



WECEEM CMS

Manual d'usuari

21 de febrer de 2014





Índex

1	Introducció	11
2	Versió standalone.....	12
2.1	Aspectes bàsics.....	12
2.2	Creació d'elements.....	14
2.2.1	Crear un Space	14
2.2.2	Crear un element dintre d'un element	15
2.2.3	Crear un widget	16
2.2.4	Crear un link	16
2.2.5	Crear un script Groovy	16
2.2.6	Crear un Blog	16
2.3	Pujada d'arxius	18
2.3.1	Pujar una imatge al servidor	18
2.3.2	Pujar un arxiu JavaScript	19
2.3.3	Pujar un arxiu d'estils CSS	19
2.4	Inserció d'elements GSP/HTML.....	20
2.4.1	Inserció d'imatges	20
2.4.2	Fer referència a una llibreria JavaScript.....	21
2.4.3	Inserció d'un widget.....	21
2.4.3.1	Enviant dades variables al widget	21
2.4.4	Inserció d'un link	21
2.4.5	Execució d'un script Groovy	22
2.5	Tags i variables d'utilitat.....	24
2.5.1	Tags	24
2.5.1.1	Atribut href.....	24
2.5.2	Variables.....	24
2.5.3	Elements.....	24
2.5.3.1	Logout.....	24



2.6	Peculiaritats.....	26
2.6.1	Edició/restauració/clonació d' <i>Spaces</i>	26
2.6.2	Templates.....	26
3	Administrador del sistema	27
3.1	Bases de dades i descripció de taules	27
3.1.1	Taules d'usuaris.....	28
3.1.1.1	Taula cmsrole	28
3.1.1.2	Taula cmsuser.....	28
3.1.1.3	Taula cmsuser_authorities	28
3.1.2	Taules de continguts	29
3.1.2.1	Taula wcm_status.....	29
3.1.2.2	Taula wcm_space	29
3.1.2.3	Taula wcm_content_version.....	30
3.1.2.4	Taula wcm_content.....	30
3.1.2.5	Taula tags	32
3.1.2.6	Taula tag_links.....	32
3.2	Estructura interna del gestor de continguts	34
3.2.1	Estructura de directoris i creació d'arxius.....	34
3.2.2	Emmagatzematge de fitxers.....	34
3.2.3	Edició i personalització del CMS original.....	35
3.3	Configuració de la plataforma.....	36
3.3.1	Entorn de desenvolupament GGTS	36
3.3.2	Rols i permisos d'accés.....	36
3.3.2.1	Config.groovy	36
3.3.2.2	Policy file	36
3.3.3	Entorn de producció (servidor de producció)	37
3.3.3.1	Fitxer de propietats	37
3.4	Solució de bugs de la plataforma	38



3.4.1	Bases de dades	38
3.4.2	No es pot eliminar l'usuari	38
3.4.3	Problemes d'identificació d'usuari	39
4	Informació i opinions d'usuaris	40
4.1	Llocs webs construïts amb Weceem	40
4.2	Opinions	40
5	Annexos.....	41
5.1	Codi font.....	41
5.1.1	DataSource.groovy	41
5.1.2	Fitxer de propietats	42
5.2	Entorn de desenvolupament.....	43
5.2.1	Creació d'un fitxer WAR	43





Índex de imatges

Imatge 1: Paràmetres per crear un nou <i>Space</i>	14
Imatge 2: Contingut d'un <i>Space</i> per defecte.....	15
Imatge 3: Aspecte d'un element seleccionat.	15
Imatge 4: Pujar una imatge al servidor.	18
Imatge 5: Pujar una imatge al servidor desde el PC.....	18
Imatge 6: Inserir una imatge.	20
Imatge 7: Inserir imatge en mode gràfic.	20
Imatge 8: Inserir un link creat amb l'editor. Primer cas.....	22
Imatge 9: Inserir un link creat amb l'editor. Segon cas.....	22
Imatge 10: Com fer servir l'atribut href dintre d'un Folder.	24





Índex de taules

Taula 1: Camps de la taula cmsrole.....	28
Taula 2: Camps de la taula cmsuser.	28
Taula 3: Camps de la taula cmsuser_authorities.	29
Taula 4: Camps de la taula wcm_status.	29
Taula 5: Camps de la taula wcm_space.....	29
Taula 6: Camps de la taula wcm_content_version.	30
Taula 7: Camps més rellevants de la taula wcm_content.....	31
Taula 8: Altres camps interessants de la taula wcm_content.....	32
Taula 9: Camps de la taula tags.....	32
Taula 10: Camps de la taula tag_links.	33



1 Introducció

Aquest document pretén mostrar totes les possibilitats que ofereix el gestor de continguts o CMS (Content Manager System) Weceem desenvolupat en la plataforma Grails.

Un apartat consisteix en explicar com funcionen els elements bàsics de la versió *standalone* que és la que es faria servir desplegada en un servidor de producció Apache Tomcat o similar.

Altres apartats estan dedicats a intentar entendre la estructura interna de la plataforma Weceem. Com es guarden les dades a la base de dades, que carpetes i arxius genera i on, inclús, com modificar les vistes originals del CMS. Tot per poder aprofitar al màxim el gestor de continguts i poder personalitzar-lo a les nostres necessitats.

Finalment es recull diferent informació per tal de poder evaluar l'impacte de l'ús en producció d'aquesta plataforma.

2 Versió standalone

L'empaquetat WAR és la versió que es desplega a un servidor de producció, i per tant, és el producte final. La versió *standalone* descarregable de la web () és un arxiu WAR amb una configuració predeterminada.

Podem baixar el codi font, personalitzar-lo per tal d'adaptar la plataforma a les nostres necessitats i crear el nostre propi WAR.

2.1 Aspectes bàsics

Un node pot ser qualsevol element: un **Server Directory**, un **Folder**, una pàgina **HTML**, etc, del que penjen altres elements.

Si es crea un nou contingut (+ **New Content**), aquest es crearà dintre de la carpeta que hi hagi seleccionada en el moment de la creació. És a dir, si es té seleccionada la carpeta **Widgets**, en el moment de fer + **New Content** → **Server File** → **Create**, l'arxiu que podrem crear/pujar en el servidor s'ubicarà dintre d'aquesta carpeta.

Quan es vol fer servir una funció carregada des d'un arxiu o carregar un estil CSS, l'arxiu en qüestió ha de ser marcat com a **Published**. En canvi, si el que es vol és carregar una imatge a una pàgina HTML no cal que la imatge estigui marcada com a **Published**. La imatge només ha de ser marcada com a **Published** si es fa servir en alguna funció JavaScript, en aquest cas no pot tenir altre estat.

El camp **Alias URI** és el que es fa servir per referenciar una fulla d'estils, un arxiu JavaScript, una imatge, etc. **Title** és el nom que veiem de l'arxiu a l'arbre de continguts del CMS (Content Manager System).

