: configuració del plugin al Config.groovy del gcrrhh 65](#_Toc417044468)

[Figura 80: Taglib personalitzat per a les aplicacions RPPAS, RPPDI i RPBEC 66](#_Toc417044469)

**Taules**

[Taula 1: Tags del CMS 56](#_Toc417044470)

[Taula 2: Variables especials del CMS 56](#_Toc417044471)

# 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 i editor amb minúscules es referirà a la zona d’edició del plugin CKEditor): 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 que tenen per defecte. A més, el *frameworks* 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 suposats 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, Spring Security 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 de la pàgina web al dispositiu mòvil redimensionant-los.

### 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 : Referenciar Bootstrap en una aplicació Grails

A la següent imatge es pot veure com quedaria el projecte amb les carpetes corresponents a l’entorn de desenvolupament GGTS:



Imatge : 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 : Codi d'exemple d'ús de Bootstrap

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



Imatge : Imatge de Bootstrap d'un exemple bàsic

## 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 la base 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 : 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 : Instal·lació del plugin Spring Security CAS

I tornar a refrescar dependències. Independentment de si s’obtenen errors o no 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 consultar **5.1 Activar pestanya *Markers***) on s’ha produït l’error, com es mostra a la següent imatge:



Imatge : 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. De manera molt senzilla, fent doble click sobre *Groovy unable to resolve...*, s’obrirà a l’editor l’arxiu problemàtic.

Una altra opció 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 : 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 *Navigator***).

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

Llavors, dintre de l’arxiu ***ReflectionsUtils.groovy*** substituir:

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

Figura : Import erroni a ReflectionsUtils.groovy

per:

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

Figura : Import correcte a ReflectionsUtils.groovy

Substituir:

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

Figura : Instrucció incorrecta a ReflectionsUtils.groovy

per:

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

Figura : Instrucció correcta a ReflectionsUtils.groovy

### 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 : Configuració de seguretat en comú 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 : 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 : 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 el nom de l’aplicació (rrhh o gcrrhh). Aixó és útil per quan una aplicació no te un nom definitiu.

S’ha de tenir en compte l’entorn que es 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  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'}" |

Figura : Possible error al configurar Spring Security

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

### 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 que el package **Security** quedi en la mateixa ubicació del 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 arxius: **BaseControler*.groovy*, Crypt*.groovy*, Database*.groovy*, FileUploader*.groovy*, Validation*.groovy***.

### Com fer servir els plugins de seguretat

Fins ara s’ha explicat quins són els plugins 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().

## 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 eliminar-los per segons quin tipus de proves, 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.

### 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 : *Config.grrovy* eliminar plugins de seguretat

### 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 : 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 desinstalar-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 : aplication.properties eliminar plugins de seguretat

### 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 mètodes:
  + *def populateUserName()* queda de la següent manera:

|  |
| --- |
| def populateUserName()  {  session["user"] = "jesus.campos"  } |

Figura : Modificació de l'arxiu BaseControler.groovy per desactivar el CAS

* + Afegir a l’inici del *mètode doSecurityCheck()*:

session.user="jesus.campos"

### Desfer els canvis (re-instal·lar CAS)

Per desfer els passos il·lustrats en el anteriors apartats de manera ràpida.

Descomentar de la closure **plugins** de l’arxiu ***BuildConfig.groovy*** l’Spring Security CAS i l’Spring Security Core:

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

Figura : Re-instal·lació del plugin Spring Security CAS i Spring Security Core

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

Anar a la carpeta del projecte als sistema de fitxers i Fer *Tortoise SVN → Revert...* (només dels arxius modificats en el moment de treure l’Spring Security CAS i l’Spring Security Core) i després *Tortoise SVN → update* del projecte per tornar a descarregar els dominis i controladors del package **Security** en cas d’haver-los eliminat. Fer botó dret → *Refresh* del projecte al GGTS.

Ara només restarà per modificar l’ error degut a l’arxiu ***ReflectionUtils.groovy***. En l’apartat **2.2.2 Modificacions al plugines mostra la solució**.

## 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 : Canviar el nom del servidor

En el cas de Grails 2.4.3 aixó no és suficient, a més a l’arxiu **BuildConfig.groovy** s’han 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],  ...  ] |

Figura : Canviar el nom del servidor a Grails 2.4.3

## Arxius CSS i Javascript

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

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



# 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 aplicable 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 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 elements amb moviment (desplegables).

## TreeTable

Plugin per visualitzar el contingut d’un espai de l’Editor de manera ordenada.

### Instal·lació 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’entorn de desenvolupament GGTS:



Imatge : Esquelet d’una 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 els continguts que es poden editar, peró no quan s’estigui editant una pàgina. Per referenciar el plugin TreeTable en 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=*"/*${grails.util.Metadata.current.'app.name'}*/css/treeTable/jquery.treetable.css"* rel=*"stylesheet"* type=*"text/css"* />  <link href=*"/*${grails.util.Metadata.current.'app.name'}*/css/treeTable/jquery.treetable.theme.default.css"* rel=*"stylesheet"* type=*"text/css"* />  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/js/treeTable/jquery.treetable.js"*>  </script>  ... |

Figura : Referenciar TreeTable en una aplicació Grails

### Ús del plugin TreeTable

Per testejar 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.

#### 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 : Codi exemple senzill amb TreeTable

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



Imatge : Exemple sencill amb TreeTable. Vista 1

Expandint el node *Folder 1* que té fills:



Imatge : 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 els altres serien els fills. Per exemple, si es vol que un element estigui dintre d’altre, l’element fill ha de tenir com a atribut data-tt-parent-id 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 exemple la fila es divideix en tres columnes, aixó pot variar. La imatge del triangle indicant que l’element té fills apareix automàticament en el cas de que el valor de l’atribut data-tt-id de l’element aparegui en l’atribut data-tt-parent-id d’un altre element. Les imatges de carpetes i fitxers es poden modificar amb l’atribut class del tag <span> on estan ubicades, indicant *folder* o bé *file* com a valor.

Per finalitzar, per tal d’indicar en quin element es vol aplicar el plugin TreeTable s’ha de posar el codi Javascript del final. S’ha d’executar després de que es carregui la pàgina. És similar a l’ús d’altres plugins, com podria ser el de incloure un calendari, i permet personalitzar-lo de manera similar, fent servir aquesta funció Javascript.

Per a més informació sobre la personalització del plugin: <http://ludo.cubicphuse.nl/jquery-treetable/>

#### Exemple complex

Aquest exemple es basa en crear un template de Grails i fer-lo servir expressament per mostrar les files.

|  |
| --- |
| <table id="expandable-tree-table-plantillas">  <thead>  <tr>  <th width="90%">Plantilla</th>  <th width="10%">Accions</th>  </tr>  </thead>  <tbody>  <g:each in="${plantillasContent}" var="c">  <g:render template="newTreeTableNode" model="[c:c, tipDoc:'plantilles']"/>  </g:each>  </tbody>  </table> |

Figura : Crida al template que construeix el TreeTable

El template no és més que una vista GSP amb el caràcter “\_” com a primera lletra del seu nom. D’aquesta manera s’aconsegueix carregar el codi d’una vista GSP dintre d’una altra GSP.

Una vegada es tenen les dades del controlador, cada element tindrà com a atribut data-tt-id el seu id de la base de dades, i en el cas que tingui fills, cridarà novamente al template que s’encarregarà de construir una nova fila per cada fill, on cada fill tindrà un data-tt-parent-id corresponent a l’identificador del pare que té indicat a la base de dades. Aquest comportament és recursiu, llavors si un fill té fills no hi ha problema, no és processarà el següent fill fins que no s’hagin tractat tots els descendents de l’element actual.

En aquesta ocasió la crida Javascript es realitza al principi de la GSP fent servir el codi de la línia 2 que serveix per executar el codi Javascript quan s’ha carregat tota la pàgina.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <script type="text/javascript">  $(document).ready(**function**(){  //TreeTable JQuery Plugin  // Highlight selected row  $("#expandable-tree-table-templates tbody, #expandable-tree-table-  plantillas tbody").on("mousedown", "tr", **function**() {  $(".selected").not(**this**).removeClass("selected");  $(**this**).toggleClass("selected");  });  $("#expandable-tree-table-templates, #expandable-tree-table-  plantillas").treetable({ expandable: **true** });  });  </script> |

Figura : Codi Javascript per executar TreeTable

El codi contingut entre les línies 5 i 9 serveix per il·luminar en un color diferent la columna sobre la que s’ha efectuat un click. Permet seleccionar i des-seleccionar una fila.

## 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. És conegut també com a *rich editor*, donat que no és un editor de text pla i sense format.

Tots els exemples i explicacions estan provats amb la versió 4.4.1.0. La versió 3.6.6.1.0 (que és la que fa servir el gestor de continguts Weceem) funciona pràcticament igual excepte en dues situacions:

* En el moment de crear un botó personalitzat amb funcions própies.
* En el moment de enviar les dades al controlador: la versió 3.6.6.1.0 envia el codi HTML amb indentacions mentres que la 4.4.1.0 no ho fa, s’ha de configurar expressament.

Aquest plugin té unes certes limitacions. Una és que s’ha de configurar de manera adequda, la configuració per defecte no serveix, per tal que no esborri els estils o atributs de determinats tags HTML (**3.2.6.2 CKEditor esborra els estils i atributs dels elements HTML**). La configuració per defecte esborra o no té en compte els següents tags HTML: <html>, <head> i <body>(**3.2.7.2 CKEditor esborra els estils i atributs dels elements HTML**).

En aquests apartat s’explica com configurar el plugin CKEditor pas a pas per entendre qué fa en cada moment i així poder arribar a la configuració desitjada per l’aplicació Editor (gcrrhh). Per una configuració ràpida verure (**5.3 Configuració ràpida del plugin CKEditor per a l’aplicació Editor**).

### Instal·lació i configuració del plugin CKEditor

En l’arxiu **BuildConfig.groovy** s’ha d’afegir la següent línea:

|  |
| --- |
| grails.project.dependency.resolution = {  ...  plugins {  ...  compile ":ckeditor:4.4.1.0"  ...  }  } |

Figura : Instal·lació del plugin CKEditor

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

A més, si es vol que es pugui construir l’arxiu WAR s’han de realitzar els passos indicats a l’apartat **3.2.7.1 Generació de l’arxiu WAR**.

