eXo Platform 3.0 - Content Functions

Chapter 1. Preface

1.1. Getting started with eXo WCM

eXo WCM is a fully integrated Content Management solution inside GateIn. Basically, we're using the extension mechanism inside GateIn to add content features. If you want more infomation about the extension mechanism, you can read the GateIn documentation : http://docs.jboss.com/gatein/portal/3.1.0-FINAL/reference-guide/en-US/html/index.html

This integrated solution provides users with a comprehensive platform for integrating applications and managing and publishing content - all from a single, familiar console.

When starting a new project, you can see eXo WCM as a WCM Java developer Platform. In this reference guide, we will give you guidelines so you can build stable applications using eXo WCM from the inside.

1.2. Packaging

- All package actions in WCM start from the `delivery` folder, so you should go to this folder first.

```
$ cd ecms/delivery
```

1.2.1. Package WCM standalone version - Tomcat bundle

```
$ cd wcm/assembly
$ mvn clean install
```

1.2.2. Package WCM with workflow enabled - Tomcat bundle

```
$ cd wkf-wcm/assembly
$ mvn clean install
```

1.2.3. Make WCM EAR packages to run with jBoss

- Make WCM extension EAR

```
$ cd packaging/wcm/ear
$ mvn clean install
```

- Make WCM demo sites EAR

```
$ cd packaging/ecmdemo/ear
```

```
$ mvn clean install
```

- Make workflow EAR

```
$ cd packaging/workflow/ear
$ mvn clean install
```

Chapter 2. Portlets

2.1. Content Detail

2.1.1. Available preferences

Preference	Type	Default Value	Description
repository	`String`	`repository`	Repository is a place where data are stored and maintained
workspace	`String`	`collaboration`	Workspace is a place where the content is stored
nodeIdentifier	`String`	`N/A`	The UUID or the path of content that you want to show
ShowTitle	`Boolean`	`true`	Show the content title on the top of the portlet
ShowDate	`Boolean`	`false`	Show the content date on the top of the portlet
ShowOptionBar	`Boolean`	`false`	Show a bar with some actions (Print, Back)
ContextEnable	`Boolean`	`false`	Define if the portlet will use the parameter on URL as the path to content to display or not
ParameterName	`String`	`content-id`	Define which parameter will be used to get the content's path
ShowVote	`Boolean`	`false`	Show the result of voting for the displayed content
ShowComments	`Boolean`	`false`	Show the existing comments of this content (if any)

2.1.2. Sample configuration

```
<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
```

```
  <preference>
    <name>workspace</name>
    <value>collaboration</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>nodeIdentifier</name>
    <value>/myfolder/mycontent</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>ShowTitle</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>ShowDate</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>ShowOptionBar</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>ShowPrintAction</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>isQuickCreate</name>
    <value>false</value>
    <read-only>true</read-only>
  </preference>
  <preference>
    <name>ContextEnable</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>ParameterName</name>
    <value>content-id</value>
    <read-only>false</read-only>
  </preference>
</portlet-preferences>
```

Note

In 2.1.x, there are some preferences that are no longer used, for example, ShowPrintAction preference.

2.2. Content List

2.2.1. Available preferences

Preference	Type	Default Value	Description
mode	`String`	`AutoViewerMode`	The mode that the portlet uses to display content: all contents in a specific folder or all specific contents in the portlet.

Preference	Type	Default Value	Description
folderPath	`String`		The path to the folder whose contents are displayed by this portlet.
orderBy	`String`	`publication:liveDate`	The property by which all the contents in the portlet are sorted.
orderType	`String`	`DESC`	The type of the content sort method: ascending or descending
header	`String`		Header of the portlet. It is displayed at the top of the portlet.
automaticDetection	`Boolean`	`true`	This value indicates whether the header of the portlet is chosen to be the title of the folder given in the **folderPath** parameter (true value) or the value given in the **header** paramter above
formViewTemplatePath	`String`	`/exo:ecm/views/templates/content List Viewer/list-by-folder/UIDocumentListPresentationDefault.g`	Path to the template used to display the contents in this portlet
paginatorTemplatePath	`String`	`/exo:ecm/views/templates/content List Viewer/paginators/UIPaginationDefault.gtmpl`	Path to the paginator used to display the contents in this portlet
itemsPerPage	`Integer`	10	Number of contents displayed in every "page" of the portlet.
showThumbnailsView	`Boolean`	`true`	This value indicates whether the content image in this portlet is shown or not.
showTitle	`Boolean`	`true`	This value indicates whether the content title in this portlet is shown or not.
showHeader	`Boolean`	`true`	This value indicates whether the content header in this portlet is shown or not.
showRefreshButton	`Boolean`	`false`	This value indicates whether the **refresh** button is shown in this

Preference	Type	Default Value	Description
			portlet or not.
showDateCreated	`Boolean`	`true`	This value indicates whether the content created date in this portlet is shown or not.
showReadmore	`Boolean`	`true`	This value indicates whether the **Readmore** button is shown in every content of the portlet (After clicking this button, user can read the whole text of the content).
showSummary	`Boolean`	`true`	This value indicates whether the content summary in this portlet is shown or not.
showLink	`Boolean`	`true`	If this value is **true**, the header of every content is also the link to view this content fully. If the value is **false**, the header is considered as a simple text
showRssLink	`Boolean`	`true`	Show the rss link of this portlet.
basePath	`String`	`detail`	Show the page in which the full content is displayed when user clicks to the **Read more** button.
contextualFolder	`String`	`contextualDisable`	Enable/ disable the contextual mode of the portlet. If enabled, the portlet can take the folder path indicated in the URL to display contents.
showScvWith	`String`	`content-id`	The name of the parameter shows the path of content in url to display when clicking the "Read more" button of this content.
showClvBy	`String`	`folder-id`	The name of the parameter shows the path

Preference	Type	Default Value	Description
			of the folder in url to display its contents.

2.2.2. Sample Configuration

```
<portlet-preferences>
  <preference>
    <name>mode</name>
    <value>AutoViewerMode</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>folderPath</name>
    <value/>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>orderBy</name>
    <value>publication:liveDate</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>orderType</name>
    <value>DESC</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>header</name>
    <value/>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>automaticDetection</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>formViewTemplatePath</name>
    <value>/exo:ecm/views/templates/Content List Viewer/list-by-folder/UIContentListPresentationDefault.gtmpl</v
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>paginatorTemplatePath</name>
    <value>/exo:ecm/views/templates/Content List Viewer/paginators/UIPaginatorDefault.gtmpl</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>itemsPerPage</name>
    <value>10</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showThumbnailsView</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showTitle</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showHeader</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showRefreshButton</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
```

```
  <preference>
    <name>showDateCreated</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showReadmore</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showSummary</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showLink</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showRssLink</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>basePath</name>
    <value>detail</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>contextualFolder</name>
    <value>contextualDisable</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showScvWith</name>
    <value>content-id</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showClvBy</name>
    <value>folder-id</value>
    <read-only>false</read-only>
  </preference>
</portlet-preferences>
```

2.3. Search

2.3.1. Available preferences

Preference	Type	Default value	Description
repository	`String`	`repository`	The repository is a place where data are stored and maintained.
workspace	`String`	`collaboration`	The workspace where the content is stored
searchFormTemplatePath	`String`	`/exo:ecm/views/templates/ECM Advance Search/search-form/UIDefaultSearchForm.gtmpl`	The path to the search form template
searchResultTemplatePath	`String`	`/exo:ecm/views/templates/ECM Advance`	The path to the search result template

Preference	Type	Default value	Description
		`Search/search-result/UIDefaultSearchResult.gtmpl`	
searchPaginatorTemplatePath	`String`	`/exo:ecm/views/templates/WCM Advance Search/search-paginator/UIDefaultSearchPaginator.gtmpl`	The path to the search paginator template
searchPageLayoutTemplatePath	`String`	`/exo:ecm/views/templates/WCM Advance Search/search-page-layout/UISearchPageLayoutDefault.gtmp`	The path to the search page template
itemsPerPage	`Integer`	5	The number of items for each page
showQuickEditButton	`Boolean`	`true`	Show or hide the quick edit icon (!)
basePath	`String`	`parameterizedviewer`	The page which is used to display the search result

2.3.2. Sample configuration

```
<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value>collaboration</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>searchFormTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-form/UIDefaultSearchForm.gtmpl</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>searchResultTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-result/UIDefaultSearchResult.gtmpl</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>searchPaginatorTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-paginator/UIDefaultSearchPaginator.gtmpl</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>searchPageLayoutTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-page-layout/UISearchPageLayoutDefault.gtmpl</value
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>itemsPerPage</name>
    <value>5</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showQuickEditButton</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>basePath</name>
    <value>parameterizedviewer</value>
    <read-only>false</read-only>
```

```
  </preference>
</portlet-preferences>
```

2.4. Explorer

2.4.1. Available preferences

Preference	Type	Default value	Description
repository	`String`	`repository`	The repository name which will be used in an instance of Site Explorer
workspace	`String`	N/A	Not in use. The workspace name was included in the Drive(!)
path	`String`	N/A	The path of node. This preference will be used when the usecase chosen is Parameterize
drive	`String`	N/A	Not in use. Replaced by driveName preference(!)
views	`String`	N/A	Not in use. The views will be displayed basing on the Drive which user has the access permission(!)
allowCreateFolders	`String`	N/A	Allow to create a folder by type. When you do not specify the value, then the default value will be nt:unstructured, nt:folder
categoryMandatoryWhenFileUpload	`Boolean`	false	Force an user to add a category when uploading or creating a document
uploadFileSizeLimitMB	`Float`	150	The maximum of file size when upload to the system (MB)
usecase	`String`	`selection`	The behavior to access Site Explorer. By default, the "selection" option is configured. Besides "selection", there are 4 other ways to configure the file explorer: **Jailed**,**Personal**,**Social**,**Parameterize**

Preference	Type	Default value	Description
driveName	`String`	`Private`	The name of a drive that an user wants to access
trashHomeNodePath	`String`	`/Trash`	The location to store the deleted nodes
trashRepository	`String`	`repository`	The name of the repository where stores the deleted nodes
trashWorkspace	`String`	`collaboration`	The name of the workspace where stores the deleted nodes
editInNewWindow	`Boolean`	`false`	Allow to edit document with or without a window popup.
showTopBar	`Boolean`	`true`	Allow to show the Top bar or not.
showActionBar	`Boolean`	`true`	Allow to show the Action bar or not.
showSideBar	`Boolean`	`true`	Allow to show the Side bar or not.
showFilterBar	`Boolean`	`true`	Allow to show the Filter bar or not.