Un **Server Directory** té el mateix valor a **Title** i a **Alias URI**. Per a aquest tipus d'element es crea un directori al servidor amb el mateix nom que el valor de **Alias URI**.

Un **Folder** pot tenir un valor al camp **Title** i un altre a **Alias URI**. Per a aquest element no es crea cap directori al servidor.

No es pot esborrar un node del que contingui elements.

Pot existir al mateix temps un **Folder** amb *JavaScript* al seu camp **Title** i *js* al seu **Alias URI**, a l'arbre de continguts **JavaScript (/js – Folder)**, i un **Server Directory** anomenat *js*, a l'arbre de continguts **js (/js – Server Directory)**.


No es poden pujar dos **Server Files** que tinguin el mateix nom (el nom original de l'arxiu, el que apareix a **File**, NO el **Title**). Però sí que es poden pujar dos **Server Files** amb noms diferents i posar-los el mateix nom a **Title**.

No es pot crear un **Server File** al node arrel.

Crear un nou **JavaScript Source** i pegar dintre una llibreria sencera com podria ser JQuery no és bona idea perquè triga molt en pujar l'arxiu a la base de dades degut a la seva grandaria. A més si es talla la pujada després s'ha d'esborrar desde la base de dades de forma manual, amb els problemes que comporta això (problemes per accedir a un arxiu corrupte, s'ha de modificar el valor actual i el següent de la seqüència).

Problemes de càrrega d'arxius a pàgines HTML

Donat el següent cas:

▶ Libraries (/libs - Folder)	Published	unknown
▼ JavaScript (/js - Folder)	Published	unknown
▼ JQuery (/JQuery - Folder)	Published	admin
 JQuery (/jquery.js - Server File)	Published	admin
▶ resources (/resources - Server Directory)	Published	unknown

Si es vol carregar l'arxiu jquery.js s'hauria de fer servir:

```
<script type="text/javascript" src="{wcm.createLink(path:'js/jquery/jquery.js')}"></script>
```

Això NO funciona. Com es pot veure *js* i *JQuery* són dos elements tipus **Folder**, i l'últim conté un **Server File**. Si es vol carregar un arxiu tipus **Server File** que estigui dintre d'un **Folder** el CMS no ho permet. En canvi, si *js* i *JQuery* fossin dos **Server Directori** sí que carregaria correctament l'arxiu *jquery.js*. També seria correcte que *js* i *JQuery* fossin dos **Folders** i *jquery.js* fos un **JavaScript Source**. Això vol dir que si tenim **Folders** el codi ha de ser escrit a ma.

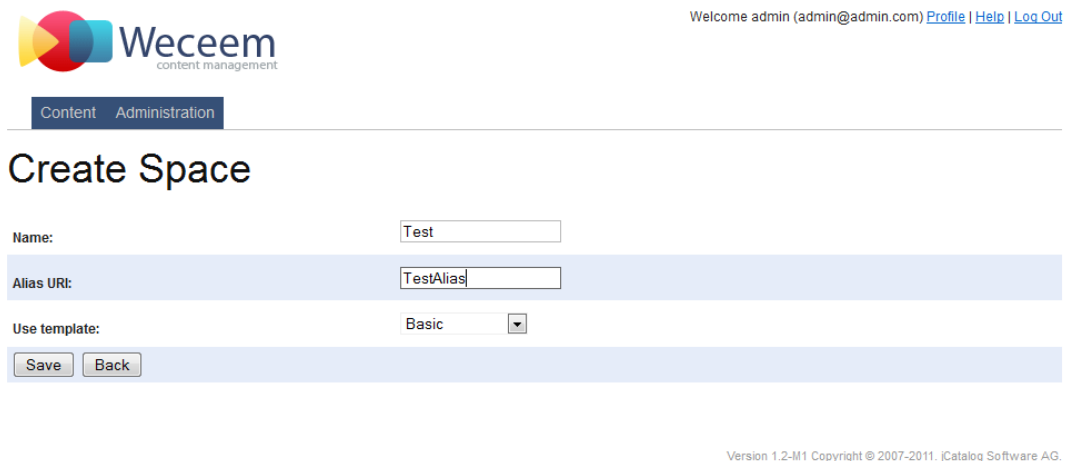
2.2 Creació d'elements

2.2.1 Crear un Space

L' *Spaces* és el contingut que es serveix al visitant. Cada *Space* representa una aplicació web diferent. Entre diferents *Spaces* no es comparteixen recursos.

Per crear un nou *Space* he de fer:

- Entrar com a administradors de la plataforma.
- Fer clic a la pestanya **Administration**.
- Fer clic al link **Spaces** (Add, edit, ...).
- Clic **Add**.
- Donar nom



Welcome admin (admin@admin.com) [Profile](#) | [Help](#) | [Log Out](#)

Content Administration

Create Space

Name:

Alias URI:

Use template: ▼

Version 1.2-M1 Copyright © 2007-2011. JCatalog Software AG.

Imatge 1: Paràmetres per crear un nou Space.

Name és el nom que apareix al desplegable **Space** de la pestanya **Content**.

Si es selecciona l'*Space* que s'ha creat es pot veure com que de bon inici hi tenim:



Content Administration

+ New Content More Actions

Space: Test

Q

Page	Status	Created By	Last changed
Welcome (/index - HTML Page)	Published	unknown	On 2010/11/18 at 03:39:28
Image (/Image - Server Directory)	Published	unknown	9 minutes ago
Our first Weceem Blog (/blog - Blog)	Published	unknown	9 minutes ago
Widgets (/widgets - Folder)	Published	unknown	9 minutes ago
Templates (/templates - Folder)	Published	unknown	9 minutes ago
Stylesheets (/css - Folder)	Published	unknown	9 minutes ago
Libraries (/libs - Folder)	Published	unknown	On 2011/03/28 at 12:54:46
JavaScript (/js - Folder)	Published	unknown	On 2009/09/29 at 11:07:04
resources (/resources - Server Directory)	Published	unknown	On 2011/03/28 at 01:08:11
Views (/views - Folder)	Published	unknown	9 minutes ago
Scripts (/scripts - Folder)	Published	unknown	9 minutes ago

+ New Content More Actions

Imatge 2: Contingut d'un *Space* per defecte.

Si volem accedir al lloc crear mitjançant un navegador hem d'escriure la següent direcció:

`https://merlot.upc.edu/weceem-1.2-M1/TestAlias/index`

S'ha fet servir el nom indicat al paràmetre **Alias** en el formulari de creació. En lloc del nom de l'arxiu **Welcome** es fa servir **/index** com s'indica a la imatge.

2.2.2 Crear un element dintre d'un element

Per crear un element (o node) dintre d'un altre, primer s'ha de seleccionar l'element que es vol que el contingui.