Amb aixó, i el següent codi a una vista GSP:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <head>  <ckeditor:resources/>  </head>  <body>  <form>  <ckeditor:editor name=*"text"* height=*"400px"* width=*"80%"*>${initialValue}</ckeditor:editor>  <g:actionSubmit action=*"save"* value=*"Desar"* />  </form>  </body> |

Figura : Codi GSP per incloure CKEditor

Ja es visualitzarà l’editor al navegador. On ${initialValue} és el text que apareixerà d’inici a l’editor. Per tractar el text que s’ha escrit a l’editor és necessari recuperar la variable params.text. Llavors el controlador que crida a la vista GSP on es troba l’editor ha de tenir la següent acció:

|  |
| --- |
| **def** index() {  [initialValue: "Text inicial a mostrar"]  }    **def** save() {  *println* "Text: \n"+params.text  redirect(action: "index")  } |

Figura : Tractar el text escrit al CKEditor

Es pot veure l’aparença de l’editor, que hauria de semblar-se a la de la imatge de sota:



Imatge : Aparença del CKEditor amb configuració ràpida

### Arxiu de configuració config.js

Si es volen editar determinades propietats o modificar alguns comportaments de l’editor es pot fer mitjançant l’arxiu de configuració del plugin anomenat ***config.js***. Aquest arxiu es troba al plugin, llavors la forma d’accedir a ell és a través de la ruta:

|  |
| --- |
| /{nom\_aplicacio}/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/web-app/js/ckeditor/config.js. |

Figura : Ruta arxiu de configuració config.js del CKEditor a la pestanya Navigator

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

Si es vol accedir des d’el sistema de fitxers:

|  |
| --- |
| %dir\_workspace%\%dir\_aplicacio%\target\work\plugins\ckeditor-4.4.1.0\web-app\js\ckeditor\config.js |

Figura : Ruta arxiu de configuració *config.js* del CKEditor al sistema de fitxers

En versions anteriors a Grails 2.4.3 aquesta carpeta en el sistema de fitxers en lloc de trobar-se al propi *workspace* es trobaria ubicada en la carpeta del usuari en una ruta similar a:

|  |
| --- |
| C:\Users\Jesus Campos\.grails\2.3.7\projects\nom\_aplicacio\plugins\ckeditor-4.4.1.0\web-app\js\ckeditor\config.js |

Figura : Ruta arxiu *config.js* al sistema de fitxers amb versions antigues de Grails

Si obrim aquest arxiu veurem que té el següent codi:

|  |
| --- |
| /\*\*  \* **@license** Copyright (c) 2003-2013, CKSource - Frederico Knabben. All rights reserved.  \* For licensing, see LICENSE.html or http://ckeditor.com/license  \*/  CKEDITOR.editorConfig = **function**( config ) {  // Define changes to default configuration here. For example:  // config.language = 'fr';  // config.uiColor = '#AADC6E';  }; |

Figura : Arxiu *config.js* original

Per fer-nos una idea del que es pot arribar a personalitzar l’editor podem posar el següent codi:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | CKEDITOR.editorConfig = **function**( config ) {  // Define changes to default configuration here. For example:  config.language = 'ca';  config.uiColor = '#69FFCD';    config.toolbar = [  { name: 'document', items: [ 'Image','Source', '-', 'NewPage', 'Preview'  ]}, { name: 'clipboard', items: [ 'Cut', 'Copy', 'Paste', 'PasteText',  'PasteFromWord', '-', 'Undo', 'Redo' ] },  '/',  { name: 'basicstyles', items: [ 'Bold', 'Italic' ] }  ];  }; |

Figura : Arxiu *config.js* modificat

Amb l’anterior exemple es pot veure com l’editor ja no té els matiexos botons per a l’edició de text, degut a a que entre les línies 6 i 12 s’ha personalitzat la barra d’eines amb config.toolbar = [...];. El color del marc on tenen cabuda els botons s’ha modificat a la línia 4, i es poden fer servir valors RGB, hexadecimal o el nom del color com per exemple: *red*, *blue*, etc. A la línia 3 s’ha configurat el llenguatge dels *tooltips* d’ajuda per a cada botó.

En la imatge següent es pot veure com quedaria l’editor:



Figura : Aparença del CKEditor amb l’arxiu *config.js* modificat

Ara bé, per norma general els plugins no es pugen als repositoris i com s’ha vist, en versions antigues l’arxiu ***config.js*** es troba fora del *workspace*. Tot aixó fa que aquest arxiu sigui fàcil de perdre, la qual cosa ens indica que potser hi ha un altre lloc on posar tota aquesta configuració de manera fàcil i que estigui segura de possibles pèrdues. A més si la nova ubicació pot ser igual d’accessible que la dels arxius de configuració, controladors o vistes, molt millor. Tot aixó existeix, i es pot aconseguir afegint a l’arxiu ***Config.groovy*** de l’aplicació principal el següent codi:

|  |
| --- |
| ckeditor {  ...  config = "/js/myckconfig.js"  ...  } |

Figura : Reubicació de l'arxiu *config.js*

Amb aquesta configuració, l’arxiu ***config.js***s’ubicarà a la carpeta */web-app/js* del projecte principal i l’arxiu s’anomenarà ***myckconfig.js*** (pot ser un altre nom). Llavors tots els canvis que vulguem fer s’escriuran aquí i no es tindran en compte els que hi hagin dintre del ***config.js*** que es torba a la ruta del plugin.

A partir d’ara, es parlarà d’arxiu de configuració o arxiu ***config.js*** per referir-se a ambdos arxius, ***config.js*** del plugin i ***myckconfig.js*** (o el seu equivalent triat pel lector). Quan s’indiqui que una línia de codi va situada en aquest arxiu, cadascú sabrà on posar-la.

El codi i detallades descripcions de l’arxiu de configuració que es fa servir a l’aplicació Editor (gcrrhh) es poden trobar a l’apartat  **5.4 Arxiu *config.js* (CKEditor) de l’aplicació Editor (gcrrhh)**.

#### Personalitzar l'idioma

Per canviar l’idioma del CKEditor afegir a l’arxiu ***config.js*** (o el seu equivalent) el següent codi:

|  |
| --- |
| CKEDITOR.editorConfig = **function**( config ) {  ...  config.language = 'ca';  ...  }; |

Figura : Canviar idioma CKEditor a l'arxiu *config.js*

#### Personalitzar la barra d'eines

Per treure i posar nous botons a la barra d’eines del CKEditor afegir a l’arxiu ***config.js*** (o el seu equivalent) el següent codi:

|  |
| --- |
| CKEDITOR.editorConfig = **function**( config ) {  ...  config.toolbar = 'Editor';  config.toolbar\_Editor = [  { name: 'document', items : [ 'Source','-','Save','NewPage','DocProps','Print'] },  { name: 'clipboard', items : [ 'Cut','Copy','Paste','PasteText','PasteFromWord','-','Undo','Redo' ] },  { name: 'editing', items : [ 'Find','Replace','-','SelectAll','-','SpellChecker', 'Scayt' ] },  { name: 'forms', items : [ 'Form', 'Checkbox', 'Radio', 'TextField', 'Textarea', 'Select', 'Button', 'ImageButton', 'HiddenField' ] },  '/',  { name: 'basicstyles', items : [ 'Bold','Italic','Underline','Strike','Subscript','Superscript','-','RemoveFormat' ] },  { name: 'paragraph', items : [ 'NumberedList','BulletedList','-','Outdent','Indent','-','Blockquote','CreateDiv',  '-','JustifyLeft','JustifyCenter','JustifyRight','JustifyBlock','-','BidiLtr','BidiRtl' ] },  { name: 'links', items : [ 'Link','Unlink','Anchor' ] },  { name: 'insert', items : [ 'Image','Flash','Table','HorizontalRule','Smiley','SpecialChar','PageBreak','Iframe' ] },  '/',  { name: 'styles', items : [ 'Styles','Format','Font','FontSize' ] },  { name: 'colors', items : [ 'TextColor','BGColor' ] },  { name: 'tools', items : [ 'Maximize', 'ShowBlocks'] },  { name: 'tools', items : [ 'About'] },  ];  ...  }; |

Figura : Personalitzar la barra d’eines de CKEditor a l'arxiu *config.js*

Tots els items que s’indiquen han d’existir, en el cas que un botó no es pugui trobar automàticament es carrega la barra d’eines *Full*.

#### Botó de indentació



Per canviar el valor per defecte al pulsar el botó de indentació ( ) del CKEditor afegir a l’arxiu ***config.js*** (o el seu equivalent) el següent codi:

|  |
| --- |
| CKEDITOR.editorConfig = **function**( config ) {  ...  config.indentOffset = 5;  ...  }; |

Figura : Canviar idioma CKEditor a l'arxiu *config.js*

Referència: <http://docs.cksource.com/ckeditor_api/symbols/CKEDITOR.config.html#.indentOffset>

#### Fulles d'estils aplicades a l’àrea d’edició

A l’hora d’escriure text en l’àrea d’edició es poden seleccionar diferents estils a aplicar al text. Aquest estils venen predefinits, però si es volen canviar s’ha d’afegir a l’arxiu ***config.js*** (o el seu equivalent) el següent codi:

|  |
| --- |
| CKEDITOR.editorConfig = **function**( config ) {  ...  config.contentsCss = "/gcrrhh/css/contents.css"  ...  }; |

Figura : Canviar idioma CKEditor a l'arxiu *config.js*

Llavors ara els estils que es carregaran seran els que es troben en aquesta fulla d’estils CSS.

I si vull carregar els estils que es troben en diverses fulles? La solució és que sigui l’arxiu contents.css qui carregui totes les fulles d’estils que es desitjin fer servir. Llavors, suposant que volem fer servir totes les fulles d’estils marcades en vermell:



Imatge : Fulles d'estils a fer servir en l'àrea d'edició del CKEditor

La fulla d’estils contents.css tindria un aspecte com el següent:

|  |
| --- |
| @import *url("bootstrap.css")*;  @import *url("bootstrapUPC.css")*;  @import *url("contacte.css")*;  @import *url("desplegables-fitxa.css")*;  @import *url("errors.css")*;  @import *url("estilsUPC.css")*;  @import *url("main.css")*;  @import *url("menus.css")*;  @import *url("mobile.css")*;  @import *url("sticky-footer.css")*;  @import *url("mobile.css")*;  @import *url("tmp.css")*; |

Figura : Canviar idioma CKEditor a l'arxiu *config.js*

#### Indentació del codi HTML a l'enviar les dades al controlador

Donat el següent exemple de disseny de pàgina web:



Imatge : indentació codi HTML del CKEditor a l'enviar dades al controlador

Quan es fa click a enviar les dades, el controlador rep el codi amb el següent format:

|  |
| --- |
| <div class="quadreContacte">  <div class="titol-contacte">Accessibilitat</div>  </div>  <!-- -->  <div class="quadreContacte">  <div class="txt-contacte">&nbsp;</div>  <!-- -->  <div class="titol-contacte">Pautes WAI</div>  <div class="txt-contacte">Requisits:<br />  <a href="" target="\_blank"><img /> </a> <img /><br />  Les implementacions m&eacute;s destacables s&oacute;n les seg&uuml;ents:  <ul class="non-margin-left">  <li>Item 1</li>  <li>Item 2</li>  <li>Item 3</li>  <li>Item 4</li>  <li>Item 5</li>  </ul>  </div>  <!-- --></div> |

Figura : Formatat de les dades per defecte de la versió 4.4.1.0 del CKEditor

En principi aixó no representa cap problema. El contratemps d’aquest situació és que si volem fer un [editor de codi HTML/GSP a partir d’un Text Area](#_Codemirror_per_a) no tindrem indentació entre tags de tipus <div> que es troben dintre d’altres tags tal i com sí tenim entre els <ul> i <li> de les llistes HTML.