2.4.2. Example of configuration

```
<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value/>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>path</name>
    <value/>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>drive</name>
    <value/>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>views</name>
    <value/>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>allowCreateFolders</name>
    <value/>
    <read-only>false</read-only>
```

```
  </preference>
  <preference>
    <name>categoryMandatoryWhenFileUpload</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>uploadFileSizeLimitMB</name>
    <value>150</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>usecase</name>
    <value>selection</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>driveName</name>
    <value>Private</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>trashHomeNodePath</name>
    <value>/Trash</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>trashRepository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>trashWorkspace</name>
    <value>collaboration</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>editInNewWindow</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showTopBar</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showActionBar</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showSideBar</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showFilterBar</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
</portlet-preferences>
```

2.5. Administration

2.5.1. Available preferences

Preference	Type	Default	Description
repository	`String`	`Repository`	The name of the current

Preference	Type	Default	Description
			repository

2.5.2. Sample Configuration

```
<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
</portlet-preferences>
```

2.6. Fast Content Creator

2.6.1. Available preferences

Preference	Type	Default Value	Description
mode	`String`	`basic`	The default mode for **fast content creator** portlet
repository	`String`	`repository`	The name of the current repository
workspace	`String`	`collaboration`	The workspace where the content is stored
path	`String`	`/Groups/platform/users/Documents`	The destination path where the content is stored
type	`String`	`exo:article`	The node type of document which will be shown on the dialog form
saveButton	`String`	`Save`	The custom button Save
saveMessage	`String`	`This node has been saved successfully`	The custom message when clicking **Save** button
isRedirect	`Boolean`	`false`	Redirect to another page or not
redirectPath	`String`	`http://www.google.com.vn/`	The path that the page will redirect to
isActionNeeded	`Boolean`	`true`	Is an action needed to save the configuration

2.6.2. Example Configuration

```
<portlet-preferences>
  <preference>
    <name>mode</name>
    <value>basic</value>
    <read-only>true</read-only>
  </preference>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value>collaboration</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>path</name>
    <value>/Groups/platform/users/Documents</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>type</name>
    <value>exo:article</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>saveButton</name>
    <value>Save</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>saveMessage</name>
    <value>This node has been saved successfully</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>isRedirect</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>redirectPath</name>
    <value>http://www.google.com.vn</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>isActionNeeded</name>
    <value>true</value>
    <read-only>true</read-only>
  </preference>
</portlet-preferences>
```

Chapter 3. Configuration

3.1. Drives

A drive is like a shortcut in the content repository, a quick access to some places for users. You can restrict the visibility of this drive to a group/user and apply a specific view depending on the content you have in this area.

A drive is the combination of:

- Path: the root folder of the drive.
- View: how we can see the contents (by list, thumbnails, coverflow, etc).
- Role: visible to everybody, a group or a single user.
- Options: allow us to specify whether to see hidden nodes or not and to create folders in this drive or not.

3.1.1. Fields

Object type: *org.exoplatform.services.cms.drives.**DriveData***

Field name	Type	Description
name	`String`	Name of the drive. It must be unique
repository	`String`	Content Repository where to find the root path
workspace	`String`	Workspace in the Content Repository
homePath	`String`	Root path in the Content Repository. **`userId`** can be used to use the userId at runtime in the path.
permissions	`String`	Visibility of the drive based on eXo rights. For example: `*:/platform/users`
icon	`String`	Url to the icon
views	`String`	The list of views you want to use, separated by commas. For example: `simple-view,admin-view`
viewPreferences	`Boolean`	**User Preference** icon will be visible if true
viewNonDocument	`Boolean`	Non-document types will be visible in the user view if true

Field name	Type	Description
viewSideBar	`Boolean`	Show/Hide the left bar (with navigation and filters)
showHiddenNode	`Boolean`	Hidden nodes will be visible if true
allowCreateFolders	`Boolean`	List of nodet ype we can create as folders. For example: `nt:folder,nt:unstructured`
allowNodeTypesOnTree	`String`	Allows you to filter node types in the navigation tree. For example, the default value is "*" to show all content types

We use the following structure for drives configuration

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.drives.ManageDriveService</target-component>
  <component-plugin>
    <name>manage.drive.plugin</name>
    <set-method>setManageDrivePlugin</set-method>
    <type>org.exoplatform.services.cms.drives.impl.ManageDrivePlugin</type>
    <description>Nothing</description>
    <init-params>
      <object-param>There are initializing attributes of org.exoplatform.services.cms.drives.DriveData object</d
    </init-params>
  </component-plugin>
</external-component-plugins>
```

The file that contains the structure above will be configured in the configuration.xml file as the following:

```
<import>war:/conf/wcm-extension/dms/drives-configuration.xml</import>
```

3.1.2. Example of configuration

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.drives.ManageDriveService</target-component>
  <component-plugin>
    <name>manage.drive.plugin</name>
    <set-method>setManageDrivePlugin</set-method>
    <type>org.exoplatform.services.cms.drives.impl.ManageDrivePlugin</type>
    <description>Nothing</description>
    <init-params>
      <object-param>
        <name>Managed Sites</name>
        <description>Managed Sites</description>
        <object type="org.exoplatform.services.cms.drives.DriveData">
          <field name="name">
            <string>Managed Sites</string>
          </field>
          <field name="repository">
            <string>repository</string>
          </field>
          <field name="workspace">
            <string>collaboration</string>
          </field>
          <field name="permissions">
            <string>*:/platform/administrators</string>
          </field>
          <field name="homePath">
            <string>/sites content/live</string>
          </field>
          <field name="icon">
```

```
          <string/>
        </field>
        <field name="views">
          <string>wcm-view</string>
        </field>
        <field name="viewPreferences">
          <boolean>false</boolean>
        </field>
        <field name="viewNonDocument">
          <boolean>true</boolean>
        </field>
        <field name="viewSideBar">
          <boolean>true</boolean>
        </field>
        <field name="showHiddenNode">
          <boolean>false</boolean>
        </field>
        <field name="allowCreateFolders">
          <string>nt:folder,nt:unstructured</string>
        </field>
        <field name="allowNodeTypesOnTree">
          <string>*</string>
        </field>
      </object>
    </object-param>
    <object-param>
      <name>Public</name>
      <description>Public drive</description>
      <object type="org.exoplatform.services.cms.drives.DriveData">
        <field name="name">
          <string>Public</string>
        </field>
        <field name="repository">
          <string>repository</string>
        </field>
        <field name="workspace">
          <string>collaboration</string>
        </field>
        <field name="permissions">
          <string>*:/platform/users</string>
        </field>
        <field name="homePath">
          <string>/Users/${userId}/Public</string>
        </field>
        <field name="icon">
          <string/>
        </field>
        <field name="views">
          <string>simple-view, admin-view</string>
        </field>
        <field name="viewPreferences">
          <boolean>false</boolean>
        </field>
        <field name="viewNonDocument">
          <boolean>false</boolean>
        </field>
        <field name="viewSideBar">
          <boolean>true</boolean>
        </field>
        <field name="showHiddenNode">
          <boolean>false</boolean>
        </field>
        <field name="allowCreateFolders">
          <string>nt:folder,nt:unstructured</string>
        </field>
        <field name="allowNodeTypesOnTree">
          <string>*</string>
        </field>
      </object>
    </object-param>
  </init-params>
 </component-plugin>
</external-component-plugins>
```

3.2. Publication

3.2.1. component configuration

Configuration name	Data type	Sample value	Description
key	`String`	`org.exoplatform.services.wcm.publication.WCMPublicationS`	Service class location
type	`String`	`org.exoplatform.services.wcm.publication.WCMPublicationS`	Service implementation location
component-plugins	`Object`	N/A	The plugins that are used inside the component
init-params	`Object`	N/A	The intitial parameter that will be used inside the component

3.2.1.1. component-plugin configuration

- **Description**:

Configuration name	Data type	Sample value	Description
name	`String`	`Simple publication`	Name of component-plugin
set-method	`String`	`addPublicationPlugin`	Set method to start using plugin
type	`String`	`org.exoplatform.services.wcm.publication.lifecycle.simpl`	Class path location of plugin

3.2.1.2. init-params configuration

- parameter will be set in pair: name-value and will be placed inside the <value-param> object

Configuration name	Data type	Sample value	
name	`String`	`useCache`	
value	`String`	`true`	

- parameter also can be set in an object, with further information will be the object's properties.

Configuration name	Data type	Sample value	Description
name	`String`	`cache.config.wcm.composer`	The name of the object
description	`String`	N/A	The brief description about the object

Configuration name	Data type	Sample value	Description
object	`Object`	N/A	This object will be passed to component as a parameter, this object includes fieldset properties

3.2.2. external-component-plugins configuration

Configuration name	Data type	Sample value	Description
target-component	`String`	`org.exoplatform.services.xml.ConfigService`	External plugin's class location
component-plugin	`Object`	N/A	Configuration of component that will be used as a plugin

3.2.3. Example

3.2.3.1. Component that use plugins example

```
<component>
  <key>org.exoplatform.services.wcm.publication.WCMPublicationService</key>
  <type>org.exoplatform.services.wcm.publication.WCMPublicationServiceImpl</type>
  <component-plugins>
    <component-plugin>
      <name>States and versions based publication</name>
      <set-method>addPublicationPlugin</set-method>
      <type>org.exoplatform.services.wcm.publication.lifecycle.stageversion.StageAndVersionPublicationPlugin</ty
      <description>This publication lifecycle publish a web content or DMS document to a portal page with more s
    </component-plugin>
    <component-plugin>
      <name>Simple publication</name>
      <set-method>addPublicationPlugin</set-method>
      <type>org.exoplatform.services.wcm.publication.lifecycle.simple.SimplePublicationPlugin</type>
      <description>This publication lifecycle publish a web content or DMS document to a portal page without ver
    </component-plugin>
  </component-plugins>
</component>
```

3.2.3.2. Component that has init-params example

```
<component>
  <key>org.exoplatform.services.wcm.publication.WCMComposer</key>
  <type>org.exoplatform.services.wcm.publication.WCMComposerImpl</type>
  <init-params>
    <value-param>
      <name>useCache</name>
      <value>true</value>
    </value-param>
  </init-params>
</component>
```

3.2.3.3. external-component-plugins configuration example

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cache.CacheService</target-component>
  <component-plugin>
    <name>addExoCacheConfig</name>
    <set-method>addExoCacheConfig</set-method>
    <type>org.exoplatform.services.cache.ExoCacheConfigPlugin</type>
    <description>Configures the cache for query service</description>
    <init-params>
      <object-param>
        <name>cache.config.wcm.composer</name>
        <description>The default cache configuration</description>
        <object type="org.exoplatform.services.cache.ExoCacheConfig">
          <field name="name">
            <string>wcm.composer</string>
          </field>
          <field name="maxSize">
            <int>300</int>
          </field>
          <field name="liveTime">
            <long>600</long>
          </field>
          <field name="distributed">
            <boolean>false</boolean>
          </field>
          <field name="implementation">
            <string>org.exoplatform.services.cache.concurrent.ConcurrentFIFOExoCache</string>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

3.3. Deployment

When a site is created, normally the end-user wants to see something in the page instead of a blank page, so we need this service for deployment of some "default" contents like Banner, Footer, Navigation, Breadcrumb, etc.