Per seleccionar un element de qualsevol tipus s'ha de fer click en qualsevol zona de la fila on es trobi l'element en la que no aparegui una icona amb forma de ma. A la imatge següent es mostra un exemple d'on es podria fer click per seleccionar el següent element: (interior zona vermella)

	ABOUT us (/about - HTML Page)	Published	unknown
	Contact us (/contact - HTML Page)	Published	unknown
	Image (/Image - Server Directory)	Published	unknown

Imatge 3: Aspecte d'un element seleccionat.

Una vegada seleccionat l'element, la seva fila tindrà un color groguenc, i tots els elements que es creïn nous fent **+ New Content** → ... penjaran

En aquest exemple s'ha creat un HTML dintre d'un altre HTML, cosa que en principi no té gaire sentit però que mostra les possibilitats que ofereix el CMS.

2.2.3 Crear un widget

Fer **+ New Content** → **Widget** → **Create** dintre de la carpeta desitjada.

Per inserir-lo, consultar l'apartat 2.4.3 Inserció d'un widget.

2.2.4 Crear un link

Serveix per poder reutilitzar el link inserint-lo en forma de tag predefinit al codi GSP.

- Seleccionar el lloc on es vol que estigui contingut el link. Aquest element no ha de ser necessàriament de caràcter **Published**.
- Clickar a **+ New Content** i triar l'opció **External Link**.
- **Alias URI** serà el nom que es farà servir per tal que el link sigui operatiu.
- A **URL** s'ha d'indicar on es vol fer el redireccionament.

2.2.5 Crear un script Groovy

Serveix per poder escriure un script Groovy que tindrà que ser vinculat mitjançant un element de tipus **Action** ala pàgina HTML.

- Seleccionar el lloc on es vol que estigui contingut l'script.
- Clickar a **+ New Content** i triar l'opció **Groovy Script**.
- A **Content** s'escriu el contingut de l'script.
- **Alias URI** serà el nom que es farà servir quan es vulgui referenciar l'script.

Un exemple d'script podria ser el següent:

```
println "----->GROOVY SCRIPT<-----"
```

Aquest script escriurà a la consola del server on estigui ubicat el CMS Weceem el missatge que hi ha entre cometes. Per tal de poder utilitzar/cridar l'script s'han de seguir el passos de l'apartat .

2.2.6 Crear un Blog



Per crear un blog s'han de fer un mínim de dues coses:

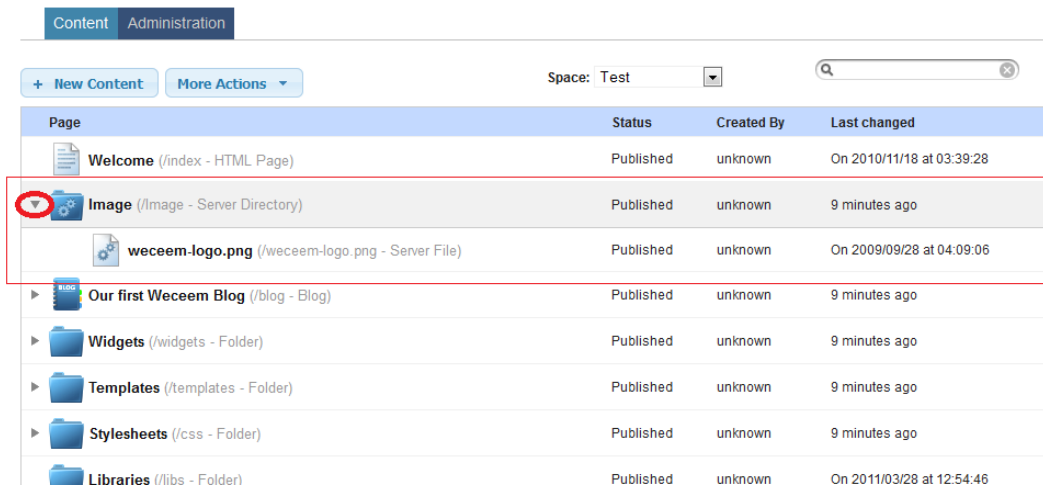
- Crear la carpeta que contindrà el blog:
- Crear una entrada per ser consultada.

2.3 Pujada d'arxius

2.3.1 Pujar una imatge al servidor

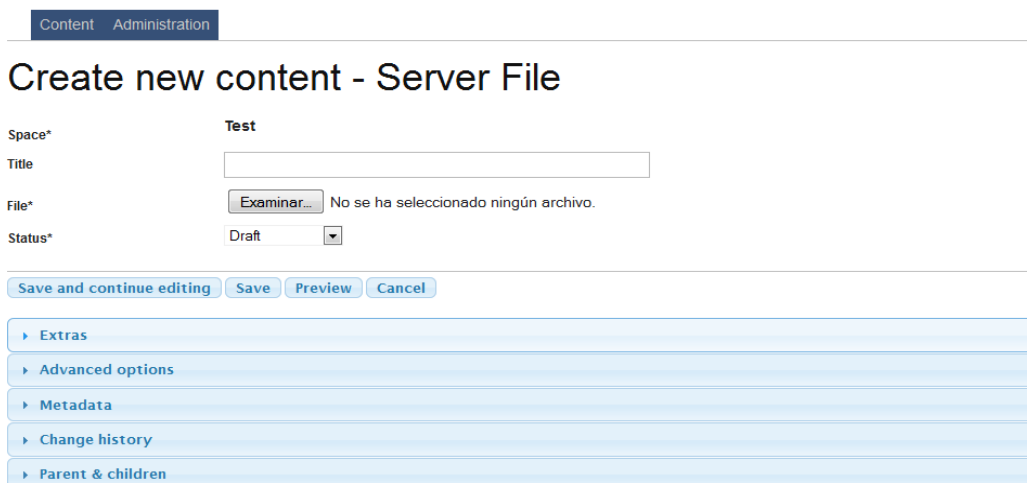
Si es vol pujar una imatge al servidor s'han de realitzar els següents passos:

- Desplegar la carpeta **Image** fent clic a la fletxeta de l'esquerra i seleccionant-la:



Imatge 4: Pujar una imatge al servidor.

- A continuació **+ New Content** i triar l'opció **Server File**.
- Clic a **Create** i apareixerà:



Content Administration

Create new content - Server File

Space* Test

Title

File* No se ha seleccionado ningún archivo.

Status* Draft

- ▶ Extras
- ▶ Advanced options
- ▶ Metadata
- ▶ Change history
- ▶ Parent & children

Imatge 5: Pujar una imatge al servidor desde el PC.

- Llavors se li posa un nom i es clicka a **Save**.