Aquest codi en la versió del CKEditor 3.6.6.1.0 (versió que fa servir el CMS Weceem) estaria formatat de la següent manera:

|  |
| --- |
| <div class="quadreContacte">   <div class="titol-contacte">     Accessibilitat</div> </div> <!-- --> <div class="quadreContacte">   <div class="txt-contacte">     &nbsp;</div>   <!-- -->   <div class="titol-contacte">     Pautes WAI</div>   <div class="txt-contacte">     Requisits:<br />     <a href="" target="\_blank"><img /> </a> <img /><br />     Les implementacions m&eacute;s destacables s&oacute;n les seg&uuml;ents:     <ul class="non-margin-left">       <li>         Item 1</li>       <li>         Item 2</li>       <li>         Item 3</li>       <li>         Item 4</li>       <li>         Item 5</li>     </ul>   </div>   <!-- --></div> |

Figura : Formatat de les dades per defecte de la versió 3.6.6.1.0 del CKEditor

Per tant, estaria bé aconseguir que la versió 4.4.1.0 es comportés com la 3.6.6.1.0 en aquest aspecte, ja que al visualitzar aquest codi en un Text Area d’HTML les indentacions són les mateixes. Aixó s’aconsegueix afegint el següent codi a l’arxiu ***config.js*** (o el seu equivalent):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | CKEDITOR.editorConfig = **function**( config ) {  ...config.language = 'ca';  };  //Permet la indentació quan s'envia el text al controlador, així en el TextArea GSP no es perd la indentació.  CKEDITOR.on( 'instanceReady', **function**( ev ) {    //Indentació tags <div>  **var** blockTags = ['div'];    **var** rules = {  indent : **true**,  breakBeforeOpen : **true**,  breakAfterOpen : **false**,  breakBeforeClose : **true**,  breakAfterClose : **true**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }    //Indentació tags <p> i <span>  blockTags = ['p', 'span'];strong    rules = {  indent : **true**,  breakBeforeOpen : **true**,  breakAfterOpen : **false**,  breakBeforeClose : **false**,  breakAfterClose : **true**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }    //Indentació tipus font bold  blockTags = ['strong'];    rules = {  indent : **false**,  breakBeforeOpen : **false**,  breakAfterOpen : **false**,  breakBeforeClose : **false**,  breakAfterClose : **false**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }  }); |

Figura : Canviar el format de tractament del text HTML del CKEditor a l'arxiu *config.js*

#### Evitar que els estils i atributs desapareguin

Veure l’apartat **3.2.6.2 CKEditor esborra els estils i atributs dels elements HTML**.

### Arxiu Config.groovy

A més de l’arxiu ***config.js***, a l’arxiu ***Config.groovy*** de l’aplicació principal també es poden configurar determinats elements, com s’ha vist amb l’arxiu ***config.js*** a l’apartat anterior.

Aquest tipus de configuracions estan relacionades amb el tipus de fitxers que es poden pujar, la seva ubicació, etc...

Un exemple per defecte seria el següent:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | ckeditor {  config = "/js/ckeditor/ckeditorconfig.js"  //skipAllowedItemsCheck = false  defaultFileBrowser = "ofm"  upload {  basedir = ""  baseurl = ""  overwrite = **false**  link {  browser = **true**  upload = **false**  allowed = ['pdf', 'doc', 'docx']  denied = []  }  image {  browser = **true**  upload = **true**  allowed = ['jpg', 'gif', 'jpeg', 'png']  denied = []  }  flash {  browser = **false**  upload = **false**  allowed = ['swf']  denied = []  }  }  } |

Figura : Configuració del CKEditor a l'arxiu Config.groovy

A la línia 2 es modifica el nom i la ubicació de l’arxiu ***config.js***. La línia 4 serveix per definir quin serà el navegador que es farà servir per pujar arxius desde l’editor.

La closure que comença a la línia 5 defineix les opcions de la pujada d’arxius. Les línies 6 i 7 indiquen on es guardaran els arxius pujats al servidor. Les closures que comencen a les línies 9, 15 i 21 serveixen perquè quan s’obri el menú lligat a aquests botons, entre le seves pestanyes apareguin les de *Puja* i *Avançat*, és a dir, quins botons tindran activada l’opció de pujada d’arxius. A les línies 12, 18 i 24 s’indiquen els tipus d’arxius, indicant només l’extensió, que es permeten pujar al servidor.

El codi i detallades descripcions que l’arxiu Config.groovy incorpora en forma de closures per tal de configurar el CKEditor fet servir a l’aplicació Editor (gcrrhh) es poden trobar a l’apartat **5.5 CKEditor: configuració del plugin al *Config.groovy* del gcrrhh**.

### Pujar arxius al servidor amb CKEditor

Per poder pujar arxius amb l’editor al final de l’arxiu ***Config.groovy*** de l’aplicació s’ha d’afegir el següent codi:

|  |
| --- |
| ckeditor {  config = "/js/ckeditor/ckeditorconfig.js"  //skipAllowedItemsCheck = false  defaultFileBrowser = "ofm"  upload {  basedir = "/uploads/"  overwrite = **false**  link {  browser = **true**  upload = **false**  allowed = ['pdf', 'doc', 'docx']  denied = []  }  image {  browser = **true**  upload = **true**  allowed = ['jpg', 'gif', 'jpeg', 'png']  denied = []  }  flash {  browser = **false**  upload = **false**  allowed = ['swf']  denied = []  }  }  } |

Figura : Config.groovy bàsic per pujar arxius al servidor amb CKEditor

Els arxius pujats seran emmagatzemats a la carpeta *File* o *Image* (depenent del tipus pujat) que es trobarà a la carpeta *uploads* que creada dintre de *web-app* quan l’aplicació estigui funcionant en l’entorn GGTS (en un servidor Apache Tomcat desplegat seria */apache-tomcat-7.0.57/webapps/gcrrhh/uploads*).

Ara bé, el gestor de continguts està format per dues aplicacions: l’Editor i el Visor. Amb l’Editor es pujaran les imatges al servidor i posteriorment, quan es visualitzi una pàgina web amb el Visor, el Visor serà l’encarregat de recuperar les imatges que es puguin trobar a la pàgina.

Per tant, seria bona idea que les imatges s’emmagatzemessin en una carpeta fora de la carpeta on es despleguen les aplicacions per tal que poguessin accedir les dues aplicacions, i a més evitar perdre les imatges al desplega l’aplicació.

La idea es guardar/llegir les imatges de la ruta base: */app/rrhh\_resources/*. Amb aquesta ruta com a base, en funció de l’espai, s’aniran creant d’altres carpetes per guardar les imatges. Per exemple, les imatges de l’espai anomenat PAS es guardarien en */app/rrhh\_resources/PAS/*. Si dintre d’aquest espai es volen crear carpetes relacionades amb els apartats de l’espai es pot fer mitjançant el propi editor del CKEditor.

Llavors al servidor (sistema de fitxers) tenim la següent situació a la carpeta *webapps*:



Figura : Carpeta del servidor

Com es pot comprovar, tenim l’aplicació de l’Editor desplegada (gcrrhh.war) amb la seva corresponent carpeta (/gcrrhh), i la mateixa situació amb el Visor (aplicació rrhh). A més tenim una carpeta anomenada *rrhh\_resources*, que no és ni més ni menys que un enllaç simbòlic a una carpeta ubicada a */app/rrhh\_resources*, és a dir, fora de la carpeta *webapps* del servidor. Aixó significa que quan l’aplicació hagi de pujar un arxiu, baixarà un nivell jeràrquic i entrarà a la carpeta *rrhh\_resources*, i quan deixi un arxiu en aquesta ubicació, realment ho farà a */app/rrhh\_resources*.

Per què no fer que directament l’editor del CKEditor deixi els arxius a */app/rrhh\_resources*?

Un aspecte són els permisos (esborrar i renombrar un arxiu) i altre és que si no es fa servir aquest sistema, quan es puja una imatge es pugui visualitzar correctament amb el Visor, però en el moment de l’edició de la pàgina amb el CKEditor, es veu un enllaç trencat. Per aixó fa falta un paràmetre extra situat dintre de la closure ckeditor.config.upload que és *baseurl*, que és on el CKEditor anirà a buscar les imatges. Aquest paràmetre necessita estar referenciat a la carpeta relacionada amb l’aplicació dintre de *webapps*, per exemple si es vol fixar baseurl a */apache-tomcat-7.0.57/webapps/rrhh\_resources/* s’hauria de fer */gcrrhh/../rrhh\_resources/*.

Tenint tot aixó en compte, anem a configurar tots el punts necessaris per tal d’aconseguir el que volem fer amb el CKEditor. Serà necessari modificar:

* Arxiu ***Config.groovy*** de l’aplicació principal.
* Vista GSP on s’inclou l’editor del CKEditor.
* Canviar codi del propi plugin.

A l’arxiu ***Config.groovy*** de l’aplicació Editor (gcrrhh) afegir el següent codi en la closure mostrada:

|  |
| --- |
| ckeditor {  config = "/js/ckeditor/ckeditorconfig.js"  //skipAllowedItemsCheck = false  defaultFileBrowser = "ofm"  upload {  basedir = "/app/rrhh\_resources/"  baseurl ="/${grails.util.Metadata.*current*.'app.name'}*/../rrhh\_resources/"*  overwrite = **false**  link {  browser = **true**  upload = **false**  allowed = ['pdf', 'doc', 'docx']  denied = []  }  image {  browser = **true**  upload = **true**  allowed = ['jpg', 'gif', 'jpeg', 'png']  denied = []  }  flash {  browser = **false**  upload = **false**  allowed = ['swf']  denied = []  }  }  } |

Figura : Arxiu Config.groovy per pujar arxius al servidor aplicació gcrrhh

Amb aixó s’aconsegueix que els arxius s’emmagatzemin al servidor a la ruta */app/rrhh\_resources*. Peró d’on surt la carpeta amb el nom de l’espai?

A continuació es modifica la vista GSP on s’inclou el CKEditor de la següent manera:

|  |
| --- |
| <ckeditor:editor id="arxiuContingut" name="arxiuContingut" height="400px" width="100%" userSpace="${space}">${params?.arxiuContingut?:arxiu?.contingut}</ckeditor:editor> |

Figura : Codi GSP per incloure l'espai en la ruta del servidor

L’aspecte important d’aquesta modificació és afegir l’atribut *userSpace*. El valor d’aquest atribut s’inclou després de la ruta */app/rrhh\_resources/*. Llavors és obligació del programador passar-li des del controlador degudament el nom de l’espai en la variable *space*. S’han modifcat els atributs *id* i *name* respecte a exemples anterios i ara a la zona d’edició es carreguen les dades llegides de les estructures de bases de dades (arxiu?.contingut) o bé de la última edició de l’arxiu sense èxit (params?.arxiuContingut).

Referència: <http://ckeditor.com/forums/Support/Having-problems-integrating-Filemanager>

#### Modificació al plugin de la ruta de pujada d’arxius al servidor

Si ara mateix tractéssim de pujar un arxiu, ho faria a la carpeta */app/rrhh\_resources/{nom\_espai}/File/* o a */app/rrhh\_resources/{nom\_espai}/Image/*. Aixó és degut a que el plugin, en funció del tipus d’arxiu pujat afegeix una carpeta al final de la ruta amb un nom que coincideix amb l’extensió. Aixó ho volem evitar, i per aconseguir-ho s’ha de realitzar la modificació que permeti eliminar la carpeta *File* o *Image* que es crea automàticament. Per fer-ho s’ha de modicar l’arxiu ***PathUtils.groovy*** que es troba dintre de la carpeta del plugin.

|  |
| --- |
| /gcrrhh/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/src/groovy/org/gualdi/ grails/plugins /ckeditor/utils/PathUtils.groovy |

Figura : Ubicació PathUtils.groovy del CKEditor amb 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 *Navigator***).