There are two main cases to use:

- The site is created only one time when the database is clean.
- The site is created at runtime, when a user uses the core feature of GateIn portal.

3.3.1. Fields

`init-params`

Element	Type	Description
object-param	`Object`	The parameter which is an Object

`object-param`

Element	Type	Default value	object-param
name	`String`	`ACME Logo data`	The name of this object parameter
description	`String`	`Deployment Descriptor`	The description of this

Element	Type	Default value	object-param
			object parameter
object	`Class`	`org.exoplatform.services.deployment.DeploymentDescriptor`	The object of this object parameter

`object`

Attribute	Type	Default value	Description
type	`String`	`org.exoplatform.services.deployment.DeploymentDescriptor` `(*)`	The type of this object

org.exoplatform.services.deployment.DeploymentDescriptor

Name	Type	Default value	Description
target	`Object`	`org.exoplatform.services.deployment.DeploymentDescriptor` `(*)`	The target node (which will contains the imported node)
sourcePath	`String`	`war:/conf/sample-portal/acme/acme-resources/acm`	The absolute path of the XML file
cleanupPublication	`Boolean`	`false`	Decide when allows to cleanup the publication lifecycle in the target folder after importing the data. true: allow false:not allow

`org.exoplatform.services.deployment.DeploymentDescriptor$Target`

Field	Type	Default value	Description
repository	`String`	`repository`	The repository of the target node
workspace	`String`	`collaboration`	The workspace of the target node
nodePath	`String`	`/sites content/live/acme/web contents/site artifacts`	The path of the target node

3.3.2. Example of configuration

```
<external-component-plugins>
  <target-component>org.exoplatform.services.deployment.WCMContentInitializerService</target-component>
  <component-plugin>
    <name>Content Initializer Service</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.deployment.plugins.XMLDeploymentPlugin</type>
    <description>XML Deployment Plugin</description>
    <init-params>
      <object-param>
        <name>ACME Logo data</name>
        <description>Deployment Descriptor</description>
        <object type="org.exoplatform.services.deployment.DeploymentDescriptor">
          <field name="target">
            <object type="org.exoplatform.services.deployment.DeploymentDescriptor$Target">
              <field name="repository">
                <string>repository</string>
              </field>
              <field name="workspace">
                <string>collaboration</string>
              </field>
              <field name="nodePath">
                <string>/sites content/live/acme/web contents/site artifacts</string>
              </field>
            </object>
          </field>
          <field name="sourcePath">
            <string>war:/conf/sample-portal/wcm/artifacts/site-resources/acme/Logo.xml</string>
          </field>
          <field name="cleanupPublication">
            <boolean>true</boolean>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

3.4. Taxonomies

Taxonomies are used to sort documents to ease searches when browsing documents online. The main idea behind that concept is to provide a multi dimensional set of paths to find a document. The best example is when you browse Yahoo news. In many cases, you can get your content by using different category paths. Therefore, after creating a document somewhere in the repository, it is possible to categorize it by adding several taxonomy references. By browsing the taxonomy tree, it will be possible to find the referencing article and display them as if they were children of the taxonomy nodes. As you can imagine, taxonomies are stored in the JCR itself and we are using the JCR Reference functionality to provide that advanced ECM feature.

Managing the tree of taxonomies is very simple, you can copy/cut nodes and paste them. Of course, you can add and remove taxonomies from the tree. Once a taxonomy has been added, any user who has access to the "Manage Categories" icon from his/her view can then browse the taxonomy tree and refer one of its nodes to the created documents.

3.4.1. Fields

Object type: *org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission*

Field name	Collection type	Description
permissions	`java.util.ArrayList<TaxonomyTreeDefaultUserPermission>`	List of default user permissions

Field name	Collection type	Description
		access the taxonomy tree

Object type: *org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission$Permission*

Field name	Type	Default value	Description
identity	`String`	`*:/platform/administrators`	user name or user group.
read	`Boolean`	`true`	permission to read the taxonomy tree.
addNode	`Boolean`	`true`	permission to add a node in the taxonomy tree.
setProperty	`Boolean`	`true`	permission to set properties for a node in the taxonomy tree .
remove	`Boolean`	`true`	permission to remove a node from the taxonomy tree.

Target Component: `org.exoplatform.services.cms.taxonomy.TaxonomyService`

Name: `predefinedTaxonomyPlugin`

Set-method: `addTaxonomyPlugin`

Type: *org.exoplatform.services.cms.taxonomy.impl.TaxonomyPlugin*

value param	Type	Default value	Description
autoCreateInNewRepository	`Boolean`	`true`	enable/ Disable the creation of the taxonomies in new created repository.
repository	`String`	`repository`	name of the repository where taxonomies are created.
workspace	`String`	`dms-system`	name of the workspace where taxonomies are created.
treeName	`String`	`system`	name of the taxonomy tree to be created.

Object param	Type	Default value	Description
permission.configuration	`String`	`org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfi`	access permission for the whole taxonomy tree.

Object param	Type	Default value	Description
predefined.actions	`string`	`org.exoplatform.services.cms.actions.impl.ActionConfig`	predefined actions for taxonomy tree root node.
taxonomy.configuration	`String`	`org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfi`	access permission for each taxonomy in tree.

Object type: *org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig*

Name	Type	Description
taxonomies	`java.util.ArrayList<TaxonomyConfig$Taxonomy>`	list of taxonomy to be configured permission.

Object type: *org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig$Taxonomy*

Name	Type	Default value	Description
name	`String`	`name`	name of the taxonomy.
path	`String`	`cms`	path to the taxonomy in taxonomy tree.
permissions	`Arraylist`	`java.util.ArrayList<TaxonomyConfig$Permission>`	list of permission for user or group to access the taxonomy.

Object type: *org.exoplatform.services.cms.actions.impl.ActionConfig*

Name	Type	Default value	Description
actions	`Arraylist`	`java.util.ArrayList<ActionConfig$TaxonomyAction>`	list of actions which are created for the taxonomy tree root node.

Object type: *org.exoplatform.services.cms.actions.impl.ActionConfig$TaxonomyAction*

Field name	Type	Default value	Description
type	`String`	`exo:taxonomyAction`	type of the action.
name	`String`	`taxonomyAction`	name of the action.
description	`String`	`N/A`	description of the action.
homePath	`String`	`dms-system:/exo:ecm/exo:actionStorage/System`	location of node where the action node is stored.
targetWspace	`String`	`N/A`	when a new node is created in the node whose path is defined in **homePath** field, it will be

Field name	Type	Default value	Description
			moved to a new location automatically. The target workspace is specified in this field.
targetPath	`String`	`collaboration`	when a new node is created in the node whose path is defined in **homePath** field, it will be moved to a new location automatically. The target path is specified in this field.
lifecyclePhase	`Arraylist`	`java.util.ArrayList<String>`	the life cycle phases, in which the action is activated.
roles	`String`	*:/platform/administrators	the users or groups only with whom the action is activated.
mixins	`Arraylist`	`java.util.ArrayList<ActionConfig$Mixin>`	list of mixins that are affected by this action.

Object type: *org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin*

Field name	Type	Default value	Description
name	`String`	`mix:affectedNodeTypes`	mixin node type name that will be added, whose responsibility is to store the affected node types.
properties	`String`	`exo:affectedNodeTypeName`	the name of the properties that store all the node types affected by the action.

3.4.2. Sample Configuration

```
Exception occured, see logs
```

3.5. Nodetypes

Every node has a type. Node types can be used to define complex storage objects consisting of multiple sub-nodes and properties, possibly many deep layers.

3.5.1. Fields

`org.exoplatform.services.jcr.core.nodetype.ExtendedNodeTypeManager`

Node Type	Type	Description
name	`String`	Name of the node type. This field is required
isMixin	`boolean`	mixin type
hasOrderableChildNodes	`boolean`	Orderable childnodes
primaryItemName	`String`	Primary item name
suppertype	`String`	Name of supper type
propertyDefinition	`Object`	Define properties of node type

3.5.2. Examples Nodetype configuration

The node definition should be placed in a special XML file (see DTD below) and declared in the service's configuration file to eXo component plugin mechanism, described as follows:

```
       <external-component-plugins>
         <target-component>org.exoplatform.services.jcr.RepositoryService</target-component>
               <component-plugin>
           <name>add.namespaces</name>
           <set-method>addPlugin</set-method>
           <type>org.exoplatform.services.jcr.impl.AddNamespacesPlugin</type>
           <init-params>
             <properties-param>
               <name>namespaces</name>
               <property name="publication" value="http://www.exoplatform.com/jcr/publication/1.1/"/>
             </properties-param>
           </init-params>
         </component-plugin>
         <component-plugin>
           <name>add.nodeType</name>
           <set-method>addPlugin</set-method>
           <type>org.exoplatform.services.jcr.impl.AddNodeTypePlugin</type>
                       <priority>99</priority>
           <init-params>
             <values-param>
               <name>autoCreatedInNewRepository</name>
               <description>Node types configuration file</description>
               <value>jar:/conf/nodetypes-publication-config.xml</value>
             </values-param>
           </init-params>
         </component-plugin>
       </external-component-plugins>
```

This is a Nodetype definition file format:

```
<nodeTypes xmlns:nt="http://www.jcp.org/jcr/nt/1.0" xmlns:mix="http://www.jcp.org/jcr/mix/1.0"
  xmlns:jcr="http://www.jcp.org/jcr/1.0">
  <nodeType name="publication:publication" isMixin="true" hasOrderableChildNodes="false" primaryItemName="">
    <propertyDefinitions>
            <propertyDefinition name="publication:lifecycleName" requiredType="String" autoCreated="false" manda
       onParentVersion="COPY" protected="false" multiple="false">
       <valueConstraints/>
     </propertyDefinition>
     <propertyDefinition name="publication:currentState" requiredType="String" autoCreated="false" mandatory="t
       onParentVersion="COPY" protected="false" multiple="false">
```