2.3.2 Pujar un arxiu JavaScript

- Desplegar la carpeta **JavaScript (/js – Folder)** i seleccionar-la.
- A continuació **+ New Content** i triar l'opció **Server File**.
- Llavors es tria l'arxiu que es desitja pujar (per exemple **jquery.js**) i a **Title** es posa el nom que es vulgui (per exemple **JQueryFile**).

En el moment de fer referència al fitxer, s'haurà de fer servir el nom original de l'arxiu, **jquery.js**, i no el que s'ha posat a Title, **JQueryFile**.

Per fer referència a l'arxiu JavaScript consultar l'apartat 2.4.2 Fer referència a una llibreria JavaScript.

2.3.3 Pujar un arxiu d'estils CSS




- Desplegar la carpeta **Stylesheets (/css – Folder)** i seleccionar-la.
- A continuació **+ New Content** i triar l'opció **Server File**.
- Llavors es tria l'arxiu que es desitja pujar (per exemple **estils.css**) i a **Title** es posa el nom que es vulgui (per exemple **CSSFile**).

En el moment de fer referència al fitxer, s'haurà de fer servir el nom original de l'arxiu, **estils.css**, i no el que s'ha posat a **Title**, **CSSFile**. És necessari que siguin **Published**.

2.4 Inserció d'elements GSP/HTML

2.4.1 Inserció d'imatges

Si tenim la següent situació:

Page	Status	Created By	Last changed
 Welcome (/index - HTML Page)	Published	unknown	0 seconds ago
▼  Image (/Image - Server Directory)	Published	unknown	4 hours ago
 weceem-logo.png (/weceem-logo.png - Server File)	Published	unknown	On 2009/09/28 at 04:09:06
 Logo (/Logo_UPC.png - Server File)	Published	admin	Never
 hide_end.png (/hide_end.png - Server File)	Published	admin	4 hours ago
 hide_start.png (/hide_start.png - Server File)	Published	admin	Never

Imatge 6: Inserir una imatge.

Podem inserir una imatge a la pàgina HTML/GSP mitjançant el següent codi:

```

```

On **src** ha de tenir la següent ruta:

/weceem-1.2-M1/WeceemFiles/[Space_Alias]/[Directori_Imatge]/[Nom_Imatge]

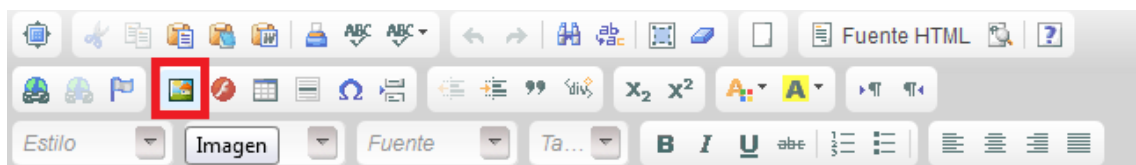
S'han de fer servir les rutes i noms assenyalats en vermell a la Imatge 6. Es fa referència a **Logo_UPC.png** i no a **Logo**. Aquesta manera només funciona a la versió *standalone*.

Una altra forma fent servir codi del gestor de continguts és:

```

```

També es pot fer servir el mode d'edició WYSWYG (edició gràfica) amb el següent botó i seleccionar la imatge desitjada:



Imatge 7: Inserir imatge en mode gràfic.

Per poder accedir a la pantalla d'edició s'ha treure la verificació a la casella **Allow GSP** i donar a **Save**.

2.4.2 Fer referència a una llibreria JavaScript

Per fer servir funcions implementades en una llibreria JavaScript i que es troba a la carpeta **JavaScript (/js – Folder)** i s'anomena **query.js** com a nom original de l'arxiu (en cas de ser una pujada) o **Alias URI** en el cas de ser un arxiu creat al servidor, s'ha de fer servir el següent codi:

```
<script type="text/javascript" src="{wcm.createLink(path:'js/jquery.js')}"></script>
```

2.4.3 Inserció d'un widget

Per inserir un widget al codi GSP s'ha de fer servir:

```
<wcm:widget path="widgets/gadget3"/>
```

Suposant que el widget s'ha inclòs en la carpeta dels widgets **Widgets (/widgets – Folder)** i s'anomeni **gadget3** (nom que té a l'apartat **Extras → Alias URI**).

No fa falta que sigui **Published** per tal de poder fer-lo servir.

2.4.3.1 Enviant dades variables al widget

Si volem passar-li algun tipus de dada al widget es pot fer de la següent manera:




```
<wcm:widget path="widgets/gadget4" model="[Xvariable:'some value',Zvariable:'other value']"/>
```

I al widget les variables es recuperen mitjançant les expressions:

```
{Xvariable}  
{Zvariable}
```

2.4.4 Inserció d'un link

Per tal d'inserir un link com el que es mostra a continuació:

 Editorial (/editorial - HTML Page)	Published	admin
▼  Links (/Links - Folder)	Draft	admin
➔  carsLinkage (/cars - Ext. Link)	Published	admin







Imatge 8: Inserir un link creat amb l'editor. Primer cas.

Que s'ha creat segons s'indica a l'apartat 2.2.4 Crear un link, s'ha de fer servir:

```
<wcm:link path="Links/cars">Aston Martin</wcm:link>
```

No és necessari que la carpeta que el contingui sigui **Published**, però sí ha de ser-ho el link.

En el cas que el link estigui ubicat dintre de la pàgina tal i com es pot veure:

▶  ETSETB (/etsetb - HTML Page)	Published	admin
 FIB (/fib - HTML Page)	Published	admin
▼  UAB Informàtica (/uab - HTML Page)	Published	admin
➔  Linkado a web externa (/Linkado - Ext. Link)	Published	admin
 Editorial (/editorial - HTML Page)	Published	admin
▶  Links (/Links - Folder)	Draft	admin

Imatge 9: Inserir un link creat amb l'editor. Segon cas.

Es fa de manera molt similar:

```
<wcm:link path="uab/Linkado">UPC</wcm:link>
```

A l'atribut **path** es fa servir l'**Alias URI** de la pàgina web que conté el link i l'**Alias URI** del link.

2.4.5 Execució d'un script Groovy

Un cop s'ha creat l'script Groovy seguint els passos de l'apartat 2.2.5 Crear un script Groovy, el que s'ha de fer es crear un element tipus **Action**. Aquest element es farà servir per vincular l'script dintre de la pàgina web.

Creació de l'element **Action**: (suposem que l'anomenem *ScriptedAction1* i es situa a l'arrel de continguts)

- Seleccionar el lloc on es vol que estigui contingut l'element tipus **Action**.
- Clickar a **+ New Content** i triar l'opció **Action**.
- A **Allowed HTTP Methods** s'ha d'indicar els mètodes que s'accepten desde la petició (*GET*, *POST*,...). NO funciona bé, si indiquem *POST* accepta crides desde un formulari amb mètode *GET*.
- A **Script** s'ha d'indicar l'script que es vol executar quan es cridi a l'element **Action**.