Per accedir desde el sistema de fitxers:

|  |
| --- |
| %dir\_workspace%\gcrrhh\target\work\plugins\ckeditor-4.4.1.0\src\groovy\org\gualdi\grails\plugins\ckeditor\utils\PathUtils.groovy |

Figura : Ubicació PathUtils.groovy del CKEditor en el sistema de fitxers

En aquest arxiu s’ha de comentar la línia 49 aproximadament, per tal que l’últim *if* del mètode *static getBaseUrl* quedi de la següent manera:

|  |
| --- |
| def typeName = PathUtils.sanitizePath(params.type?.toLowerCase())  if (typeName) {  typeName = WordUtils.capitalize(typeName)  // baseUrl += "/" + typeName  } |

Figura : Modificació al fitxer PathUtils.groovy per eliminar la capeta *File/Image*

D’aquesta manera l’aplicació pujarà els arxius o crearà carpetes a la ruta */app/rrhh\_resources/{nom\_espai}/* del servidor. En local, si l’aplicació funciona a l’entorn GGTS, no té perquè deixar esborrar arxius o renombrar-los.

Fins aquí l’explicació de com realitzar la modificació. Per evitar modificar l’arxiu ***PathUtils.groovy*** cada vegada que es descarrega l’aplicació sencera s’ha pujat aquest arxiu al repositori. Llavors el procediment a seguir és:

* Descarregar el projecte gcrrhh.
* Obrir l’entorn de desenvolupament GGTS i importar el projecte. Llavors es descarreguen tots el plugins i aixó modificarà l’arxiu ***PathUtils.groovy*** i deixarà el que porta el plugin per defecte.
* Fer *Tortoise SVN → Revert...* del projecte per tenir la mateixa versió que al repositori.
* Fer fer botó dret → *Refresh* del projecte al GGTS.

### Codemirror per a rich editor

És una modificació interessant per la seva utilitat. Amb el botó de *Codi font* es pot veure el codi HTML/GSP peró per defecte aquest es veu sense indentació ni colors.

Per tal de poder veure amb colors, indentació i fins i tot *highlight* (és a dir, que quan hi hagi un tag seleccionat es remarqui en un color diferent el seu corresponent tag d’apertura/tancament) s’han de realitzar els següents passos:

* Descarregar el plugin de: <http://ckeditor.com/addon/codemirror> o bé fer servir directament .
* Descomprimir l’arxiu i copiar la carpeta *codemirror* en: /{nom\_aplicacio}/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/web-app/js/ckeditor/plugins desde la pestanya [*Navigator*](#_Activar_pestanya_Navigator).
* En l’arxiu de configuració de l’editor (***ckeditorconfig.js***) afegir:

|  |
| --- |
| CKEDITOR.editorConfig = **function**( config ) {  ...  config.extraPlugins = 'codemirror';    config.codemirror = {  theme: 'default',  lineNumbers: true,  lineWrapping: true,  matchBrackets: true,  autoCloseTags: true,  autoCloseBrackets: true,  enableSearchTools: true,  enableCodeFolding: true,  enableCodeFormatting: true,  autoFormatOnStart: true,  autoFormatOnUncomment: true,  highlightActiveLine: true,  highlightMatches: true,  showFormatButton: true,  showCommentButton: true,  showUncommentButton: true  };  ...  }; |

Figura : Arxiu config.js per fer servir Codemirror al CKEditor

Fins aquí l’explicació de com realitzar la modificació. Per estalviar passos quan es descarrega l’aplicació gcrrhh desde zero, s’ha pujat només la carpeta que s’ha de copiar dintre del plugin al repositori. Llavors el procediment a seguir és:

* Descarregar el projecte gcrrhh.
* Obrir l’entorn de desenvolupament GGTS i importar el projecte. Llavors es descarreguen tots el plugins i aixó esborra la carpeta *codemirror* que hi ha dintre de la ruta */gcrrhh/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/web-app/js/ckeditor/plugins*.
* Fer *Tortoise SVN → update* del projecte per tornar a baixar la carpeta *codemirror*
* Fer fer botó dret → *Refresh* del projecte al GGTS.

A més, amb el *Codemirror* es poden afegir una sèrie de botons extres per a l’edició de codi en el cas de tenir customitzada la barra d’eines (actualment no està aplicada aquesta secció):

|  |
| --- |
| { name: 'editCode', items : [ 'autoFormat', 'CommentSelectedRange', 'UncommentSelectedRange', 'AutoComplete'] } |

Figura : Botons adicionals a CKEditor deguts a Codemirror

### Codemirror per a non-rich editor

Quan el CKEditor no carrega la llibreria <ckeditor:resources/> es comporta com un Text Area d’HTML i no apareixen cap dels botons d’edició. La idea és aprofitar aixó per fer un editor de codi HTML/GSP juntament amb el plugin Codemirror per aconseguir indentació automàtica en el moment d’escriptura entre d’altres opcions.

Per tal d’aconseguir els millors resultats d’aquest plugin, com a complement, consultar l’apartat **3.2.2.5 Indentació del codi HTML a l'enviar les dades al controlador**.

#### Instal·lació de Codemirror per a non-rich editor

Per instal·lar el Codemirror s’han de realitzar els següent passos:

* Descarregar el codi de <https://codemirror.net/>.
* Descomprimir l’arxiu i canviar el nom de la carpeta *codemirror-5.0* per *codemirror*. Copiar-la en la carpeta *web-app* de l’aplicació.
* En la vista GSP afegir les llibreries:

|  |
| --- |
| <script src=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/lib/codemirror.js"* ></script>  <link rel=*"stylesheet"* href=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/lib/codemirror.css"*>  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/mode/xml/xml.js"* ></script> |

Figura : Afegir llibreries Codemirror non-rich editor

L’arxiu xml.js és que s’encarrga del format desitjat. Es poden fer servir d’altres, com per exemple javascript.js.

* Al final de la mateixa vista GSP afegir el següent codi Javascript

|  |
| --- |
| <script>  **var** editor = CodeMirror.fromTextArea(document.getElementById('idEditor'), {  theme: 'default',  lineNumbers: **true**  });  </script> |

Figura : Codi Javascript per a Codemirror non-rich editor

On idEditor ha de tenir el mateix valor de l’atribut *id* fet servir al CKEditor o Text Area.

* Si es desitjen fer servir un *theme* diferent del que hi ha per defecte, s’ha d’afegir la llibreia a utilitzar al <head>:

|  |
| --- |
| <link rel=*"stylesheet"* href=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/theme/monokai.css"*> |

Figura : Afegir un *theme* a Codemirror non-rich editor

#### Exemple bàsic de Codemirror per a non-rich editor

El codi d’una vista GSP per veure el funcionament anteriorment explicat és:

|  |
| --- |
| <head>  <!-- <ckeditor:resources/> -->  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/lib/codemirror.js"*></script>  <link rel=*"stylesheet"* href=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/lib/codemirror.css"*>  <!-- <script src="/${grails.util.Metadata.current.'app.name'}/codemirror/mode/javascript/javascript.js"></script> -->  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/mode/xml/xml.js"*></script>  <link rel=*"stylesheet"* href=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/theme/monokai.css"*>  </head>  <body>  <form>  <ckeditor:editor id=*"text"* name=*"text"* height=*"400px"* width=*"80%"*>${initialValue}</ckeditor:editor>  <g:actionSubmit action=*"save"* value=*"Desar"* />  </form>  <script>  **var** editor = CodeMirror.fromTextArea(document.getElementById('text'), {  theme: 'monokai',  lineNumbers: **true**  });  </script>  </body> |

Figura : Exemple bàsic de Codemirror per a non-rich editor

Dintre del codi Javascript es poden afegir més opcions que es poden consultar a <https://codemirror.net/doc/manual.html#config>. El paràmetre lineNumbers afegit serveix per numerar les línies de codi.

Aquest plugin es podria fer servir en un Text Area d’HTML sense necessitat de tenir instal·lat el plugin CKEditor. Substituïnt el tag <ckeditor> per un <textarea> el codi queda:

|  |
| --- |
| <head>  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/lib/codemirror.js"*></script>  <link rel=*"stylesheet"* href=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/lib/codemirror.css"*>  <!-- <script src="/${grails.util.Metadata.current.'app.name'}/codemirror/mode/javascript/javascript.js"></script> -->  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/mode/xml/xml.js"*></script>  <link rel=*"stylesheet"* href=*"/*${grails.util.Metadata.current.'app.name'}*/codemirror/theme/monokai.css"*>  </head>  <body>  <form>  <textarea id=*"text"* name=*"text"*>${initialValue}</textarea>  <g:actionSubmit action=*"save"* value=*"Desar"* />  </form>  <script>  **var** editor = CodeMirror.fromTextArea(document.getElementById('text'), {  theme: 'monokai',  lineNumbers: **true**  });  </script>  </body> |

Figura : Exemple bàsic de Codemirror per a non-rich editor en un Text Area

#### Ús de Codemirror per a non-rich editor en l’aplicació Editor

En l’aplicació Editor volem fer servir per defecte el CKEditor, però quan l’usuari marqui el checkbox *Editar codi GSP* haurà d’aparèixer un editor de codi. Aixó significa que depenent de determinats paràmetres s’ha de carragar o no la llibreria <ckeditor:resources/>.

|  |
| --- |
| <html>  <head>  ...  <link rel=*"stylesheet"* href=*"/*${grails.util.Metadata.current.'app.name'}*/css/code/codemirror.css"*>  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/js/code/codemirror.js"*></script>  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/js/code/runmode.js"*></script>  <script src=*"/*${grails.util.Metadata.current.'app.name'}*/js/code/xml.js"*></script>  ...  <g:if test=*"*${arxiu?.permetreGSP==0 || arxiu?.permetreGSP==**null**}*"*>  <ckeditor:resources/>  </g:if>  </head>  <body>  ...  <!-- Edició del contingut -->  <ckeditor:editor id=*"arxiuContingut"* name=*"arxiuContingut"* height=*"400px"* width=*"100%"* userSpace=*"*${space}*"*>${params?.arxiuContingut?:arxiu?.contingut}</ckeditor:editor>  ...  Editar codi GSP  <g:if test=*"*${arxiu?.permetreGSP==1}*"*>  <input type=*"checkbox"* id=*"arxiuPermetreGSP"* name=*"arxiuPermetreGSP"* value=*"1"* checked>  </g:if>  <g:else>  <input type=*"checkbox"* id=*"arxiuPermetreGSP"* name=*"arxiuPermetreGSP"* value=*"0"*>  </g:else>  ...  <g:if test=*"*${!(arxiu?.permetreGSP==0 || arxiu?.permetreGSP==**null**)}*"*>  <script>  CodeMirror.fromTextArea(document.getElementById('arxiuContingut'), {});  </script>  </g:if>  </body>  </html> |

Figura : Codemirror en non-rich editor per l'aplicació gcrrhh

### Solució de problemes

Els següents subapartats són configuracions per personalitzar el plugin CKEditor com ho són els anteriors, peró aquestes s’han de realitzar per corretgir errors del comportament del plugin per tal que l’aplicació pugui funcionar. Potser amb altres versions de Grails a la indicada amb que succeeix el problema o altra versió del CKEditor no es donen.

#### Generació de l’arxiu WAR

Una aplicació que estigui realitzada amb Grails 2.4.3 i incorpori el plugin CKEditor 4.4.1.0 pot donar problemes a l’hora de realitzar l’arxiu WAR. Aixó és degut a que un dels plugins que ve per defecte amb la creació d’un projecte amb la versió Grails esmentada és asset-pipeline:1.9.6. Aquest plugin s’encarrega de minimitzar el tamany de l’aplicació en el moment de realitzar el WAR, comprimint arxius CSS i JS, entre d’altres. El problema és que aquest plugin no pot comprimir el plugin CKEditor i l’error no permet construir l’arxiu WAR.