```
        <valueConstraints/>
      </propertyDefinition>
      <propertyDefinition name="publication:history" requiredType="String" autoCreated="false" mandatory="true"
        onParentVersion="COPY" protected="false" multiple="true">
        <valueConstraints/>
      </propertyDefinition>
    </propertyDefinitions>
  </nodeType>
</nodeTypes>
```

3.6. Views

A view can include many object parameters. Parameters are used to create default views, templates and actions of **Manage View** service. View enables administrators to customize view classification that can impact on users in exploring workspace. Each object-param has a type that is a class representing all properties of a view.

3.6.1. Fields

- **target-component**: *org.exoplatform.services.cms.views.ManageViewService</target-component-*
- **Name**: `manage.view.plugin`
- **Set-method**: `setManageViewPlugin`
- **Type**: `org.exoplatform.services.cms.views.impl.ManageViewPlugin`
- **Description**: This plugin manages user views

Object type: *org.exoplatform.services.cms.views.ViewConfig*

Name	Type	Default value	Description
name	`String`	`system-view`	The name of the view. It must be unique inside WCM.
permissions	`String`	*:/platform/administrators	Visibility of the view based on eXo rights.
template	`String`	`/exo:ecm/views/templates/explorer/system-view`	Specify path to the template location.
tabList	`ArrayList`	`java.util.ArrayList`	Include a set of view names.

Object type: *org.exoplatform.services.cms.views.ViewConfig$Tab*

Name	Type	Default value	Description
tabName	`String`	`Info`	The name of the tab. It must be unique.
button	`String`	`viewNodeType;`	Specify a set of view

Name	Type	Default value	Description
		`viewPermissions; viewProperties; showJCRStructure`	component names.

Object type: *org.exoplatform.services.cms.views.TemplateConfig*

Name	Type	Default value	Description
type	`String`	`ecmExplorerTemplate`	Specify if a name is truly a class representing all properties of a view.
name	`String`	`system-view`	Specify a set of view component names.
warPath	`String`	`/ecm-explorer/SystemView.gtmpl`	Specify a template location to view.

3.6.2. Example of configuration

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.views.ManageViewService</target-component>
  <component-plugin>
    <name>manage.view.plugin</name>
    <set-method>setManageViewPlugin</set-method>
    <type>org.exoplatform.services.cms.views.impl.ManageViewPlugin</type>
    <description>this plugin manage user view</description>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>predefinedViewsLocation</name>
        <value>war:/conf/dms-extension/dms/artifacts</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <object-param>
        <name>System-View</name>
        <description>View configuration for System workspace</description>
        <object type="org.exoplatform.services.cms.views.ViewConfig">
          <field name="name">
            <string>system-view</string>
          </field>
          <field name="permissions">
            <string>*:/platform/administrators</string>
          </field>
          <field name="template">
            <string>/exo:ecm/views/templates/ecm-explorer/SystemView</string>
          </field>
          <field name="tabList">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.views.ViewConfig$Tab">
                  <field name="tabName">
                    <string>Info</string>
                  </field>
                  <field name="buttons">
                    <string>viewNodeType; viewPermissions; viewProperties; showJCRStructure</string>
                  </field>
                </object>
```

```
            </value>
          </collection>
        </field>
      </object>
    </object-param>
    <object-param>
      <name>System Template</name>
      <description>Template for display documents in list style</description>
      <object type="org.exoplatform.services.cms.views.TemplateConfig">
        <field name="type">
          <string>ecmExplorerTemplate</string>
        </field>
        <field name="name">
          <string>SystemView</string>
        </field>
        <field name="warPath">
          <string>/ecm-explorer/SystemView.gtmpl</string>
        </field>
      </object>
    </object-param>
  </init-params>
 </component-plugin>
</external-component-plugins>
```

3.7. Templates

3.7.1. Application template

A template is a presentation while displaying the saved information.

Template service is used to select the right template corresponding to the saved content.

3.7.1.1. Available configuration

3.7.1.1.1. Init-params

Configuration name	Data type	Default Value	Description
autoCreateInNewRepository	`Boolean`	`true`	Enable the application to import predefined templates at the start-up of template service automatically.
storedLocation	`String`	`war:/conf/dms-extension/jcrtemplates`	Location of stored templates
repository	`String`	`repository`	Location of stored templates
object-param	`Structure`		Configuration for each instance.

3.7.1.1.2. Object-param

Configuration name	Data type	Default Value	Description
name	`String`	`template.configuration`	The name of this

Configuration name	Data type	Default Value	Description
			configuration
description	`String`	`configuration for the location of templates to inject in jcr`	Description of template instance
object-type	`Structure`		Details configuration for each instance type

Object-type defines all available template files, using the "collection type" configuration

3.7.1.1.3. Object

type: It is the name of each object type. It means the type of template, the further configurations for this type are defined by some specified fields.

Field name	Data type	Description
nodetypeName	`String`	The name of template that is saved as a node in system
documentTemplate	`Boolean`	Determine if the node type is a document type
label	`String`	Visual display of the title for this node

There are three further information related to the presentation of each template:

- **`referencedView`** : Determine how to display a view.
- **`referencedDialog`** : Determine how to display a dialog to input information.
- **`referencedSkin`** : Determine the stylesheet for displaying.

Each type includes some definitions as an ?`object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template"` with the fields as the properties.

Field name	Data type	Description
templateFile	`String`	The location of the file store for the template's presentation
roles	`String`	Determine who can access this object (View/Dialog/CSS)

3.7.1.2. Example of template configuration

This below example is configuration for the *nt:file_ template, any other template will be put in the same level with this template start from the line* `<object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$NodeType">` *as the another NodeType.*

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.templates.TemplateService</target-component>
  <component-plugin>
    <name>addTemplates</name>
    <set-method>addTemplates</set-method>
    <type>org.exoplatform.services.cms.templates.impl.TemplatePlugin</type>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>storedLocation</name>
        <value>war:/conf/dms-extension/dms/artifacts/templates</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <object-param>
        <name>template.configuration</name>
        <description>configuration for the localtion of templates to inject in jcr</description>
        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig">
          <field name="nodeTypes">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$NodeType">
                  <field name="nodetypeName">
                    <string>nt:file</string>
                  </field>
                  <field name="documentTemplate">
                    <boolean>true</boolean>
                  </field>
                  <field name="label">
                    <string>File</string>
                  </field>
                  <field name="referencedView">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                          <field name="templateFile">
                            <string>/file/views/view1.gtmpl</string>
                          </field>
                          <field name="roles">
                            <string>*</string>
                          </field>
                        </object>
                      </value>
                      <value>
                        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                          <field name="templateFile">
                            <string>/file/views/admin_view.gtmpl</string>
                          </field>
                          <field name="roles">
                            <string>*:/platform/administrators</string>
                          </field>
                        </object>
                      </value>
                    </collection>
                  </field>
                  <field name="referencedDialog">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                          <field name="templateFile">
                            <string>/file/dialogs/dialog1.gtmpl</string>
                          </field>
                          <field name="roles">
                            <string>*</string>
                          </field>
                        </object>
```

```
                    </value>
                    <value>
                      <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                        <field name="templateFile">
                          <string>/file/dialogs/admin_dialog.gtmpl</string>
                        </field>
                        <field name="roles">
                          <string>*:/platform/administrators</string>
                        </field>
                      </object>
                    </value>
                  </collection>
                </field>
                <field name="referencedSkin">
                  <collection type="java.util.ArrayList">
                    <value>
                      <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                        <field name="templateFile">
                          <string>/file/skins/Stylesheet-lt.css</string>
                        </field>
                        <field name="roles">
                          <string>*</string>
                        </field>
                      </object>
                    </value>
                    <value>
                      <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                        <field name="templateFile">
                          <string>/file/skins/Stylesheet-rt.css</string>
                        </field>
                        <field name="roles">
                          <string>*</string>
                        </field>
                      </object>
                    </value>
                  </collection>
                </field>
              </object>
            </value>
          </collection>
        </field>
      </object>
    </object-param>
  </init-params>
 </component-plugin>
</external-component-plugins>
```

3.7.2. Nodetype template

3.8. Views

A view can include many object parameters. Parameters are used to create default views, templates and actions of **Manage View** service. View enables administrators to customize view classification that can impact on users in exploring workspace. Each object-param has a type that is a class representing all properties of a view.

3.8.1. Fields

- **target-component**: *org.exoplatform.services.cms.views.ManageViewService</target-component-*
- **Name**: `manage.view.plugin`
- **Set-method**: `setManageViewPlugin`

- **Type**: `org.exoplatform.services.cms.views.impl.ManageViewPlugin`
- **Description**: This plugin manages user views

Object type: *org.exoplatform.services.cms.views.ViewConfig*

Name	Type	Default value	Description
name	`String`	`system-view`	The name of the view. It must be unique inside WCM.
permissions	`String`	*:/platform/administrators	Visibility of the view based on eXo rights.
template	`String`	`/exo:ecm/views/templates/ecm-explorer/SystemView`	Specify path to the template location.
tabList	`ArrayList`	`java.util.ArrayList`	Include a set of view names.

Object type: *org.exoplatform.services.cms.views.ViewConfig$Tab*

Name	Type	Default value	Description
tabName	`String`	`Info`	The name of the tab. It must be unique.
button	`String`	`viewNodeType; viewPermissions; viewProperties; showJCRStructure`	Specify a set of view component names.

Object type: *org.exoplatform.services.cms.views.TemplateConfig*

Name	Type	Default value	Description
type	`String`	`ecmExplorerTemplate`	Specify if a name is truly a class representing all properties of a view.
name	`String`	`system-view`	Specify a set of view component names.
warPath	`String`	`/ecm-explorer/SystemView.groovy`	Specify a template location to view.

3.8.2. Example of configuration

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.views.ManageViewService</target-component>
  <component-plugin>
    <name>manage.view.plugin</name>
    <set-method>setManageViewPlugin</set-method>
```

```
    <type>org.exoplatform.services.cms.views.impl.ManageViewPlugin</type>
    <description>this plugin manage user view</description>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>predefinedViewsLocation</name>
        <value>war:/conf/dms-extension/dms/artifacts</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <object-param>
        <name>System-View</name>
        <description>View configuration for System workspace</description>
        <object type="org.exoplatform.services.cms.views.ViewConfig">
          <field name="name">
            <string>system-view</string>
          </field>
          <field name="permissions">
            <string>*:/platform/administrators</string>
          </field>
          <field name="template">
            <string>/exo:ecm/views/templates/ecm-explorer/SystemView</string>
          </field>
          <field name="tabList">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.views.ViewConfig$Tab">
                  <field name="tabName">
                    <string>Info</string>
                  </field>
                  <field name="buttons">
                    <string>viewNodeType; viewPermissions; viewProperties; showJCRStructure</string>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
      <object-param>
        <name>System Template</name>
        <description>Template for display documents in list style</description>
        <object type="org.exoplatform.services.cms.views.TemplateConfig">
          <field name="type">
            <string>ecmExplorerTemplate</string>
          </field>
          <field name="name">
            <string>SystemView</string>
          </field>
          <field name="warPath">
            <string>/ecm-explorer/SystemView.gtmpl</string>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

Chapter 4. Inside WCM Templates

4.1. Content types

Overview

The templates are applied to a NodeType or a metadata MixinType. Two kinds of templates exist :

- **dialogs** : are html forms that allow to create node instances.
- **views** : are html fragments used to display nodes.

From the ECM admin portlet, *Manage Template* lists existing NodeTypes that have been associated to Dialog and/or View templates. These templates can be attached to permissions (in the usual *membership:group_ form), so that a specific one is displayed according to the rights of the user (very useful in a content validation workflow activity).*

DocumentType

1. checkbox allows to say if the NodeType should be considered as a **DocumentType**. File Explorer considers such nodes as user content and applies the following behavior :

- View template will be used to display the DocumentType nodes
- DocumentTypes nodes can created by the '*Add Document*' action
- non DocumentType are hidden (unless *'Show non document types'* option is checked)

Templates are written using Groovy Templates and will require some experience with JCR API and HTML notions.

4.1.1. Dialog

Dialogs are groovy templates that generate forms by mixing static HTML fragments and groovy calls to the components responsible for building the UI at runtime. The result is a simple but powerful syntax.

4.1.1.1. Common parameters

These following parameters are common and can be used for every input fields.

Parameter	Type	Required	Description	Example
jcrPath	`String`		relative path inside the current node	`jcrPath=/node/exo:title`
mixintype	`String` with comma `,` character		List of the mixin types you want to initialize when	`mixintype=mix:i18n`

Parameter	Type	Required	Description	Example
			creating the content	`mixintype=mix:votable,mix:co`

validate	`String` with comma `,` character		List of the validators you want to apply to the input. Possible values are : `name`, `email`, `number`, `empty`, `null`, `datetime`, `length` OR validator classes	`validate=empty`
				`validate=empty,name`
				`validate=org.exoplatform.web`

editable	`String`		The input will be editable only if the value of this parameter is `if-null` and the value of this input is null or blank	`editable=if-null`
multiValues	`Boolean`		Show a multi values component if true. Must be used only with corresponding multi-values properties. The default value of this parameter is `false`	`multiValues=true`
visible	`Boolean`		The input is visible if this value is `true`	`visible=true`

See also:

- Text field
- Hidden field
- Text area field
- Rich text field
- Calendar field
- Upload field
- Radio field

- Select box field
- Checkbox field
- Mixin field
- Action field

Warning

Note that `mixintype` can be used only in the root node field (commonly known as the name field)

4.1.1.2. Text Field

4.1.1.2.1. Addition parameters

None

See also: Common parameters

4.1.1.2.2. Example