A continuació s'ha d'inserir l'element **Action** en una pàgina HTML. A continuació es mostren dos exemples de com fer-ho.

En el primer es farà servir un link per cridar a l'element **Action**. Dintre del codi GSP s'ha d'incloure:

```
<wcm:link src="${wcm.createLink(path:'ScriptedAction1')}">Action Groovy</wcm:link>
```

O bé:

```
<wcm:link src="ScriptedAction1">Action Groovy</wcm:link>
```

En aquest cas, cada vegada que es faci clic al link, es tornarà a recarregar la pàgina.

En el segon exemple es mostra com fer-lo servir des d'un formulari:






```
<form action="ScriptedAction1" method="POST">  
  <button type="submit">Click Me!</button>  
</form>
```

2.5 Tags i variables d'utilitat

2.5.1 Tags

2.5.1.1 Atribut href

En el cas que tinguem la següent situació:

▼  PersonalA (/Personal-A - Folder)	Published	admin
 Marcatges Personals (/Marcatges-personals - HTML Page)	Published	admin
 Marcatges Pendants (/Marcatges-pendants - HTML Page)	Published	admin
 Permisos Per Hores (/Permisos-per-hores - HTML Page)	Published	admin
 Permisos Per Dies (/Permisos-per-dies - HTML Page)	Published	admin

Imatge 10: Com fer servir l'atribut href dintre d'un Folder.

Diferents documents **HTML** en un **Folder**, anomenat per exemple *Personal*, i es vulgui fer referència a un altre document que estigui dintre del mateix folder s'haurà de fer servir la ruta de la següent manera:

```
<a href="Marcatges-pendants">Marcatges pendants</a>
```

2.5.2 Variables

- Idioma: `{node.language}`
- Obtenir l'Espai: `{node.space}`
- Nom de l'usuari: `{wcm.loggedInUserName().encodeAsHTML()}`
- Email de l'usuari: `{wcm.loggedInUserEmail().encodeAsHTML()}`

2.5.3 Elements

2.5.3.1 Logout

Per obtenir el link de *logout* original:

```
<g:link url="{wcm.userLogOutUrl().encodeAsHTML()}"><g:message
```




```
code="admin.user.logout"/></g:link>
```

On:

```
<g:message code="admin.user.logout"/>
```

És el nom que es mostra en la vista. Si es vol que tingui l'aparença de l'aplicació *Tempus* només cal aplicar-li la classe adient:

```
<g:link class="exit" url="{wcm.userLogOutUrl().encodeAsHTML()}"><g:message  
code="admin.user.logout"/></g:link>
```

Cuando se realiza un *logout* se redirige a la página /index del primer *Space* teniendo en cuenta el orden alfabético.

2.6 Peculiaritats

Es descriuen situacions que no es poden considerar pròpiament com a errònies, però que segurament no es comporten com es podria esperar. En altres casos només són indicacions.

2.6.1 Edició/restauració/clonació d'*Spaces*

Els *Spaces* apareixen a la llista de l'administrador en un determinat ordre, que és l'ordre de creació. En el moment que es modifica el nom o l'alias d'un *Space* aquest passa a aparèixer en últim lloc. Això té rellevància degut a que l'índex HTML que carrega la plataforma en determinades ocasions (com quan s'escriu al navegador la direcció del CMS sense indicar cap *Space* en concret) és el de l'*Space* que es troba en primer lloc a la llista.

Quan tenim un *Space* amb un determinat nom i s'importa una estructura d'un *Space* que tenia originalment un altre, el nom de l'*Space* original (no el de l'importat) es manté. És a dir s'importa tot menys el nom i l'alias de l'*Space*.

En el moment d'importar un *Space* es poden donar dues situacions:

- S'hagi comés un error i es vulgui tornar a una versió anterior.
- Es vulgui replicar el contingut de l'*Space* en un altre gestor o en un altre *Space*, per exemple.

En el primer cas tot deuria funcionar correctament. Però en el segon, s'ha experimentat que els fitxers pujats com a **Server Files** amb extensió **.css** i que es troben dintre de **File Directory** es necessari torna a pujar-los.

2.6.2 Templates

Si es modifica l'**ALIAS URI** d'un template, automàticament els arxius HTML que el feien servir veuran com el seu camp **Extres** → **Template** queda modificat també.

3 Administrador del sistema

En aquesta secció es descriuran les relacions entre les diferents taules creades pel CMS per tal de poder funcionar correctament.

A més es mostraran aspectes que són interessants per l'usuari que fa servir el codi font. Per exemple, com editar les pantalles GSP originals del CMS, com eliminar un usuari (en la versió *standalone* una vegada creat l'usuari no es pot esborrar), etc.

Cal dir que una vegada s'ha descarregat el codi font l'aplicació pot funcionar de dues formes:

- Arrancada de manera local desde l'entorn GGTS.
- Crear l'arxiu war a partir del codi font desde l'entorn GGTS i fer que funcioni com una versió *standalone*. L'aventatge d'aquest sistema és que es pot aconseguir una versió *standalone* modificada al nostre gust.

En alguns casos, depenent de com funcioni el nostre CMS la descripció dels següents apartats serà diferent per cada cas.

3.1 Bases de dades i descripció de taules

El CMS crea un total de 10 taules a la base de dades. Això és vàlid tant per la versió que corre sobre l'entorn de desenvolupament GGTS i per la que ho en un contenidor Apache Tomcat o similar.

Totes les claus primàries de totes les taules segueixen la mateixa numeració. És a dir, una clau primària de una taula no pot tenir el mateix valor una altra clau primària d'una altra taula. Aquest control el porta **hibernate_sequence** de l'apartat **Seqüències** (PostgreSQL). Si s'esborra manualment algun camp, és a dir, sense fer servir l'editor de continguts del CMS, s'haurà de modificar també de manera manual aquest apartat per establir el valor de manera correcta.

El codi que es mostra a continuació està ubicat a l'arxiu **DataSource.groovy** i permet crear la bases de dades i fer-la servir en futures sessions: (per a més informació consultar l'apartat 5.1.1 DataSource.groovy)

```
...
production {
    dataSource {
        dbCreate = "update"
        url = "jdbc:postgresql://localhost:5432/weceem"
        username = "user"
        password = "pass"
    }
}
```

```
}
}
...
```

3.1.1 Taules d'usuaris

Emmagatzemen dades referents a l'autenticació d'usuari i rols que poden tenir entre d'altres.

3.1.1.1 Taula cmsrole

És la taula d'autoritats (authorities). Emmagatzema els diferents rols que pot tenir un usuari.