La solució és comentar a l’arxiu ***BuildConfig.groovy*** la línia on s’afegeix aquest plugin:

|  |
| --- |
| plugins {  ...  //compile ":asset-pipeline:1.9.6"  ...  } |

Figura : Arxiu *BuildConfig.groovy* per generar el WAR amb un projecte amb CKEditor

#### CKEditor esborra els estils i atributs dels elements HTML

Si al *rich editor* ens arriba el següent codi/text:

|  |
| --- |
| <p class="txt-descripcio">Informaci&oacute; relacionada amb  l&#39;acc&eacute;s a la Universitat (acollida, declaraci&oacute;  d&#39;activitats),  classificaci&oacute; dels col&middot;lectius PAS i PR, Relaci&oacute;  de Llocs de Treball (descripci&oacute; i perfils) i  desvinculaci&oacute; (ren&uacute;ncia).</p> |

Figura : Exemple codi HTML/GSP amb estils i atributs que es perden per culpa de CKEditor

Quan la vista GSP enviï aquestes dades cap al controlador, el CKEditor amb la vista en *rich editor* i no en *Codi font* haurà esborrat class="txt-descripcio" i es perdrà per sempre. Llavors, que podem fer per solucionar-ho?

Incloure a l’arxiu ***config.js*** (o el seu equivalent) el següent codi:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CKEDITOR.editorConfig = **function**( config ) {  ...  config.allowedContent=true  config.extraAllowedContent = '\*(\*)';  ...  }; |

Figura : Arxiu *config.js* per evitar que es perden els estils i atributs per culpa de CKEditor

Referència: <http://stackoverflow.com/questions/15659390/ckeditor-automatically-strips-classes-from-div>

#### CKEditor/Text Area afegeix espais i tabulacions

Si en l’editor de codi HTML/GSP realitzat amb CKEditor sense llibreries i *Codemirror* tenim una cosa similar a:



Imatge : Espais/tabulacions afegides en l'editor de codi HTML/GSP

És a dir, amb espais i tabulacions que no volem com poden ser les zones en vermell, és perqué segurament hi han espais entre els tags d’apertura i tancament <ckeditor> i el text que es mostrarà en l’editor com es pot veure a continuació marcat en groc:

|  |
| --- |
| <ckeditor:editor id="arxiuContingut" name="arxiuContingut" height="400px" width="100%" userSpace="${space}">\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_${params?.arxiuContingut?:arxiu?.contingut}  </ckeditor:editor> |

Figura : Codi GSP que provoca espais/tabulacions afegides en l'editor de codi HTML/GSP

Aixó provocarà que quan no es carregui la llibreria <ckeditor:resources/>, el Text Area que surt per defecte afegeixi espais/tabulacions de manera automàtica, ja que aquest salts de línia i espais es compten com a caràcters en un Text Area. Aquests espais s’han d’eliminar.

#### L’editor del CKEditor no apareix al navegador

Moltes vegades l’error és degut a que l’arxiu de configuració ***config.js*** té alguna instrucció incorrecta. Algunes d’elles són difícils de detectar i es posen diversos exemples:

* Fer servir un *extraPlugin* que no existei com a tal. Tenir config.extraPlugins = 'codemirror'; o altre plugin (creat per nosaltres) i no tenir la carpeta *codemirror* a la ruta */{nom\_aplicacio}/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/web-app/js/ckeditor/plugins*.
* Afegir més d’un *extraPlugin* i deixar espais entre ells (error en vermell):
  + Versió correcta: config.extraPlugins = 'divDspl,listInDspl';
  + Versió incorrecta: config.extraPlugins = 'divDspl, listInDspl';

Altres vegades falta incloure la llibreria <ckeditor:resources/>.

### Més opcions per l’arxiu config.js

Es descriuran altres configuracions interessants que es poden afegir a l’arxiu ***config.js*** o el seu equivalent.

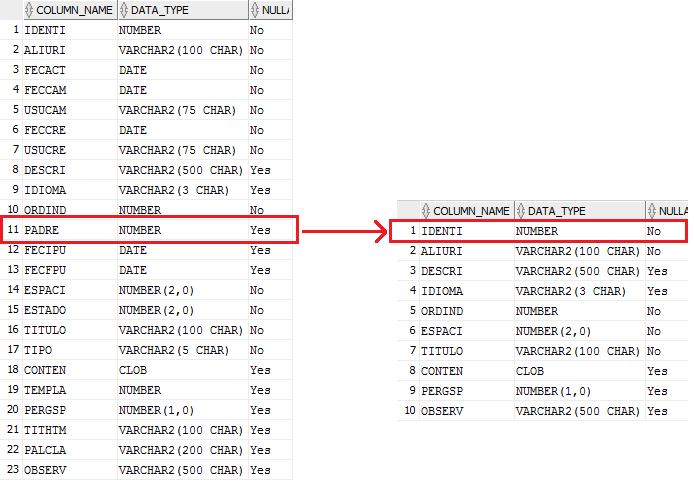
### Creació d’un botó de la barra d’eines

# Aplicació Visor (rrhh)

L’aplicació Visor o rrhh és l’encarregada de visualitzar les pàgines realitzades amb l’Editor (gcrrhh).

## Mapejat del domini

A la següent imatge es mostra la taula dels Continguts (esquerra) i la dels Templates (dreta) i la relació que hi ha entre elles:



Imatge : Taules de Continguts i Templates

El domini en el Visor está mapejat de la següent manera:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | **class** Contingut{    **static** *MAX\_PARAULES\_CLAU\_SIZE* = 200    **def** absoluteURI  **def** breadCrumb    String aliesUri  Date dataActualitzacio  Integer ordre  Integer espai  String titol  String tipusContingut  String contingut  String paraulesClau    **static** *hasMany* = [fills: Contingut]  **static** *hasOne* = [pare: Contingut, template: Template]    **static** *mapping* = {    *table* 'POR\_CCONTEN01'  *id* name:*'id'*  *id* generator:'sequence', *column*:'identi', params:[sequence:'SEC\_POR\_CCONTEN01\_IDENTI', *column*:'nextval']  *version* **false**    *aliesUri* *column*: 'aliuri'  *dataActualitzacio* *column*: 'fecact'  *ordre* *column*: 'ordind'  pare *column*: 'padre'  *espai* *column*: 'espaci'  *titol* *column*: 'titulo'  *tipusContingut* *column*: 'tipo'  *contingut* *column*: 'conten'  template *column*: 'templa'  *paraulesClau* *column*: 'palcla'  }    **static** *constraints* = {  pare(blank:**true**, nullable:**true**)  *contingut*(blank:**true**, nullable:**true**)  template(blank:**true**, nullable:**true**)  *paraulesClau*(blank:**true**, nullable:**true**, maxSize: Contingut.*MAX\_PARAULES\_CLAU\_SIZE*)  }  } |

Figura : Taula de Continguts mapejada a l'aplicació Visor

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | **class** Template{    String aliesUri  String contingut    **static** *hasMany* = [continguts: Contingut]    **static** *mapping* = {    *table* 'POR\_CTEMPLA01'  *id* name:*'id'*  *id* generator:'sequence', params:[sequence:'SEC\_POR\_CTEMPLA01\_IDENTI']  *version* **false**  *id* *column*: 'identi'    *aliesUri* *column*: 'aliuri'  *contingut* *column*: 'conten'  }    **static** *constraints* = {  *contingut*(blank:**true**, nullable:**true**)  }  } |

Figura : Taula de Templates mapejada a l'aplicació Visor

A les línies 17 i 18 de la **Figura 63: Taula de Continguts mapejada a l'aplicació Visor** i a la 6 de la **Figura 64: Taula de Templates mapejada a l'aplicació Visor** s’estableix el tipus de relació entre les dues classes. D’aquesta manera quan es consulti el camp *template* de la classe Contingut, en lloc d’obtenir-ne un identificador (Integer) s’obtindrà un objecte amb les dades del Template. Amb aixó es poden executar les següents instruccions:

|  |
| --- |
| **def** webPage = Contingut.get(0).template.contingut  **def** webPage = Contingut.get(0).template.aliesUri |

Figura : Consulta del contingut d'un Template

En aquesta ocassió estem recuperant el codi HTML (camp *contingut*) d’un Template a partir d’un Contingut que el fa servir.

L’avantatge que presenta aquesta estratègia és que donat un Contingut (la seva URL) només cal una consulta a base de dades per obtenir les seves dades i del Template que fa servir, si no, es necessitaria una consulta per obtenir l’objecte de tipus Contingut i una altra per al de tipus Template.

Per altra banda presenta un inconvenient que no afecta pel que fa a l’aplicació Visor ja que només es llegeix de la base de dades i no es guarda res en ella. Si volem crear un nou Contingut hem d’omplir totes les seves dades i en el moment de posar dades a l’atribut *template* hem d’assignar un objecte (bé creant-lo o recupreant-lo de base de dades), no podem assignar la clau primària del Template. Mapejant el camp *template* com a Integer podem assignar-li un xifra, és a dir, el valor que prengui la clau primària que es fa servir per dientificar el Template.

Aquesta explicació és aplicable als atributs *pare* i *fills* de la classe Contingut.

## Controlador de visualització

Donanada una direcció URL i una vegada l’aplicació redirecciona cap al controlador de visualització, s’identifica la pàgina que l’usuari vol visitar i busca a la base de dades el seu codi HTML i si té o no un Template associat. Amb aquestes dades construeix el codi de la pàgina definitiva que es mostrarà al navegador web fent servir els criteris de l’apartat **4.4 TAGs del CMS**.

S’ha de tenir en compte que el CKEditor esborra els tags HTML: <html>, <head> i <body>, aixó significa que si es vol que una pàgina carregui algun arxiu CSS o JS no ho podrà fer perquè l’editor ho esborrarà. Per aquest motiu la major part de les pàgines realitzades amb aquest gestor de continguts carregaran un Template.

Com a cas excepcional estan les págines que es faran servir per recarregar codi HTML enviat des del controlador a una vista GSP (**5.7 Taglib's per renderitzar HTML des del controlador en una vista GSP**).

### Redirecció

Si la direcció escrita al navegador web no coincideix amb el nom d’un controlador i una acció, aixó vol dir que es vol una pàgina web realitzada amb el gestor de continguts, i per tant s’ha d’executar el controlador ***VisualController.groovy*** que és l’encarregat de construir la pàgina web.

Per fer aquesta diferenciació s’ha d’afegir a l’arxiu ***UrlMappings.groovy*** el següent codi:

|  |
| --- |
| "/"(controller:"visual")  "/PAS/\*\*"(controller:"visual")  "/PDI/\*\*"(controller:"visual")  "/BEC/\*\*"(controller:"visual")  "/GES/\*\*"(controller:"visual") |

Figura : Redirecció URL

És a dir, sempre que una URL comenci per:

* {nomn\_servidor}/rrhh/PAS
* {nomn\_servidor}/rrhh/PDI
* {nomn\_servidor}/rrhh/BEC
* {nomn\_servidor}/rrhh/GES

No es tindrà en compte el que hi ha despés, es redireccionarà directament cap al controlador de visualització.