```
<%
  String[] fieldTitle = ["jcrPath=/node/exo:title", "validate=empty"] ;
  uicomponent.addTextField("title", fieldTitle) ;
%>
```

4.1.1.3. Hidden Field

4.1.1.3.1. Addition parameters

None

See also: Common parameters

4.1.1.3.2. Example

4.1.1.4. Text Area Field

4.1.1.4.1. Addition parameters

Parameter	Type	Required	Description	Example
rows	`Number`		The initial textarea's number of rows. The value is `10` by default	`rows=20`
cols	`Number`		The initial textarea's number of cols. The value is `30` by default	`cols=50`

See also: Common parameters

4.1.1.4.2. Example

```
<%
        String[] fieldDescription = ["jcrPath=/node/exo:description", "validate=empty"] ;
        uicomponent.addTextAreaField("description", fieldDescription)
%>
```

4.1.1.5. Rich Text Field

4.1.1.5.1. Addition parameters

Parameter	Type	Required	Description	Example
options	`String` with semicolon `;` character		Some options for CKEditor field: toolbar, width and height	`options=CompleteWCM;width:'1`

Parameter	Type	Required	Description	Example
toolbar	`String`		The pre-define toolbar for CKEditor. The value can be: `Default`, `Basic`, `CompleteWCM`, `BasicWCM`, `SuperBasicWCM`	`options=CompleteWCM`
width	`String`		The width of CKEditor. The value can be the percent of pixel	`options=width:'100%'`
height	`String`		The height of CKEditor. The value can be the percent of pixel	`options=height:'200px'`

See also: Common parameters

4.1.1.5.2. Example

```
<%
        String[] fieldSummary = ["jcrPath=/node/exo:summary", "options=toolbar:CompleteWCM,width:'100%',height:'
        uicomponent.addRichtextField("summary", fieldSummary) ;
%>
```

4.1.1.6. Calendar Field

4.1.1.6.1. Addition parameters

Parameter	Type	Required	Description	Example
options	`String`		An option for calendar field: Display time	`options=displaytime`

See also: Common parameters

4.1.1.6.2. Example

```
<%
	String[] fieldPublishedDate = ["jcrPath=/node/exo:publishedDate", "options=displaytime", "validate=datet
	uicomponent.addCalendarField("publishedDate", fieldPublishedDate) ;
%>
```

4.1.1.7. Upload Field

4.1.1.7.1. Addition parameters

None

See also: Common parameters

4.1.1.7.2. Example

There are 2 main cases when create an upload form: You want to store the image as a property OR as a node.

- If you want to use property so you can do as follow:

```
<%
	String[] fieldMedia = ["jcrPath=/node/exo:image"] ;
	uicomponent.addUploadField("media", fieldMedia) ;
%>
```

- If you want to use node so you can do as follow:

```
<%
	String[] hiddenField1 = ["jcrPath=/node/exo:image", "nodetype=nt:resource", "visible=false"] ;
	String[] hiddenField2 = ["jcrPath=/node/exo:image/jcr:encoding", "visible=false", "UTF-8"] ;
	String[] hiddenField3 = ["jcrPath=/node/exo:image/jcr:lastModified", "visible=false"] ;
	uicomponent.addHiddenField("hiddenInput1", hiddenField1) ;
	uicomponent.addHiddenField("hiddenInput2", hiddenField2) ;
	uicomponent.addHiddenField("hiddenInput3", hiddenField3) ;

	String[] fieldMedia = ["jcrPath=/node/exo:image"] ;
	uicomponent.addUploadField("media", fieldMedia) ;
%>
```

- But, this code is not complete. If you want to display the **upload** field, the image must be blank, otherwise you can display the image and an action enables you to remove it. You can do as follows:

```
<%
	def image = "image";
	// If you're trying to edit the document
	if(uicomponent.isEditing()) {
		def curNode = uicomponent.getNode();
		// If the image existed
		if (curNode.hasNode("exo:image")) {
			def imageNode = curNode.getNode("exo:image") ;
			// If the image existed and available
			if (imageNode.getProperty("jcr:data").getStream().available() > 0 && (uicomponent.findCo
				def imgSrc = uicomponent.getImage(curNode, "exo:image");
				def actionLink = uicomponent.event("RemoveData", "/exo:image");
				%>
					<div>
						<img src="$imgSrc" width="100px" height="80px"/>
						<a href="$actionLink">
							<img src="/eXoResources/skin/DefaultSkin/background/Blar
						</a>
					</div>
				<%
			} else {
				String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
				uicomponent.addUploadField(image, fieldImage) ;
			}
		} else {
				String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
				uicomponent.addUploadField(image, fieldImage) ;
		}
	}  else if(uicomponent.dataRemoved()) {
		String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
		uicomponent.addUploadField(image, fieldImage) ;
	} else {
		String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
		uicomponent.addUploadField(image, fieldImage) ;
	}
%>
```

4.1.1.8. Radio Field

4.1.1.8.1. Addition parameters

Parameter	Type	Required	Description	Example
options	`String` with comma `,` characters		Some radio values	`options=radio1,radio2,radio3`

See also: Common parameters

4.1.1.8.2. Example

```
<%
	String[] fieldDeep = ["jcrPath=/node/exo:isDeep", "defaultValues=true", "options=radio1,radio2,radio3"];
	uicomponent.addRadioBoxField("isDeep", fieldDeep);
%>
```

4.1.1.9. SelectBox Field

Parameter	Type	Required	Description	Example
options	`String` with comma `,` characters		Some option values	`options=option1,option2,opti`

Parameter	Type	Required	Description	Example

See also: Common parameters

4.1.1.9.1. Example

```
<%
        String[] fieldDeep = ["jcrPath=/node/exo:isDeep", "defaultValues=true", "options=checkbox1,checkbox2,che
        uicomponent.addCheckBoxField("isDeep", fieldDeep);
%>
```

4.1.1.10. CheckBox Field

4.1.1.10.1. Addition parameters

Parameter	Type	Required	Description	Example
options	`String` with comma `,` characters		Some checkbox values	`options=checkbox1,checkbox2,`

See also: Common parameters

4.1.1.10.2. Example

```
<%
        String[] fieldDeep = ["jcrPath=/node/exo:isDeep", "defaultValues=true", "options=checkbox1,checkbox2,che
        uicomponent.addCheckBoxField("isDeep", fieldDeep);
%>
```

4.1.1.11. Mixin Field

4.1.1.11.1. Addition parameters

None

See also: Common parameters

4.1.1.11.2. Example

```
<%
        String[] fieldId = ["jcrPath=/node", "editable=false", "visible=if-not-null"] ;
        uicomponent.addMixinField("id", fieldId) ;
%>
```

4.1.1.12. Action Field

4.1.1.12.1. Addition parameters

Parameter	Type	Required	Description	Example
selectorClass	`String`		The component to display	`selectorClass=org.exoplatfor`
selectorIcon	`String`		The action icon	`selectorIcon=SelectPath24x24`

Depends on the `selectorClass`, some other parameters can be added For example The component `org.exoplatform.ecm.webui.tree.selectone.UIOneNodePathSelector` need these parameters

Parameter	Type	Required	Description	Example
workspaceField	`String`		The field which allow to select a workspace	`workspaceField=targetWorkspa`

The component `org.exoplatform.ecm.webui.selector.UIPermissionSelector` does not need any special parameters.

See also: Common parameters

4.1.1.12.2. Example

```
<%
        String[] fieldPath = ["jcrPath=/node/exo:targetPath", "selectorClass=org.exoplatform.ecm.webui.tree.sele
        uicomponent.addActionField("targetPath", fieldPath)  ;
%>
```

4.1.1.13. Interceptors

To add an intercepter into a dialog, we can use this method **`uicomponent.addInterceptor(String scriptPath, String type)`**

		Parameters
scriptPath	`String`	The relative path to the script file
type	`String`	The type of intercepter: `prev` or `post`

4.1.1.13.1. Example

```
<%
        uicomponent.addInterceptor("ecm-explorer/interceptor/PreNodeSaveInterceptor.groovy", "prev");
%>
```

4.1.1.14. How to add new ECM template with tabs.

To avoid refreshing the first tab for every action execution, we add a new private function to the template with tabs. In the template, we must insert new piece of code like the following:

```
private String getDisplayTab(String selectedTab) {
        if ((uicomponent.getSelectedTab() == null && selectedTab.equals("mainWebcontent"))
                        || selectedTab.equals(uicomponent.getSelectedTab())) {
                return "display:block";
        }
        return "display:none";
}

private String getSelectedTab(String selectedTab) {
        if (getDisplayTab(selectedTab).equals("display:block")) {
                return "SelectedTab";
        }
        return "NormalTab";
}
```

Changing in every event of onclick is have to be done like this :

```
        <div class="UITab NormalTabStyle">
                <div class="<%=getSelectedTab("mainWebcontent")%>">
                        <div class="LeftTab">
                                <div class="RightTab">
                                        <div class="MiddleTab" onClick="<%=uicomponent.event("ChangeTab", "mainW
                                </div>
                        </div>
                </div>
        </div>

        <div class="UITab NormalTabStyle">
                <div class="<%=getSelectedTab("illustrationWebcontent")%>">
                        <div class="LeftTab">
                                <div class="RightTab">
                                        <div class="MiddleTab" onClick="<%=uicomponent.event("ChangeTab", "illus
                                </div>
                        </div>
                </div>
        </div>

        <div class="UITab NormalTabStyle">
                <div class="<%= getSelectedTab("contentCSSWebcontent")%>">
                        <div class="LeftTab">
                                <div class="RightTab">
                                        <div class="MiddleTab" onClick="<%=uicomponent.event("ChangeTab", "conte
                                </div>
                        </div>
                </div>
        </div>
```

Finally, to display the selected tab, we will add it into the style of UITabContent class.