Nom	Definició	Descripció
id [PK]	bigint NOT NULL	L'identificador del rol d'usuari.
authority	character varying(128) NOT NULL	El tipus de rol.

Taula 1: Camps de la taula cmsrole.

3.1.1.2 Taula cmsuser

És la taula d'usuaris. Conté les dades d'un usuari. L'email, la contrasenya encriptada, si està actiu o no el seu identificador (**id**), etc.

Nom	Definició	Descripció
id [PK]	bigint NOT NULL	Identificador de l'usuari.
email	bigint NOT NULL	Email d'usuari.
passwd	character varying(128) NOT NULL	Contrasenya encriptada.
username	character varying(40) NOT NULL	Nom d'usuari.

Taula 2: Camps de la taula cmsuser.

3.1.1.3 Taula cmsuser_authorities

Relaciona l'identificador de la taula d'autoritats amb l'identificador de la taula d'usuaris. D'aquesta forma es pot saber els rols assignats a un usuari.

La taula únicament en té dos camps que són els següents:

Nom	Definició	Descripció
cmsuser_id [PK]	bigint NOT NULL	Identificador de l'usuari. Camp id de la taula cmsuser .
cmsrole_id	bigint NOT NULL	Identificador del rol. Camp id de la taula cmsrole .

Taula 3: Camps de la taula **cmsuser_authorities**.

3.1.2 Taules de continguts

Emmagatzemen dades referents als elements que es troben al CMS i que posteriorment es faran servir per construir els diferents espais.

3.1.2.1 Taula **wcm_status**

Emmagatzema els tipus d'estats que poden tenir els elements.

Nom	Definició	Descripció
id [PK]	bigint NOT NULL	Identificador de l'estat.
public_content	boolean NOT NULL	Indica si l'estat és accessible o no.
description	character varying(80) NOT NULL	Nom de l'estat.

Taula 4: Camps de la taula **wcm_status**.

3.1.2.2 Taula **wcm_space**

Emmagatzema els diferents Spaces creats.

Nom	Definició	Descripció
id [PK]	bigint NOT NULL	Identificador de l'Space.
aliasuri	boolean NOT NULL	Alias URI de l'arxiu (Edició → Extras → Alias URI).
name	character varying(80) NOT NULL	Title de l'arxiu (Edició → Title).

Taula 5: Camps de la taula **wcm_space**.

3.1.2.3 Taula wcm_content_version

Emmagatzema el número de revisió (versió que passar a quedar obsoleta) de cada arxiu al que se li fa **Save** en la pàgina d'edició.

És a dir, un arxiu que s'ha creat només però no s'ha modificat no apareixerà reflectit en aquesta taula. Aquí només apareixeran arxius que han sigut modificats, i la taula reflexarà la versió anterior a la actual.

Nom	Definició	Descripció
id [PK]	bigint NOT NULL	Identificador del tag.
name	character varying(255) NOT NULL	Nom del tag.

Taula 6: Camps de la taula wcm_content_version.

Cada cop que fem un **Save** a la pàgina d'edició en crea un registre nou en aquesta taula.

3.1.2.4 Taula wcm_content

Aquesta taula conté totes les dades principals del elements que hi ha al CMS. Els relaciona amb les seves dependències, diu com s'han d'ordenar i en segons quins casos guarda el seu contingut o sino indica on poder aconseguir-lo.

Els camps més importants d'aquesta taula són:

Nom	Definició	Descripció
aliasuri	character varying(50) NOT NULL	Alias URI de l'arxiu (Edició → Extras → Alias URI).
content_dependencies	character varying(500)	Referència dels recursos que fa servir el contingut. Aquest camp es pot modificar a Edició → Advanced options → Depends on content. Exemple: l'Action actionA(/actionA – Scripted Action) executa l'script New Groovy Script(/gscr – Groovy Script) que es troba a la carpeta Scripts(scripts – Folder) . Llavors en el cas de l'Action, aquest camp valdrà scripts/gscr .
order_index	integer	Ordre que ocupa l'element al node on estigui ubicat, ja sigui a l'arbre general de continguts o node pare (com el cas de la pàgina per defecte Welcome) o dintre de la carpeta/element on es trobi.

parent_id	bigint	Es correspon amb el camp id d'aquesta taula del node al que pertany.
space_id	bigint NOT NULL	Identificador (camp id de la taula wcm_space) de l'Space.
content	character varying(500000)	En el cas que sigui un arxiu editat aquest camp tindrà el codi HTML/CSS/JS. Els Server Files no es guarda aquí el seu contingut, encara que siguin arxius de text.
status_id	bigint NOT NULL	Indica l'estat de l'element i el relaciona amb l' id de l'estat (taula wcm_status).
valid_for	integer	El temps indicat en segons que han de passar per tal que s'actualitzi l'arxiu (Edició → Advanced options → Is updated). Altres valors: 0 → All the time Buit → Template decides

Taula 7: Camps més rellevants de la taula **wcm_content**.

N'hi altres camps a la taula que també és interessant destacar encara que no siguin tan importants com els anteriors:

Nom	Definició	Descripció
changed_by	character varying(255)	Quin usuari va modificar l'arxiu.
created_by	character varying(255)	Quin usuari va crear l'arxiu.
created_on	timestamp without time zone	El moment de creació de l'arxiu.
description	character varying(500)	Camp Edició → Extras → Description .
identifier	character varying(80)	Camp Edició → Extras → Identifier .
language	character varying(3)	Camp Edició → Extras → Language .
meta_copyright	character varying(200)	Camp Edició → Metadata → Copyright .



meta_creator	character varying(80)	Camp Edició → Metadata → Creator .
meta_publisher	character varying(80)	Camp Edició → Metadata → Publisher .
meta_source	character varying(80)	Camp Edició → Metadata → Source .

Taula 8: Altres camps interessants de la taula **wcm_content**.

3.1.2.5 Taula tags

Emmagatzema els tags amb els seus identificadors.

Camps més significatius de la taula:

Nom	Definició	Descripció
content_title	character varying(255) NOT NULL	Nom de l'arxiu que s'ha modificat (Title).
created_by	character varying(255)	Qui va realitzar la modificació. Nom usuari de l'usuari.
created_on	timestamp without time zone	Moment en que es va realitzar la modificació.
revision	integer	Número de revisió.
space_name	character varying(255) NOT NULL	Space al que pertany l'arxiu modificat.

Taula 9: Camps de la taula **tags**.

3.1.2.6 Taula tag_links

Relaciona el tag amb el contingut al que fa referència.

Alguns camps de la taula són:

Nom	Definició	Descripció
tag_id	bigint NOT NULL	Identificador del tag en la taula wcm_tags .
tag_ref	bigint NOT NULL	A quin contingut fa referència el tag. Es correspon amb el camp id de la taula wcm_content de l'element al que fa referència, que és en el que es van incloure els tags en la pàgina d'edició.