Cal destacar la primera línia de la **Figura 63: Redirecció URL**. L’usuari generalment accedirà al Visor amb {nomn\_servidor}/rrhh/, la qual cosa significa que aquesta serà la pàgina d’entrada al Portal en la majoria dels casos. Si l’usuari decideix entrar amb aquesta URL, el conrolador de visualització buscarà a la base de dades el seu perfil i el redireccionarà cap a l’espai corresponent amb:

|  |
| --- |
| redirect uri: "/${espaiNom}*/index"* |

Figura : Redirecció al VisulController.groovy en el cas{nom\_servidor}/rrhh

El codi que es fa servir per trobar l’espai relacionat amb el perfil de l’usuari és:

|  |
| --- |
| //Perfil de l'usuari  **def** princ = springSecurityService.getPrincipal()    **if** (princ **instanceof** String) {  espaiNom = "PAS"  } **else** {  **def** username = springSecurityService.getPrincipal()?.username  //Se consigue el grupo de la persona: PAS, PDI o BECario  **def** persona = Persona.*findByUsuari*(username.toString())    **if**(persona){  **if**(llistaPAS.contains(persona.areaPersonal.toString().toUpperCase())){  espaiNom = "PAS"  }  **else** **if** (llistaPDI.contains(persona.areaPersonal.toString().toUpperCase())){  espaiNom = "PDI"  }  **else** **if**(llistaBEC.contains(persona.areaPersonal.toString().toUpperCase())){  espaiNom = "BEC"  }  }  } |

Figura : Determinació del perfil de l'usuari

Aixó només és en el cas que no es trobi indicat l’espai a la URL. A més si l’usuari no té un perfil (PAS, PDI o BEC) es posarà per defecte l’espai PAS.

### Mètode render

Quan el controlador ha obtingut el codi HTML final de la pàgina que l’usuari vol consultar fa servir el mètode *render()* per mostra-la al navegador web:

|  |
| --- |
| render(text: webPage, contentType: "text/html", encoding: "UTF-8") |

Figura : Renderització de la pàginaweb amb VisualController.groovy

On webpage és un String amb el contingut HTML de la pàgina. Aquest controlador no es redirigeix a cap vista GSP de l’aplicació

## Controladors de cerca

Els controladors ***SearcherController.groovy*** i ***ActualitzacionsController.groovy*** fets servir per realitzar cerques reben l’espai com a paràmetre (params.espai) de la vista GSP i el van mantenint per saber en tot moment a quin espai pertany l’usuari.

La cerca a ***SearcherController.groovy*** es realitza per paraules clau. La paraula que busca l’usuari s’ha de trobar dintre d’aquest atribut.

A més es vol que a la base de dades una paraula amb accent o sense accent sigui tractada de la mateixa manera. És a dir, si l’usuari cerca “permís” ha de trobar els Continguts que a la base de dades tinguin en paraules clau “permís”, però també els que tinguin “permis”.

Per aconseguir-ho s’ha de fer servir la següent consulta SQL:

|  |
| --- |
| **def** query = {  and{  eq("espai", espaiId)  'in'("tipusContingut", *allowedTypes*)  paraulesClau.*each*{  sqlRestriction("upper(translate(palcla, 'áàâéèêíìîïóòôúùûü', 'aaaeeeiiiiooouuuu')) like upper(translate('%${it.toString()}%', 'áàâéèêíìîïóòôúùûü', 'aaaeeeiiiiooouuuu'))")  }  }  order("titol", "asc")  } |

Figura : Consulta SQL SearcherController.groovy

I tenir en compte la codificació de la página (UTF-8). Si no és aquesta no funcionarà correctament!!!

A ***ActualitzacionsController.groovy*** es cerca per data d’actualització i que tingui alguna cosa a les paraules clau:

|  |
| --- |
| **def** query = {  and{  eq("espai", espaiId)  'in'("tipusContingut", allowedTypes)  sqlRestriction("trim(palcla) is not null")  }  order("dataActualitzacio", "desc")  order("titol", "asc")  } |

Figura : Consulta SQL ActualitzionsController.groovy

## TAGs del CMS

Per tal d’incloure determinades dades als Templates i pàgines web de manera fàcil i ràpida es poden fer servir els següents tags a l’Editor:

|  |  |
| --- | --- |
| **TAG** | **Funcionalitat** |
| <!--pageContent--> | Només es fa servir als Templates. Inserta el codi HTML de la pàgina que fa servir aquest Template. |
| <!--titolContent--> | Inserta el camp *titol* de la pàgina corresponent. |
| <!--breadCrumb--> | Inserta un fil d’Ariadna. Recomanable fer-lo servir en només fitxes. |
| <!--buscadorUPC--> | Afegeix un buscador tipus ***SearchController.groovy***. |
| <!--userName--> | Nom de l’usuari. |
| <!--dataActualitzacioFitxa--> | Inserta el camp *dataActualitzacio* de la pàgina corresponent. |

Taula : Tags del CMS

Per a un reaprofitament i escalabilitat dels Templates i pàgines web és aconsellable fer servir el següent codi HTML/GSP en lloc dels seus valors:

|  |  |
| --- | --- |
| **Variable** | **Funcionalitat** |
| ${grails.util.Metadata.current.'app.name'} | Insereix el nom del servidor on es troba l’aplicació. |
| ${space?.name} | Insereix el nom de l’espai. |
| ${session.user} | Insereix el nom de l’usuari. |

Taula : Variables especials del CMS

D’aquesta manera un mateix Template/pàgina web pot servir per diversos espais i també per diferents servidors.

## Aplicacions RPPAS, RPPDI i RPBEC

A continuació es descriurà com crear un client en una aplicació Grails per a connectar-se a un servidor de web services.

### Configuració de web services

Afegir en la closure **plugins** de l’arxiu ***BuildConfig.groovy***:

|  |
| --- |
| plugins {  ...  compile ":cxf:1.1.1"  compile ":cxf-client:1.6.0"  ...  } |

Figura : Arxiu BuildConfig.groovy Instal·lació dels plugins CXF per connectar-se a web services

Després, sobre el projecte fer botó dret → *Grails Tools* → *Refresh depencies* (Alt+G,R) per actualitzar les dependències del projecte. Si es produeix algun error consultar l’apartat **4.5.6.1 Error al descarregar plugins**.

Afegir a l’arxiu ***Config.groovy*** el següent codi tenint en compte que la variable *clientInterface* no pot aparèixer, o com a mínim ha d’estar comentada:

|  |
| --- |
| cxf {  client {  tiempoClient {  wsdl = "http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx?WSDL"  //clientInterface = org.tempuri.WSMeteorologicoSoap  serviceEndpointAddress = "http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx"  }  }  } |

Figura : Exemple d'arxiu Config.groovy per configurar un client web service

Aquest exemple s’ha realitzat amb un servidor gratuït trobat a internet que ofereix informació meteorològica. En aquest cas per tractar-se d’un web service realitzat amb tecnologia .NET, si es vol trobar una descripció de les operacions compatibles, és a dir, mètodes que es poden invocar, només cal escriure al navegador web la direcció <http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx>. Per veure una descripció de tots els serveis que ofereix el servidor de web services cal anar a <http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx?WSDL>. A l’apartat **4.5.5 Exemples de web services** es poden trobar més exemples de web services el seu codi per l’arxiu ***Config.groovy***.



Anar a *Command Grails History* ( ) i escriure la insturcció *“wsdl2java”* i executar-la. Llavors a la consola s’obtindrà un missatge similar al següent:

|  |
| --- |
| Loading Grails 2.3.4  |Configuring classpath  .  |Environment set to development  ..................  |Starting wsdl2java  |Generating java stubs from http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx?WSDL  [java] Loading FrontEnd jaxws ...  [java] Loading DataBinding jaxb ...  [java] wsdl2java -verbose -d src/java http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx?WSDL  [java] wsdl2java - Apache CXF 2.6.2  .  |Completed wsdl2java |

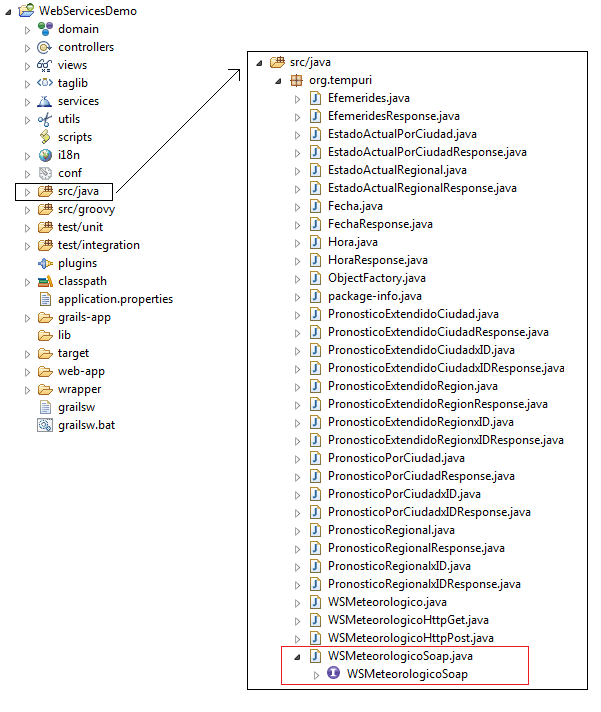
Figura : Missatge obtingut a la consola després d'executar exitosament la instrucció wsdl2java

El que ha passat és que s’han descarregat del web service remot que ofereix els serveis les classes Java equivalents a les operacions compatibles a executar al nostre projecte per fer-les servir.

Per finalitzar la configuració del nostre client de web services s’ha de identificar la variable clientInterface. Després només cal executar el projecte i no sempre que s’executi *“wsdl2java”* s’ha de tenir comenta/eliminada la variable *clientInterface*.