```
<div class="UITabContent" style="<%=getDisplayTab("mainWebcontent")%>">
```

4.1.2. View

```
<%
        def node = uicomponent.getNode() ;
        def originalNode = uicomponent.getOriginalNode()
%>
```

```
<%=node.getName()%>
```

```
<%if(node.hasProperty("exo:title")) {%>
  <%=node.getProperty("exo:title").getString()%>
<%}%>
```

```
<%
        import java.text.SimpleDateFormat ;
        SimpleDateFormat dateFormat = new SimpleDateFormat() ;
%>
...

<%if(node.hasProperty("exo:date")) {
  dateFormat.applyPattern("MMMMM dd yyyy") ;
%>
  <%=dateFormat.format(node.getProperty("exo:date").getDate().getTime())%>
<%}%>
```

```
<%=_ctx.appRes("Sample.view.label.node-name")%>
```

4.2. List of Contents

4.2.1. Folder based Template

4.2.2. Category tree Template

Chapter 5. Inside WCM Explorer

5.1. CSS

- By using WCM, all the stylesheets of each site can be managed online easily. You do not need to access to the file system to modify and wait for the server restarted. About the structure, each site has its own CSS folder, this folder can contain one or more CSS files. These CSS files have the data, and the priority. If they have the same CSS definition, the bigger priority will be applied. You also can disable some of them to make sure the disabled style sill not to be applied in to the site.

- For example: By default, a WCM demo package has two sites: ACME and Classic. The Classic site has a CSS folder which contains a CSS file called **DefaultStylesheet**. Most of the stylesheets of this site are defined within this stylesheet. Moreover, the ACME site has two CSS files called **BlueStylesheet** and **GreenStylesheet**. The blue one is enabled and the green one is disabled by default. All you need to test is to disable the blue one (by editing it and setting Available equals false) and enable the green one. Now, back to the homepage and you will see the magic.

Note

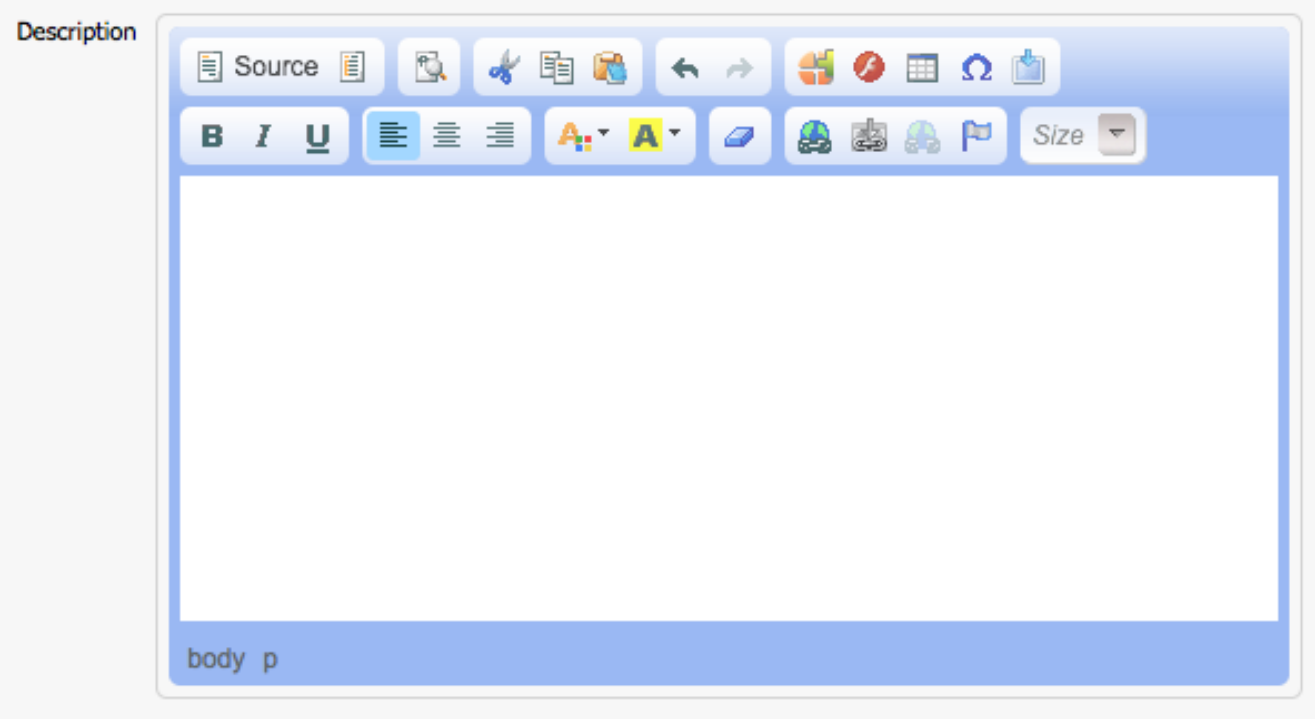
Remember the cache and refresh the browser first if you do not see any changes. Normally, this is the main reason why the new style is not applied.

5.2. Javascript

5.3. CKEditor

Basically, if you want to add a rich text area into your dialogs, you can use addRichtextField method. However, sometimes you want to add the rich text editor manually. So, first, you need to use addTextAreaField method and some additionl javascripts like the below:

```
<%
        String[] fieldDescription = ["jcrPath=/node/exo:description"] ;
        uicomponent.addTextAreaField("description", fieldDescription)
%>
<script>
  var instances = CKEDITOR.instances['description'];
  if (instances) instances.destroy(true);
  CKEDITOR.replace('description', {
    toolbar : 'CompleteWCM',
    uiColor : '#9AB8F3'
  });
</script>
```


Description
Source
Size
body p

Chapter 6. Extensions

6.1. REST Services

6.1.1. REST Overview

REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of "representations" of "resources". A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

At any particular time, a client can either be in transition between application states or "at rest". A client in a REST state is able to interact with its user, but creates no load and consumes no per-client storage on the set of servers or on the network.

The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be in transition. The representation of each application state contains links that may be used the next time, the client chooses to initiate a new state transition.

REST is initially described in the context of HTTP, but is not limited to that protocol. RESTful architectures can be based on other Application Layer protocols if they already provide a rich and uniform vocabulary for applications based on the transfer of meaningful representational state. RESTful applications maximize the use of the pre-existing, well-defined interface and other built-in capabilities provided by the chosen network protocol, and minimize the addition of new application-specific features on its top.

6.1.2. Restful Web Service

6.1.2.1. HTTP Methods

Here is the convention we should follow:

Method	Definition
GET	Get a Resource. It should never modify a state
POST	Create a Resource (or anything that don't fit elsewhere)
PUT	Update a Resource
DELETE	Delete a Resource

6.1.2.2. Formats

We should support this format for all our APIs:

- Json: because it's easy to parse in a lot of language like Javascript, Python or Ruby

- XML: Java developers like it
- Atom: It is the standard, it can be used by many applications.

listener

6.1.2.3. Parameters common to every APIs

6.1.2.4. Data Format

The default format is JSON.

The format of the response can be specified by a parameter in the request: "*format*" Specify the format requested.

6.1.2.5. REST configuration

First, we have to register REST service class to the configuration file in the package named **conf.portal**

```
<configuration
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd http://www.exoplaform.org/xml/ns/kernel_1
   xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd">

   <component>
          <type>org.exoplatform.services.ecm.publication.REST.presentation.document.publication.PublicationGetDo
   </component>
</configuration>
```

6.1.2.6. Create a REST service

We start to create GetEditedDocumentRESTService that implements from the ResourceContainer interface.

```
@Path("/presentation/document/edit/")
        public class GetEditedDocumentRESTService implements ResourceContainer {
                @Path("/{repository}/")
                @GET
                public Response getLastEditedDoc(@PathParam("repository") String repository,
                   @QueryParam("showItems") String showItems) throws Exception {
                   ........
                }
        }
```

Parameters	**Definition**
@Path("/presentation/document/edit/")	Specify the URI path that a resource or class method will serve requests for.
@PathParam("repository")	Bind the value **repository** of a URI parameter or a path segment containing the template parameter to a resource method parameter, resource class field, or resource class bean property.
@QueryParam("showItems")	Bind the value **showItems** of a HTTP query parameter to a resource method parameter, resource class field, or resource class bean property.

6.2. UI Extensions

This part describes how to create an sample of UI extension.

6.2.1. Overview

Since DMS 2.5, it is possible to extend the **File Explorer** and the **ECM Administration** with the **UI Extension Framework**. Indeed, you can add your own action buttons to the **File Explorer** and/or add your own managers to the **ECM Administration**.

6.2.2. How to add your own tab in ECM Administration

6.2.2.1. Add your own UIAction

- Create a pom.xml from the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:sch
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.exoplatform.ecms</groupId>
    <artifactId>exo-ecms-examples-uiextension-framework</artifactId>
    <version>2.2.0-SNAPSHOT</version>
  </parent>
  <artifactId>exo-ecms-examples-uiextension-framework-manage-wcm-cache</artifactId>
  <name>eXo WCM Cache Examples </name>
  <description>eXo WCM Cache Examples </description>
  <dependencies>
    <dependency>
      <groupId>org.exoplatform.kernel</groupId>
      <artifactId>exo.kernel.container</artifactId>
      <version>2.2.6-CR02</version>
      <scope>compile</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.commons</groupId>
      <artifactId>exo.platform.commons.webui.ext</artifactId>
      <version>1.0.1</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.ecms</groupId>
      <artifactId>exo-ecms-core-webui</artifactId>
      <version>2.1.2-CR01</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.ecms</groupId>
      <artifactId>exo-ecms-core-webui-administration</artifactId>
      <version>2.1.2-CR01</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  <build>
    <resources>
      <resource>
        <directory>src/main/java</directory>
        <includes>
          <include>**/*.xml</include>
        </includes>
      </resource>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>**/*.properties</include>
          <include>**/*.xml</include>
```

```
          <include>**/*.jar</include>
          <include>**/*.pom</include>
          <include>**/*.conf</include>
          <include>**/*.gtmpl</include>
          <include>**/*.gif</include>
          <include>**/*.jpg</include>
          <include>**/*.png</include>
        </includes>
      </resource>
    </resources>
  </build>
</project>
```

- Create the directories *src/main/java* and start launching *mvn eclipse:eclipse_. You can then, launch your eclipse and import this new project.*

- Create a new class called **org.exoplatform.wcm.component.cache.UIWCMCacheComponent** that extends **org.exoplatform.ecm.webui.component.admin.manager.UIAbstractManagerComponent**.