3.2 Estructura interna del gestor de continguts

En aquest apartat es pretén mostrar com es reflexa al servidor cada acció que es realitza al CMS com a usuari. Això significa saber on es guarden, en el cas que es faci, els arxius que s'han pujat a l'Space.

3.2.1 Estructura de directoris i creació d'arxius

En el cas que el CMS estigui funcionant sobre l'entorn de desenvolupament GGTS per defecte els arxius i directoris es guarden a la direcció:

```
C:\var\www\weceem.org\uploads
```

En aquest directori és on es generen les carpetes dels diferents Spaces que conté el CMS. Originalment només hi existeixen les carpetes **_ROOT** i **Defaulty**. Per cada nou Space al CMS es crearà una nova carpeta amb el valor indicat a **URI Alias** en el moment de creació.

Aquesta direcció per defecte es troba indicada a l'arxiu **Config.groovy** de l'apartat **conf** del workspace, en la definició **environments** → **development**. Modificant aquesta definició es pot variar la ubicació d'on es guarden els fitxers en aquest mode de funcionament.

En el cas que estiguem utilitzant una versió *standalone* sobre un contenidor Apache Tomcat o similar la ruta d'emmagatzematge per defecte serà:

```
C:\Users\Nom_Usuari\weceem-uploads\WeceemFiles
```

Aquesta configuració es pot canviar modificant l'arxiu **WcmContentRepositoryService.groovy** que es troba ubicat a:

```
C:\Users\Nom_Usuari\.grails\2.3.4\projects\weceem\plugins\weceem-1.2-M1\grails-app\services\org\weceem\services
```

Dintre d'aquest arxiu, la instrucció **System.getProperty("user.home")** ens permet aconseguir el directori de l'usuari.

3.2.2 Emmagatzematge de fitxers

Referent a la pujada d'arxius

Encara que es tingui accés al directori del servidor on està instal·lat el Apache Tomcat on es desplega l'aplicació, NO es poden pujar arxius copiant-los directament al directori del servidor que utilitza el CMS (no es poden copiar arxius per darrere). Això és degut a que quan es puja un arxiu queda enregistrat a la base de dades, i quan es vol referenciar a algun element



sempre es va a buscar on trobar-lo a la base de dades. Copiar-lo directament al directori del servidor és no tenir en registrar l'arxiu. → S'han de pujar tots els arxius d'un en un.

3.2.3 Edició i personalització del CMS original

Per editar la pàgina que carrega l'editor → `wcmEditor/edit.gsp`, que es troba a `C:\User\grails\2.3.4\projects\Weceem\plugins\weceem-1.2-M1\grails-app\views`.

3.3 Configuració de la plataforma

3.3.1 Entorn de desenvolupament GGTS

3.3.2 Rols i permisos d'accés

3.3.2.1 Config.groovy

Una opció per restringir l'accés de visitants a les pàgines és modificar l'arxiu de configuració **Config.groovy** modificant la variable **interceptUrlMap** que es troba a dintre de **grails** → **plugins** → **springsecurity**. Un exemple podria ser el següent:

```
grails {
  plugins {
    springsecurity {
      ...
      securityConfigType = 'InterceptUrlMap'
      interceptUrlMap = [
        '/admin/users/**':    ['ROLE_ADMIN', 'IS_AUTHENTICATED_REMEMBERED'],
        '/admin/**':         ['IS_AUTHENTICATED_REMEMBERED'],
        '/Test/**':          ['ROLE_ADMIN', 'IS_AUTHENTICATED_ANONYMOUSLY'],
        '/Tempus/**':        ['ROLE_GUEST', 'IS_AUTHENTICATED_REMEMBERED'],
        '/Tempus_v2/**':     ['IS_AUTHENTICATED_REMEMBERED'],
        ...
      ]
      ...
    }
  }
}
```

En aquest cas, al *Space* anomenat **Test** i al seu contingut podran accedri-hi només els usuaris que tinguin assignat un rol tipus administrador (**ROLE_ADMIN**). Mentre que a l'*Space* **Tempus** només podran fer-ho els que tinguin un rol de convidat (**ROLE_GUEST**). Encara que el rol d'administrador otorgui més privilegis que el de convidat, a **Tempus** no podran accedir els usuaris que tinguin únicament assignat un **ROLE_ADMIN**, per poder fer-ho a més hauran de tenir assignat el **ROLE_GUEST**.

3.3.2.2 Policy file



A la pàgina del desenvolupador es parla de restringir l'accés a determinats continguts del editor del CMS. Aixó significa que tingui l'usuari que està editant contingut pot veure o no determinats apartats depenent del seu rol.

Per aconseguir-ho s'ha d'establir permisos i restriccions a un fitxer amb extensió .groovy i indicant la seva ubicació en aquest cas a l'entorn de desenvolupament.

Llavors s'ha de passar la següent instrucció com a paràmetre a la línia de comandes del GGTS per tal de poder trobar l'arxiu:

```
-Dweceem.security.policy.path="C:/wec/mypolicy.groovy"
```

L'arxiu anomenat mypolicy.groovy està ubicat al directori C:\wec.

3.3.3 Entorn de producció (servidor de producció)

3.3.3.1 Fitxer de propietats

El fitxer de propietats permet serveix per indicar diferents aspectes com quina base de dades es farà servir, la seva configuració (user i contrasenya) i el driver que s'ha de fer servir, depenent de si és PostgreSQL o MySQL.

NOTA: encara que aquest arxiu és necessari, sobretot per tal que Weceem pugui trobar el driver de la base de dades corresponent, el CMS es connectarà a la base de dades indicada al fitxer **DataSource.groovy**.

Un exemple del fitxer es pot trobar a l'apartat 5.1.2 Fitxer de propietats.

3.4 Solució de bugs de la plataforma

Alguns casos no són errors, només indicacions del funcionament de la plataforma davant determinades situacions.

3.4.1 Bases de dades

Tenim una versió del Weceem CMS funcionant a l'entorn de desenvolupament GGTS i volem fer-la funcionar sobre un contenidor Apache Tomcat.

Llavors el que fem és crear l'arxiu WAR per tal de desplegar-lo al servidor de producció, és a dir es crea una versió *standalone*. Una vegada s'ha desplegat, com es pot veure a l'apartat 3.2.1 Estructura de directoris i creació d'arxius, els directoris fets servir en el GGTS i el servidor de producció no són els mateixos. Per tant, s'hauran de tornar a pujar un per un els arxius, ja siguin imatges, CSS, JavaScript, etc. Només els arxius que s'havien pujat com a Server Files, donat que els CSS i JavaScripts que s'han creat des de l'editor sí que es troben. Aquest últim es poden trobar sense problemes perquè el seu contingut es guarda directament en un registre de la base de dades.