#### Obtenir el ClientInterface

Les classes Java que genera *“wsdl2java”* es troben a la carpeta *src/java* del projecte. Aquí és on es guarden els arxius descarregats del web service. En el package s’haurà creat una interfície amb un nom ***xxx\_Soap.java*** o similar. Aquest serà el nom del *clientInterface* a posar en l’arxiu ***Config.groovy*** de manera completament qualificada. En aquest cas seria la classe seria ***WSMeteorologicoSoap.java***, per tant s’hauria de posar clientInterface = org.tempuri.WSMeteorologicoSoap. Després de l’execució de la instrucció *“wsdl2java”* tenim el següent escenari on es poden identificar tots els elements descrits:

 Imatge : Arxius generats per la instrucció wsdl2java

### Ús del webservice

A continuació es mostra el codi d’un controlador que es connecta al web service anteriorment configurat de la forma més habitual en Grails:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **package** webserviceclientsoapcxf  **import** org.tempuri.WSMeteorologicoSoap  **class** PruebaController {  //Aquest nom ha de coincidir amb el de l'apartat/node cxf - client  //de l'arixu Config.groovy. NO pot ser un nom de variable qualsevol.  WSMeteorologicoSoap tiempoClient  ...  def meteoCxfHora (){  String hora  try{  hora = tiempoClient.hora()  println hora  }  catch(Exception e){  String service = e.message  }  [hora:hora]  }  } |

Figura : Exemple de codi d'un client web service seguint l'estils de programació de Grails

També es pot realitzar la mateixa acció amb el següent codi:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **package** webserviceclientsoapcxf  **import** org.tempuri.Hora  **import** org.tempuri.WSMeteorologico  **import** org.tempuri.WSMeteorologicoSoap  **class** PruebaController {  ...  def meteoCxfHora (){  WSMeteorologico h =new WSMeteorologico();  WSMeteorologicoSoap s = h.getWSMeteorologicoSoap()  String hora = s.hora()  println "Hora: "+hora  [hora:hora]  }  } |

Figura : Exemple de codi de client web service seguint l'estil de programació Java

### Peculiaritats del web service RebutNomina

### Exemples de web services

### Solució de problemes

#### Error al descarregar plugins

Si en el moment de refrescar les dependències per tal d’actualitzar els plugins es produeix un error similar al que es mostra a continuació:

|  |
| --- |
| Starting process on PC24094/147.83.168.49  Loading Grails 2.4.3  |Configuring classpath  .  |Environment set to development  ......  |Installing zip cxf-1.1.1.zip...  ...  |Installed plugin cxf-1.1.1  ....................  |Compiling 116 source files  Note: D:\WecWorkspace\gpwebservice\target\work\plugins\cxf-1.1.1\src\java\org\grails\cxf\artefact\DefaultGrailsEndpointClass.java uses unchecked or unsafe operations.  Note: Recompile with -Xlint:unchecked for details. [groovyc] org.codehaus.groovy.control.MultipleCompilationErrorsException: startup failed:  [groovyc] D:\WecWorkspace\gpwebservice\target\work\plugins\cxf-1.1.1\src\groovy\org\grails\cxf\utils\CxfConfigHandler.groovy: 3: unable to resolve class org.codehaus.groovy.grails.commons.ConfigurationHolder  [groovyc] @ line 3, column 1.  [groovyc] import org.codehaus.groovy.grails.commons.ConfigurationHolder as CH  [groovyc] ^  [groovyc]  [groovyc] 1 error  .Error  |  Compilation error: startup failed:  D:\WecWorkspace\gpwebservice\target\work\plugins\cxf-1.1.1\src\groovy\org\grails\cxf\utils\CxfConfigHandler.groovy: 3: unable to resolve class org.codehaus.groovy.grails.commons.ConfigurationHolder  @ line 3, column 1.  import org.codehaus.groovy.grails.commons.ConfigurationHolder as CH  ^  1 error  Error |  org.codehaus.groovy.grails.cli.ScriptExitException  Error |  at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)  <la traça d’error s’exten a 655 línies més>  Error |  at gant.Gant$\_dispatch\_closure5.doCall(Gant.groovy:381)  Error |  ... 76 more |

Figura : Error al descarregar plugins per a web services

És degut a que no es troba la versió indicada del plugin per poder-la descarregar del repositori. Una opció seria buscar un repositori que la tingués i afegir-lo als repositoris del ***BuildConfig.groovy***. Una altra és executar al *Grails Command History* ( ) la instrucció *“install-plugin cxf”* i s’obtindrà un resultat similar a:



|  |
| --- |
| Since Grails 2.3, it is no longer possible to install plugins using the install-plugin command.  Plugins must be declared in the grails-app/conf/BuildConfig.groovy file.  Example:  grails.project.dependency.resolution = {  ...  plugins {  compile ":cxf:2.0.1"  }  } |

Figura : Resultat de la instrucció install-plugin cxf

Aixó indica que l’última versió disponible del plugin cxf és la 2.0.1, i que per afegir-la al nostre projecte no es pot fer servir mitjançant la línia de comandes (desde Grails 2.3), si no que s’ha d’afgir al ***BuilConfig.groovy***.

Llavors la solució és canviar la versió a l’arxiu BuildConfig.groovy del plugin cxf.

#### No fa cas el Config.groovy

#### Forked Grails VM exited with error

Aquest és només el missatge final de la següent traça d’error:

|  |
| --- |
| |Loading Grails 2.4.3  |Configuring classpath  .  |Environment set to development  .................................  |Packaging Grails application  ...........  |Compiling 1 source files  .............................  |Running Grails application  Web service client tiempoClient cannot be created before setting the clientInterface=[:] and serviceEndpointAddress=http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx properties  Error |  2015-04-20 14:31:55,437 [localhost-startStop-1] ERROR CxfClientGrailsPlugin - Web service client tiempoClient cannot be created before setting the clientInterface=[:] and serviceEndpointAddress=http://www.imn.ac.cr/MeteorologicoWS/MeteorologicoWS.asmx properties  Error |  2015-04-20 14:32:02,683 [localhost-startStop-1] ERROR context.GrailsContextLoaderListener - Error initializing the application: Error creating bean with name 'tiempoClient': Initialization of bean failed; nested exception is org.springframework.beans.TypeMismatchException: Failed to convert property value of type 'java.lang.String' to required type 'java.lang.Class' for property 'clientInterface'; nested exception is java.lang.IllegalArgumentException: Could not load class []!  Message: Error creating bean with name 'tiempoClient': Initialization of bean failed; nested exception is org.springframework.beans.TypeMismatchException: Failed to convert property value of type 'java.lang.String' to required type 'java.lang.Class' for property 'clientInterface'; nested exception is java.lang.IllegalArgumentException: Could not load class []!  Line | Method  ->> 262 | run in java.util.concurrent.FutureTask  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  | 1145 | runWorker in java.util.concurrent.ThreadPoolExecutor  | 615 | run . . . in java.util.concurrent.ThreadPoolExecutor$Worker  ^ 744 | run in java.lang.Thread  Caused by TypeMismatchException: Failed to convert property value of type 'java.lang.String' to required type 'java.lang.Class' for property 'clientInterface'; nested exception is java.lang.IllegalArgumentException: Could not load class []!  ->> 262 | run in java.util.concurrent.FutureTask  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  | 1145 | runWorker in java.util.concurrent.ThreadPoolExecutor  | 615 | run . . . in java.util.concurrent.ThreadPoolExecutor$Worker  ^ 744 | run in java.lang.Thread  Caused by IllegalArgumentException: Could not load class []!  ->> 262 | run in java.util.concurrent.FutureTask  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  | 1145 | runWorker in java.util.concurrent.ThreadPoolExecutor  | 615 | run . . . in java.util.concurrent.ThreadPoolExecutor$Worker  ^ 744 | run in java.lang.Thread  Error |  Forked Grails VM exited with error |

Figura : Error degut a intentar connectar amb el web service sense la variable clientInterface adequada o inexistent

Aquest error és degut a que no s’ha posat o descomentat a l’arxiu ***Config.groovy*** la variable clientInterface després d’executar *“wsdl2java”*.

# Annexos

## 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.*

## Activar 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.*

## Crear un Taglib personalitzat

Fer botó dret sobre el projecte *→ New → Taglib* o bé fer botó dret sobre el projecte *→ New → Other* i escriure *Taglib*, acte seguit apareixerà la paraula *Taglib* a sota de la carpeta *Grails.*

## Configuració ràpida del plugin CKEditor per a l’aplicació Editor

Es mostra la instal·lació i configuració del plugin CKEditor de manera ràpida.

Després es mostren les modificacions que s’han de tenir en compte, perquè no són usuals, per adaptar-lo a les nostres necessitats.

### Instal·lació i configuració

#### Arxiu BuildConfig.groovy

En l’arxiu **BuildConfig.groovy** s’ha d’afegir la següent línea:

|  |
| --- |
| grails.project.dependency.resolution = {  ...  plugins {  ...  compile ":ckeditor:4.4.1.0"  ...  }  } |

Figura : Instal·lació del plugin CKEditor

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

#### Arxiu Config.groovy

Al final de l’arxiu ***Config.groovy*** de l’aplicació incloure el següent codi:

|  |
| --- |
| ckeditor {  config = "/js/ckeditor/ckeditorconfig.js"  //skipAllowedItemsCheck = false  defaultFileBrowser = "ofm"  upload {  basedir = "/app/rrhh\_resources/"  baseurl ="/${grails.util.Metadata.*current*.'app.name'}*/../rrhh\_resources/"*  overwrite = **false**  link {  browser = **true**  upload = **false**  allowed = ['pdf', 'doc', 'docx']  denied = []  }  image {  browser = **true**  upload = **true**  allowed = ['jpg', 'gif', 'jpeg', 'png']  denied = []  }  flash {  browser = **false**  upload = **false**  allowed = ['swf']  denied = []  }  }  } |

Figura : Arxiu Config.groovy

#### Arxiu config.js (ckeditorconfig.js)

En la carpeta *web-app/js* de l’aplicació es crea una carpeta anomenada *ckeditor* i dintre d’ella un arxiu anomenat ***ckeditorconfig.js*** amb el següent contingut:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  88 86  87  88  89  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130 | CKEDITOR.editorConfig = **function**( config ) {  //Define changes to default configuration here. For example:  config.language = 'ca';  config.allowedContent=**true**  config.extraAllowedContent = '\*(\*)';  //Fulles d'estil del CKEditor  config.contentsCss = "/gcrrhh/css/contents.css"    config.toolbar = 'Editor';  config.toolbar\_Editor = [  { name: 'document', items : [ 'Source','-','Save','NewPage','DocProps','Print'] },  { name: 'clipboard', items : [ 'Cut','Copy','Paste','PasteText','PasteFromWord','-','Undo','Redo' ] },  { name: 'editing', items : [ 'Find','Replace','-','SelectAll','-','SpellChecker', 'Scayt' ] },  { name: 'forms', items : [ 'Form', 'Checkbox', 'Radio', 'TextField', 'Textarea', 'Select', 'Button', 'ImageButton', 'HiddenField' ] },  '/',  { name: 'basicstyles', items : [ 'Bold','Italic','Underline','Strike','Subscript','Superscript','-','RemoveFormat' ] },  { name: 'paragraph', items : [ 'NumberedList','BulletedList','-','Outdent','Indent','-','Blockquote','CreateDiv',  '-','JustifyLeft','JustifyCenter','JustifyRight','JustifyBlock','-','BidiLtr','BidiRtl' ] },  { name: 'links', items : [ 'Link','Unlink','Anchor' ] },  { name: 'insert', items : [ 'Image','Flash','Table','HorizontalRule','Smiley','SpecialChar','PageBreak','Iframe' ] },  '/',  { name: 'styles', items : [ 'Styles','Format','Font','FontSize' ] },  { name: 'colors', items : [ 'TextColor','BGColor' ] },  { name: 'tools', items : [ 'Maximize', 'ShowBlocks'] },  { name: 'tools', items : [ 'About'] },  ];    config.indentOffset = 5;    //Configuració Codemirror per CKeditor ---> Edició de codi amb "Codi font"  //Només activar config.extraPlugins i config.extraPlugins si existeix  // /{nombre\_de\_la\_aplicación}/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/web-app/js/ckeditor/plugins  config.extraPlugins = 'codemirror';  config.codemirror = {  theme: 'default',  lineNumbers: **true**,  lineWrapping: **true**,  matchBrackets: **true**,  autoCloseTags: **true**,  autoCloseBrackets: **true**,  enableSearchTools: **true**, // Whether or not to enable search tools, CTRL+F (Find), CTRL+SHIFT+F (Replace), CTRL+SHIFT+R (Replace All), CTRL+G (Find Next), CTRL+SHIFT+G (Find Previous)  enableCodeFolding: **true**,  enableCodeFormatting: **true**,  autoFormatOnStart: **true**,  autoFormatOnUncomment: **true**,  highlightActiveLine: **true**,  highlightMatches: **true**,  showFormatButton: **true**,  showCommentButton: **true**,  showUncommentButton: **true**  };  };  //Permet la indentació quan s'envia el text al controlador, així en el TextArea GSP no es perd la indentació.  CKEDITOR.on( 'instanceReady', **function**( ev ) {    //Indentació tags <div>  **var** blockTags = ['div'];    **var** rules = {  indent : **true**,  breakBeforeOpen : **true**,  breakAfterOpen : **false**,  breakBeforeClose : **true**,  breakAfterClose : **true**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }    //Indentació tags <p> i <span>  blockTags = ['p', 'span'];strong    rules = {  indent : **true**,  breakBeforeOpen : **true**,  breakAfterOpen : **false**,  breakBeforeClose : **false**,  breakAfterClose : **true**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }    //Indentació tipus font bold  blockTags = ['strong'];    rules = {  indent : **false**,  breakBeforeOpen : **false**,  breakAfterOpen : **false**,  breakBeforeClose : **false**,  breakAfterClose : **false**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }  }); |