6.2.2.2. Add your own ActionListener

The webui framework allows you to be notified when a given action has been triggered, you just need to call your own action listener as follows (*ACTION*NAME) **ActionListener**. For example: You create your own action listener name **CacheView**, so your action listener then will be named as **CacheViewActionListener**.

- First, add a static inner class called **CacheViewActionListener** that extends **org.exoplatform.ecm.webui.component.admin.listener.UIECMAdminControlPanelActionListener** .

See the expected code below:

```
public class CacheViewComponent extends UIAbstractManagerComponent {
public static class CacheViewActionListener extends UIECMAdminControlPanelActionListener<UIWCMCacheComponent> {
  public void processEvent(Event<UIWCMCacheComponent> event) throws Exception {UIECMAdminPortlet portlet = event
     UIECMAdminWorkingArea uiWorkingArea = portlet.getChild(UIECMAdminWorkingArea.class);
     uiWorkingArea.setChild(UIWCMCachePanel.class) ;
     event.getRequestContext().addUIComponentToUpdateByAjax(uiWorkingArea);
  }
 }
```

- Create the directories *src/main/java* and launch *mvn eclipse:eclipse_. You can then, launch your eclipse and import this new project.*

- Create a new configuration file *conf/portal/configuration.xml* to register your action (see 6.2.2.3) with the service *org.exoplatform.webui.ext.UIExtensionManager*.

6.2.2.3. Register your UI Action

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<configuration
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd http://www.exoplaform.org/xml/ns/kernel_1
   xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd">

        <external-component-plugins>
        <target-component>org.exoplatform.webui.ext.UIExtensionManager</target-component>
        <component-plugin>
            <name>Add.Actions</name>
                <set-method>registerUIExtensionPlugin</set-method>
                <type>org.exoplatform.webui.ext.UIExtensionPlugin</type>
```

```
               <init-params>
                  <object-param>
                       <name>CacheView</name>
                       <object type="org.exoplatform.webui.ext.UIExtension">
                               <field name="type"><string>org.exoplatform.ecm.dms.UIECMAdminControlPanel</strin
                               <field name="name"><string>CacheView</string></field>
                               <field name="category"><string>GlobalAdministration</string></field>
                               <field name="component"><string>org.exoplatform.wcm.component.cache.UIWCMCacheCo
                           </object>
                  </object-param>
                       <object-param>
                      <name>UIWCMCacheManager</name>
                      <object type="org.exoplatform.webui.ext.UIExtension">
                          <field name="type"><string>org.exoplatform.ecm.dms.UIECMAdminControlPanel</string></f
                          <field name="name"><string>UIWCMCacheManager</string></field>
                          <field name="category"><string>GlobalAdministration</string></field>
                          <field name="component"><string>org.exoplatform.wcm.manager.cache.UIWCMCacheManagerCo
                      </object>
                  </object-param>
              </init-params>
      </component-plugin>
</external-component-plugins>
</configuration>
```

Launch **mvn clean install** and copy the file *target/exo-ecms-examples-webui-2.1.0-SNAPSHOT.jar* into *(TOMCAT*HOME)/lib

6.2.2.4. Define label for your actions and register the resource bundle

Note

All resources can be located in the *src/main/resource* package because the resources (*.xml, images, conf file) and the code are separated. This is very useful in a hierarchical structure.

Create *ExamplePortlet*en.xml *with the following content and add it to* src/main/resource *package:*

```
<?xml version="1.0" encoding="UTF-8"?>
<bundle>
  <!--
  ################################################################################
  #             org.exoplatform.wcm.component.cache.UIWCMCacheForm #
  ################################################################################
  -->

 <UIWCMCacheForm>
          <action>
                  <Cancel>Cancel</Cancel>
                  <Save>Save</Save>
                <Clear>Clear the cache</Clear>
          </action>
         <label>
                <maxsize>Max size :</maxsize>
                <livetime>Live time in sec :</livetime>
                <isCacheEnable>Cache enabled(should always be on production enviroment)</isCacheEnable>
                <hit>Hit count :</hit>
                <currentSize>Current size</currentSize>
                <miss>Miss count :</miss>
        </label>
 </UIWCMCacheForm>

<!--
  ################################################################################
  #             org.exoplatform.wcm.manager.cache.UIWCMCacheManagerForm #
  ################################################################################
  -->

 <UIWCMCacheManagerForm>
          <action>
                  <Cancel>Cancel</Cancel>
```

```
                  <Save>Save</Save>
                <Clear>Clear the cache</Clear>
          </action>
          <label>
                <cacheModify>Cache to modify :</cacheModify>
                <maxsize>Max size :</maxsize>
                <livetime>Live time in sec :</livetime>
                <isCacheEnable>Cache enabled(should always be on production enviroment)</isCacheEnable>
                <hit>Hit count :</hit>
                <currentSize>Current size</currentSize>
                <miss>Miss count :</miss>
         </label>
 </UIWCMCacheManagerForm>

 <UIECMAdminControlPanel>
    <tab>
      <label>
        <GlobalAdministration>Global Administration</GlobalAdministration>
      </label>
    </tab>
    <label>
      <UIWCMCache>WCM Cache</UIWCMCache>
      <UIWCMCachePanel>WCM Cache Administration</UIWCMCachePanel>
      <UIWCMCacheManager>Managing Caches</UIWCMCacheManager>
      <UIWCMCacheManagerPanel>WCM Cache Management</UIWCMCacheManagerPanel>
    </label>
  </UIECMAdminControlPanel>
</bundle>
```

You must add the following content to *configuration.xml- to register the resource bundle.*

Note

By being added this configuration, the resource bundle has been completely separated the resource bundle from the original system, this is so clearly useful, you get an independent plugin.

```
<external-component-plugins>
    <!-- The full qualified name of the ResourceBundleService -->
    <target-component>org.exoplatform.services.resources.ResourceBundleService</target-component>
    <component-plugin>
      <!-- The name of the plugin -->
      <name>ResourceBundle Plugin</name>
      <!-- The name of the method to call on the ResourceBundleService in order to register the ResourceBundles
      <set-method>addResourceBundle</set-method>
      <!-- The full qualified name of the BaseResourceBundlePlugin -->
      <type>org.exoplatform.services.resources.impl.BaseResourceBundlePlugin</type>
      <init-params>
          <values-param>
              <name>init.resources</name>
              <description>Store the following resources into the db for the first launch </description>
              <value>locale.portlet.cache.ExamplePortlet</value>
          </values-param>
          <values-param>
              <name>portal.resource.names</name>
              <description>The properties files of the portal , those file will be merged
              into one ResoruceBundle properties </description>
              <value>locale.portlet.cache.ExamplePortlet</value>
          </values-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
```

6.2.2.5. Run your own UI extension sample

- Run tomcat

- Sign in as root
- Go to the ECM Administration.

You will see a new extension has been already added to ECM Administrator.

6.3. Authoring Extension

Publication addons for WCM 2.1.0

6.3.1. Extended Publication Plugin

6.3.1.1. States

In this extended publication, we will allow new states and new profiles. These states already exist in WCM but not used and we will enable them.

1. Profiles
   - Author : This profile can edit a content and mark this content as redacted
   - Approver : This profile approves a pending content (marked by the Author)
   - Publisher : This profile publishes contents or marks them as "Ready for publication" in multi-server mode
   - Archiver : An administrative profile which moves contents to an archive storage.
2. States
   - enrolled : It's a pure technical state, generally used for content creation.
   - draft (Author) : Content is in editing phase
   - pending (Author) : Author validates the content
   - approved (Approver) : Content is approved by the Manager
   - inreview (Manager) : This state can be used when a second approval state is needed (for i18 translation for example)
   - staged (Publisher) : Content is ready for publication (multi-server mode)
   - published (Publisher or Automatic) : Content is published and visible in Live mode
   - unpublished (Publisher or Automatic) : Content is not visible in Live mode
   - obsolete : Content can still be published but we consider it's not in an editing lifecycle anymore
   - archived (Automatic) : Content is archived and ready to be moved in archive workspace if enabled.

6.3.1.2. Start/End publication dates

In most cases, Customers don't want to publish directly a content but at a defined date and they also want to unpublish automatically the content after that. In the new publication plugin, we add new properties to manage this:

- publication:startPublishedDate
- publication:endPublishedDate

The WCM 2.0 rendering engine doesn't know anything about publication dates, so another service needs to manage that. When the publisher sets start/end publication dates, he can "stage" the content. The content will go automatically to "published" state when start date arrives and to "unpublished" state after end date. A cron job checks every hour (or less) all contents which need to be published (start date in the past + "staged" state) or unpublished (end date in the past + "published" state).

Because of that, publication dates are not mandatory and a content can go to:

- staged : in multi-server mode, publisher can only put the content to staged and wait for auto-publication
- published : in single-server mode, publisher can directly publish a content (with or without publication dates).

6.3.1.3. New Publication Mixin

```
<nodeType hasOrderableChildNodes="false" isMixin="true" name="publication:authoringPublication" primaryItemName=
  <supertypes>
    <supertype>publication:stateAndVersionBasedPublication</supertype>
  </supertypes>
  <propertyDefinitions>
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="publication:startPublishedDa
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="publication:endPublishedDate
      <valueConstraints/>
    </propertyDefinition>
  </propertyDefinitions>
</nodeType>
```

6.3.1.3.1. Publication plugin UI

Note that some labels like "Pr?t ? publier" could be not well displayed in the publication UI. You can extend the current UI State button width by adding :

```
.UIPublicationPanel .StatusTable .ActiveStatus {
width:75px !important;
}
```

Another quick note about publication date inputs. UIPublicationPanel should not initialize the dates to any default value. The publishing and unpublish cron jobs will do this :

- a staged document with null publication start date is published instantly
- a document with null publication end date is published forever

See export section for more info about cron jobs.

6.3.2. Publication Manager

The Publication Manager manages lifecycles and contexts in the WCM Platform. It allows to manages different lifecycles based on different publication plugin in the platform.