Al crear la versió WAR, si hi havia una base de dades configurada, PostgreSQL o MySQL, diferent de la que ve amb el codi font del CMS, es farà servir la de l'arxiu WAR encara que després a la configuració d'Apache Tomcat s'indiqui una altra cosa a l'arxiu de propietats.

3.4.2 No es pot eliminar l'usuari

A la pantalla d' **Administration** → **Users** es mostra la llista d'usuaris que estan enregistrats al sistema. Es pot veure com cada usuari té un botó per editar-lo i per eliminar-lo. Aquest últim, el botó que teòricament elimina l'usuari no funciona en la versió descarregable del codi font (<https://github.com/jCatalog/weceem-app>) ni en la versió *standalone* per desplegar-la directament a un Apache Tomcat (<http://www.weceem.org/weceem/Download>).

Aquesta pantalla es correspon amb la vista **list** del controlador **CMSUserController** i el formulari on està ubicat aquest botó fa una petició tipus *GET* i crida al mètode **delete** (`<g:link action="delete" ...` a la pàgina GSP). L'error és degut al controlador s'indica que el mètode **delete** només pot rebre peticions de tipus *POST* mitjançant la següent ordre a l'inici del controlador:

```
static allowedMethods = [save: "POST", update: "POST", delete: "POST"]
```

Una solució seria fer el formulari de tipus *POST* i permetre que al mètode edit es pugui accedir mitjançant una petició de tipus *POST*. També es pot modificar la línia de codi anterior i posar *GET* al mètode **delete**.



3.4.3 Problemes d'identificació d'usuari

A vegades és impossible entrar a

4 Informació i opinions d'usuaris

Es recullen diferents enllaços a pàgines que estiguin creades amb el gestor de continguts Weceem per poder valorar el seu rendiment en aplicacions ja fetes de manera professional.

4.1 Llocs webs construïts amb Weceem

Web realitzada amb Weceem:

<http://idmllib.com/ILWebsite/documentation>

Catàleg dels diferents CMS's existents classificats per plataforma de desenvolupament:

http://en.wikipedia.org/wiki/List_of_content_management_systems

4.2 Opinions

<http://www.linkedin.com/groups/Anyone-used-Weceem-Maybe-someone-39757.S.65913362>

Resum: Opinions de tot tipus. És la plataforma a escollir si es desenvolupa amb Grails però totes coincideixen en que encara queda molt camí per arribar al nivell dels Joomla, Drupal, etc. Tot i això reconeixen que el core de l'aplicació és bastant estable. Hi ha opinions que recomanen quina plataforma fer servir en lloc de Weceem.

<http://www.linkedin.com/groups/weceem-CMS-anyone-using-it-67067.S.230747176>

Resum: és una bona plataforma però la costumització necessita temps i experiència. La gent queda sorpresa per Weceem i ho manté com una opció futura tot i decantar-se per altres plataformes ara per ara.

http://www.thorntech.com/2012/09/case-study-high-traffic-mobile-website-using-grails-and-weceem-plugin/?goback=.gde_39757_member_160148664

Resum: article d'una empresa de software que parla del rendiment de Weceem amb un volum alt de visites. Diu que el rendiment és satisfactori.

5 Annexos

5.1 Codi font

Els exemples de codi estan trets de proves reals i que han funcionat correctament.

5.1.1 DataSource.groovy

El servidor de base de dades és PostgreSQL i s'han creat dues bases de dades, una que es diu **Pruebas** i una altra es diu **weceem**. Una serà la base de dades feta servir a test i l'altra a producció respectivament. L'usuari anomenat user amb contrasenya user en aquest cas és el propietari de les bases de dades. L'exemple de codi del fitxer **DataSource.groovy** és:

```
dataSource {
    pooled = true
    driverClassName = "org.postgresql.Driver"
}
hibernate {
    cache.use_second_level_cache=true
    cache.use_query_cache=false
    cache.provider_class = 'net.sf.ehcache.hibernate.EhCacheProvider'
}
environments {
    development {
        dataSource {
            dbCreate = "update" // one of 'create', 'create-drop','update'
            url = "jdbc:postgresql://localhost:5432/Pruebas"
            username = "user"
            password = "pass"
        }
    }
}
test {
    dataSource {
        dbCreate = "update"
        url = "jdbc:postgresql://localhost:5432/Pruebas"
```

```
        username = "user"
        password = " pass"
    }
}
production {
    dataSource {
        dbCreate = "update"
        url = "jdbc:postgresql://localhost:5432/weceem"
        username = "user"
        password = " pass"
    }
}
}
```

Aquesta configuració permet que la primera vegada que s'arrenca l'aplicació es generin totes les taules a la base de dades indicada, i que posteriorment, en les properes arrencades del servidor es facin servir aquestes taules ja creades.

5.1.2 Fitxer de propietats

És el fitxer que en tots els exemples anomenen **weceem.properties**.

```
# Control whether or not connection pooling is enabled
dataSource.pooled=true
# Set the JDBC driver class name - class must be on classpath
dataSource.driverClassName=org.postgresql.Driver
# The user name for the SQL database
dataSource.username=user
# The password for the SQL database
dataSource.password=user
# The database update mode. Leave as "update"
dataSource.dbCreate=update
# The JDBC URL of your database
dataSource.url=jdbc:postgresql://localhost:5432/weceem
# The path to use for storing search index files - MUST be writable
```

```
searchable.index.path=D:/wec
```

Aquesta versió és la que es faria servir si es volgués arrencar l'aplicació *standalone* descarregada de la pàgina web (<http://www.weceem.org/weceem/Download>). Però com ja s'ha comentat a l'apartat 3.3.3.1 Fitxer de propietats, si es genera l'arxiu WAR a partir del codi font per tal d'obtenir una versió *standalone* personalitzada, es farà servir la base de dades indicada a l'arxiu **DataSource.groovy** i no es farà cas de la indicada en el fitxer de propietats.

Per tant, per les versions personalitzades del CMS es pot fer servir un arxiu més reduït com el següent, on s'indica el driver de la base de dades i el directori d'índexs:

```
# Control whether or not connection pooling is enabled
dataSource.pooled=true

# Set the JDBC driver class name - class must be on classpath
dataSource.driverClassName=org.postgresql.Driver

searchable.index.path=D:/wec
```

5.2 Entorn de desenvolupament

5.2.1 Creació d'un fitxer WAR

Per crear un arxiu WAR que posteriorment es pugui desplegar a un servidor de producció es poden seguir dues alternatives:

- Escriure i executar **test war** a la línia de comandes de GGTS i generar un arxiu WAR amb la configuració que es fa servir a l'apartat **test**.
- Escriure i executar **test** a la línia de comandes de GGTS i generar un arxiu WAR amb la configuració que es fa servir a l'apartat **production**.