Figura : Arxiu ckeditorconfig.js

#### Pàgines GSP

Per incloure l’editor s’ha d’afegir en la pàgina GSP el següent codi:

|  |
| --- |
| <head>  <ckeditor:resources/>  </head>  <body>  <ckeditor:editor id="arxiuContingut" name="arxiuContingut" height="400px" width="100%" userSpace="${space}">${params?.arxiuContingut?:arxiu?.contingut}</ckeditor:editor>  </body> |

Figura : Afegir CKeditor de manera ràpida a una vista GSP

Si es desitja que el rich editor es comporti com un Text Area per a l’edició de codi HTML/GSP llavors no s’haurà de carregar la llibreria <ckeditor:resources/>, llavors normalment es fa servir:

|  |
| --- |
| <head>  <g:if test="${arxiu?.permetreGSP==0 || arxiu?.permetreGSP==null}">  <ckeditor:resources/>  </g:if>  </head> |

Figura : Afegir libreria <ckeditor:resources> depenent de si es fa servir CKEditor o editor de codi HTML/GSP

El fer servir espais entre els tags d’apertura i tancament <ckeditor> i el text que es mostrarà en l’editor com es pot veure a continuació marcat en groc:

|  |
| --- |
| <ckeditor:editor id="arxiuContingut" name="arxiuContingut" height="400px" width="100%" userSpace="${space}">\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_${params?.arxiuContingut?:arxiu?.contingut}  </ckeditor:editor> |

Figura : Codi GSP que provoca espais/tabulacions afegides en l'editor de codi HTML/GSP

Provocarà que quan no es carregui la llibreria <ckeditor:resources/>, el Text Area que surt per defecte afegeixi espais/tabulacions de manera automàtica, ja que aquest salts de línia i espais es compten com a caràcters.

#### Controladors

En l’acció del controlador que carrega la vista en la que hi ha l’editor s’ha d’enviar la variable anomenada *space*, que serà la que es farà servir per identificar l’espai on s’emmagatzemaran les imatges. Per exemple: [ param1: "valor 1", param2: "valor 2", space: "PAS"].

Per recuperar les dades de l’editor en el controlador s’ha de llegir la variable params.arxiuContingut.

### Modificacions

Per estalviar passos quan es descarrega l’aplicació gcrrhh desde zero, s’han pujat al repositori només els arxius/carpetes que s’han modificat dintre del plugin. Llavors el procediment a seguir és:

* Descarregar el projecte gcrrhh.
* Obrir l’entorn de desenvolupament GGTS i importar el projecte. Llavors es descarreguen tots el plugins i aixó esborra tots els elements que s’han baixat amb el projecte el primer cop:
  + L’arxiu que es troba a la ruta: */gcrrhh/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/src/groovy/org/gualdi/grails/plugins/ckeditor/utils/PathUtils.groovy*.
  + La carpeta *codemirror* que hi ha dintre de la ruta */gcrrhh/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/web-app/js/ckeditor/plugins*.
* Fer *Tortoise SVN → Revert...* i després *Tortoise SVN → update* del projecte per tenir la mateixa versió de l’arxiu ***PathUtils.groovy*** que al repositori i per tornar a baixar la carpeta *codemirror*.
* Fer fer botó dret → *Refresh* del projecte al GGTS.

En cas de dubtes o problemes es poden consultar els següents apartats on es pot trobar de manera detallada com realitzar cada modificació:

* **3.2.4.1 Modificació al plugin de la ruta de pujada d’arxius al servidor**
* **3.2.5 Codemirror per a rich editor**

## Arxiu *config.js* (CKEditor) de l’aplicació Editor (gcrrhh)

L’arxiu de configuració ***config.js*** que es fa servir a l’aplicació Editor (gcrrhh) és:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  88 86  87  88  89  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130 | CKEDITOR.editorConfig = **function**( config ) {  //Define changes to default configuration here. For example:  config.language = 'ca';  config.allowedContent=**true**  config.extraAllowedContent = '\*(\*)';  //Fulles d'estil del CKEditor  config.contentsCss = "/gcrrhh/css/contents.css"    config.toolbar = 'Editor';  config.toolbar\_Editor = [  { name: 'document', items : [ 'Source','-','Save','NewPage','DocProps','Print'] },  { name: 'clipboard', items : [ 'Cut','Copy','Paste','PasteText','PasteFromWord','-','Undo','Redo' ] },  { name: 'editing', items : [ 'Find','Replace','-','SelectAll','-','SpellChecker', 'Scayt' ] },  { name: 'forms', items : [ 'Form', 'Checkbox', 'Radio', 'TextField', 'Textarea', 'Select', 'Button', 'ImageButton', 'HiddenField' ] },  '/',  { name: 'basicstyles', items : [ 'Bold','Italic','Underline','Strike','Subscript','Superscript','-','RemoveFormat' ] },  { name: 'paragraph', items : [ 'NumberedList','BulletedList','-','Outdent','Indent','-','Blockquote','CreateDiv',  '-','JustifyLeft','JustifyCenter','JustifyRight','JustifyBlock','-','BidiLtr','BidiRtl' ] },  { name: 'links', items : [ 'Link','Unlink','Anchor' ] },  { name: 'insert', items : [ 'Image','Flash','Table','HorizontalRule','Smiley','SpecialChar','PageBreak','Iframe' ] },  '/',  { name: 'styles', items : [ 'Styles','Format','Font','FontSize' ] },  { name: 'colors', items : [ 'TextColor','BGColor' ] },  { name: 'tools', items : [ 'Maximize', 'ShowBlocks'] },  { name: 'tools', items : [ 'About'] },  ];    config.indentOffset = 5;    //Configuració Codemirror per CKeditor ---> Edició de codi amb "Codi font"  //Només activar config.extraPlugins i config.extraPlugins si existeix  // /{nombre\_de\_la\_aplicación}/.link\_to\_grails\_plugins/ckeditor-4.4.1.0/web-app/js/ckeditor/plugins  config.extraPlugins = 'codemirror';  config.codemirror = {  theme: 'default',  lineNumbers: **true**,  lineWrapping: **true**,  matchBrackets: **true**,  autoCloseTags: **true**,  autoCloseBrackets: **true**,  enableSearchTools: **true**, // Whether or not to enable search tools, CTRL+F (Find), CTRL+SHIFT+F (Replace), CTRL+SHIFT+R (Replace All), CTRL+G (Find Next), CTRL+SHIFT+G (Find Previous)  enableCodeFolding: **true**,  enableCodeFormatting: **true**,  autoFormatOnStart: **true**,  autoFormatOnUncomment: **true**,  highlightActiveLine: **true**,  highlightMatches: **true**,  showFormatButton: **true**,  showCommentButton: **true**,  showUncommentButton: **true**  };  };  //Permet la indentació quan s'envia el text al controlador, així en el TextArea GSP no es perd la indentació.  CKEDITOR.on( 'instanceReady', **function**( ev ) {    //Indentació tags <div>  **var** blockTags = ['div'];    **var** rules = {  indent : **true**,  breakBeforeOpen : **true**,  breakAfterOpen : **false**,  breakBeforeClose : **true**,  breakAfterClose : **true**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }    //Indentació tags <p> i <span>  blockTags = ['p', 'span'];strong    rules = {  indent : **true**,  breakBeforeOpen : **true**,  breakAfterOpen : **false**,  breakBeforeClose : **false**,  breakAfterClose : **true**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }    //Indentació tipus font bold  blockTags = ['strong'];    rules = {  indent : **false**,  breakBeforeOpen : **false**,  breakAfterOpen : **false**,  breakBeforeClose : **false**,  breakAfterClose : **false**  };    **for** (**var** i=0; i<blockTags.length; i++) {  ev.editor.dataProcessor.writer.setRules( blockTags[i], rules );  }  }); |

Figura : Arxiu config.js (CKEditor) de l’aplicació Editor (gcrrhh)

## CKEditor: configuració del plugin al *Config.groovy* del gcrrhh

El codi que serveix per configurar el CKEditor que fa servir a l’aplicació Editor (gcrrhh) és i està ubicat a l’arxiu ***Config.groovy*** és el següent:

|  |
| --- |
| ckeditor {  config = "/js/ckeditor/ckeditorconfig.js"  //skipAllowedItemsCheck = false  defaultFileBrowser = "ofm"  upload {  basedir = "/app/rrhh\_resources/"  baseurl ="/${grails.util.Metadata.*current*.'app.name'}*/../rrhh\_resources/"*  overwrite = **false**  link {  browser = **true**  upload = **false**  allowed = ['pdf', 'doc', 'docx']  denied = []  }  image {  browser = **true**  upload = **true**  allowed = ['jpg', 'gif', 'jpeg', 'png']  denied = []  }  flash {  browser = **false**  upload = **false**  allowed = ['swf']  denied = []  }  }  } |

Figura : CKEditor: configuració del plugin al Config.groovy del gcrrhh

## Taglib's per renderitzar HTML des del controlador en una vista GSP

Realment es fa servir a aplicacions com RPPAS, RPPDI i RPBEC.

Aquesta tècnica permetrà que mitjançant l’Editor es puguin crear missatges amb format (negreta, cursiva, color de fons, etc ...) que siguin insertats a vistes GSP de les aplicacions RPPAS, RPPDI i RPBEC. I, per què és necessari? Perqué hi ha vistes GSP d’aquestes aplicacions que requereixen actualitzar el seu text o part d’ell de manera molt freqüent (mensualment) i d’aquesta manera s’evita haver de pujar l’aplicació al servidor cada cop que es realitza una modificació.

Només cal incloure el tag <dades:avisModificacio aliasuri=*"AvisModificacioAdreca"*/> a la vista GSP on es vulgui que aparegui el text i llavors s’ha de [crear un Taglib personalitzat](#_Crear_un_Taglib) amb el següent codi al seu interior:

|  |
| --- |
| **static** *namespace* = "dades"  /\*\*  \* Con atributos en la utilización del TAG  \* <dades:adreca arxiu="algun\_valor"/>  \*/  **def** avisModificacio = { attrs ->  **def** aliasuri = attrs['aliasuri'] ?:''  **def** text = Contingut.*findByAliesUriAndEspai*(aliasuri,1).contingut  out << text  } |

Figura : Taglib personalitzat per a les aplicacions RPPAS, RPPDI i RPBEC

Executarà la closure **avisModificacio** i el resultat l’afegirá a l’HTML de la vista GSP.

# Referències

## Bootstrap

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

## 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/>