```
public interface PublicationManager {

  public List<Lifecycle> getLifecycles();

  public List<Context> getContexts();

  public Context getContext(String name);

  public Lifecycle getLifecycle(String name);

  public List<Lifecycle> getLifecyclesFromUser(String remoteUser, String state);
}
```

- getLifecycles : returns a list of lifecycles (see below), with lifecycle name, publication plugin involved and possible states
- getContexts : return a list of Context, with name, related Lifecycle and other properties (see below)
- getContext : return a context by its name
- getLifecycle : return a lifecycle by its name
- getLifecycleFromUser : returns a list of Lifecycles in which the user has rights (based on membership property)

6.3.2.1. Lifecycle

A lifecycle is defined by a simple vertical workflow with steps (states) and profiles (membership). Each lifecycle is related to a Publication plugin (to be compliant with JBPM or Bonita business processes). Example : *Two lifecycles with/without states*

```
Exception occured, see logs
```

In the last example, we have two lifecycles :

1. lifecycle 1 : based on *States and versions based publication*#* This allows to be backward compliant with older WCM releases. If all your site contents are using an existing plugin, you can create a lifecycle for it and it will work.
    - For new instances, we recommend to use the new plugin with dynamic states capabilities
2. lifecycle 2 : based on new *Authoring publication*#* Visibility : we define only the "visible" steps. In this example, there's no step for 'enrolled'. Even if this step exists, it will not be displayed in the UI.
    - Automatic : we can set a step as "automatic". In this mode, the step will be visible in the UI but it will be managed by the system (a cron job for example).
3. lifecycle 3 : simulates the *States and versions based publication* plugin
    - Note that this simple lifecycle will work in a single server configuration

6.3.2.1.1. Listening to a lifecycle

When we change state, we broadcast an event in order for partners to add features if they want to. Event could look like this :

```
listenerService.broadcast(AuthoringPlugin.POST_UPDATE_STATE_EVENT, null, node);
```

Listener declaration could look like this :

```
<external-component-plugins>
  <target-component>org.exoplatform.services.listener.ListenerService</target-component>
  <component-plugin>
    <name>PublicationService.event.postUpdateState</name>
    <set-method>addListener</set-method>
    <type>org.exoplatform.services.wcm.publication.listener.post.PostUpdateStateEventListener</type>
    <description>this listener will be called every time a content changes its current state</description>
  </component-plugin>
</external-component-plugins>
```

6.3.2.2. Context

A context is defined by simple rules. In WCM 2.1+, we can choose to enroll the content in a specific lifecycle (i.e. publication plugin) based on context parameters. We have 3 parameters we can use to define contexts :

- Remote User : The current user who create/edit the content.
- Current sitename : The Site from where the content is created (not the storage but the navigation)
- Node : The node which we want to enroll.

From these parameters, we can easily connect and define contexts based on :

- Membership : Does the current user have this membership ?
- Site : On this particular site, we want to enroll contents in a specific lifecycle
- Path : We can enroll content in lifecycles based on their path (from the Node).
- Type of content : We can enroll content in lifecycles based on their nodetype (from the Node).

Because each site has a content storage (categories + physical storage), we can choose the right lifecycle for the right storage/site. We can have conflicts on contexts and we will avoid that by setting a priority (less is best)

Example : *Different Contexts*

```
Exception occured, see logs
```

The logic is very simple. When we create a content, we will go lifecycle by lifecycle starting with the better priority :

- *context4* is the most important (priority=50) : we will enroll the content in the lifecycle 'lifecycle4' if :
  - the content creator has the 'manager:/company/finances' membership

- the content is stored in 'repository:collaboration:/documents/company/finances' or any subfolders
- the content is a 'exo:article'

- if not we will continue with *context3*
- etc

We can see that *context1* and *context2* have the same priority. We can do that if the contexts are different (ex : different sites)

Note

It's important to notice that contexts will be used only when the content is created and when we want to enroll it in a lifecycle for the first time. Once we have the corresponding lifecycle, we will set the lifecycle inside the content (see New authoring Mixin) and the context service will not be called again for this content.

6.3.2.3. New Authoring Mixin

```
<nodeType hasOrderableChildNodes="false" isMixin="true" name="publication:authoring" primaryItemName="">
  <propertyDefinitions>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="publication:lastUser" onPar
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="publication:lifecycle" onPa
      <valueConstraints/>
    </propertyDefinition>
  </propertyDefinitions>
</nodeType>
```

When adding the content in a lifecycle, we set the *publication:lifecycle_ property with the corresponding lifecycle.*

Note

A content can be in one lifecycle only.

Each time, we change from one state to another, we set the user who changed the state in *publication:lastUser_*.

6.3.2.3.1. Querying based on publication status

By adding this mixin to contents, we can access contents by simple queries based on the current user profile. For example:

- All my draft contents
  - query : select * from nt:base where publication:currentState='draft' and publication:lastUser='benjamin';
- All the contents I have to approve
  - call : PublicationManager.getLifecycles('benjamin', 'approved') => returns lifecycles where I can go to the 'approved' state.

  - query : select * from nt:base where publication:currentState='pending' and (publication:lifecycle='lifecycle1' or publication:lifecycle='lifecycle3');

- All the content that will be published tomorrow
  - query : select * from nt:base where publication:currentState='staged' and publication:startPublishedDate>'xxxx';

Chapter 7. Java Services

7.1. TaxonomyService

Taxonomy service is used to work with taxonomies. In this service, there are many functions which enable you to add, find, delete taxonomies from a node.

Method	Return	Prototype	Description
getTaxonomyTree	`Node`	`getTaxonomyTree(String repository, String taxonomyName, boolean system) throws RepositoryException;`	Returns the root node of the given taxonomy tree @param repository: The name of repository @param taxonomyName: The name of the taxonomy @param system: Indicates whether the nodes must be retrieved using a session system or user session @throws RepositoryException: if the taxonomy tree could not be found
getTaxonomyTree	`Node`	`getTaxonomyTree(String repository, String taxonomyName) throws RepositoryException;`	Returns the root node of the given taxonomy tree with the user session @param repository: The name of repository @param taxonomyName: The name of the taxonomy @throws

Method	Return	Prototype	Description
			RepositoryException: if the taxonomy tree could not be found
getAllTaxonomyTrees	`List<Node>`	`getAllTaxonomyTrees(String repository, boolean system) throws RepositoryException;`	Returns the list of all the root nodes of the taxonomy tree available @param repository: The name of repository @param system: Indicates whether the nodes must be retrieved using a session system or user session @throws RepositoryException: if the taxonomy trees could not be found
getAllTaxonomyTrees	`List<Node>`	`getAllTaxonomyTrees(String repository) throws RepositoryException;`	Returns the list of all the root nodes of the taxonomy tree available with the user session @param repository: The name of repository @throws RepositoryException: if the taxonomies could not be found
hasTaxonomyTree	`boolean`	`hasTaxonomyTree(String repository, String taxonomyName) throws RepositoryException;`	Checks if a taxonomy tree with the given name has already been defined @param repository: The name of repository

Method	Return	Prototype	Description
			@param taxonomyName: The name of the taxonomy @throws RepositoryException: if the taxonomy name could not be checked
addTaxonomyTree	`void`	`addTaxonomyTree(Node taxonomyTree) throws RepositoryException, TaxonomyAlreadyExistsException;`	Defines a node as a new taxonomy tree @param taxonomyTree: The taxonomy tree to define @throws TaxonomyAlreadyExistsException: if a taxonomy with the same name has already been defined @throws RepositoryException: if the taxonomy tree could not be defined
updateTaxonomyTree	`void`	`updateTaxonomyTree(String taxonomyName, Node taxonomyTree) throws RepositoryException;`	Re-defines a node as a taxonomy tree @param taxonomyName: The name of the taxonomy to update @param taxonomyTree: The taxonomy tree to define @throws RepositoryException: if the taxonomy tree could not be updated

Method	Return	Prototype	Description
removeTaxonomyTree	`void`	`removeTaxonomyTree(String taxonomyName) throws RepositoryException;`	Remove the taxonomy tree definition @param taxonomyName: The name of the taxonomy to remove @throws RepositoryException: if the taxonomy tree could not be removed
addTaxonomyNode	`void`	`addTaxonomyNode(String repository, String workspace, String parentPath, String taxoNodeName, String creator) throws RepositoryException, TaxonomyNodeAlreadyExistsException;`	Adds a new taxonomy node at the given location @param repository: The name of the repository @param workspace: The name of the workspace @param parentPath: The place where the taxonomy node will be added @param taxoNodeName :The name of taxonomy node @param creator: The name of the user creating this node @throws TaxonomyNodeAlreadyExistsExceptio if a taxonomy node with the same name has already been added @throws

Method	Return	Prototype	Description
			RepositoryException: if the taxonomy node could not be added
removeTaxonomyNode	`void`	`removeTaxonomyNode(String repository, String workspace, String absPath) throws RepositoryException;`	Remove the taxonomy node located at the given absolute path @param repository: The name of the repository @param workspace: The name of the workspace @param absPath: The absolute path of the taxonomy node to remove @throws RepositoryException: if the taxonomy node could not be removed
moveTaxonomyNode	`void`	`moveTaxonomyNode(String repository, String workspace, String srcPath, String destPath, String type) throws RepositoryException;`	Copies or cuts the taxonomy node from source path to destination path. The parameter type indicates if the node must be cut or copied @param repository: The name of the repository @param workspace: The name of the workspace @param srcPath: The source path of this taxonomy @param destPath: The destination path of the

Method	Return	Prototype	Description
			taxonomy @param type: If **type** is equal to **cut**, the process will be cut. If **type** is equal to **copy**, the process will be copied @throws RepositoryException: if the taxonomy node could not be moved
hasCategories	`boolean`	`hasCategories(Node node, String taxonomyName) throws RepositoryException;`	Returns true if the given node has categories in the given taxonomy @param node: The node to check @param taxonomyName: The name of the taxonomy @throws RepositoryException: if categories cannot be checked
hasCategories	`boolean`	`hasCategories(Node node, String taxonomyName, boolean system) throws RepositoryException;`	Return true if the given node has categories in the given taxonomy @param node: The node to check @param taxonomyName: The name of the taxonomy @param system: check system provider or not

Method	Return	Prototype	Description
			@throws RepositoryException: if categories cannot be checked
getCategories	`List<Node>`	`getCategories(Node node, String taxonomyName) throws RepositoryException;`	Returns all the paths of the categories (relative to the root node of the given taxonomy) which have been associated to the given node for the given taxonomy @param node: The node for which we seek the categories @param taxonomyName: The name of the taxonomy @throws RepositoryException: if the categories cannot be retrieved
getCategories	`List<Node>`	`getCategories(Node node, String taxonomyName, boolean system) throws RepositoryException;`	Returns all the paths of the categories(relative to the root node of the given taxonomy) which have been associated to the given node for the given taxonomy @param node: The node for which we seek the categories @param taxonomyName: The name of the taxonomy @param system

Method	Return	Prototype	Description
			@throws RepositoryException: if the categories cannot be retrieved
getAllCategories	`List<Node>`	`getAllCategories(Node node) throws RepositoryException;`	Returns all the paths of the categories which have been associated to the given node @param node: The node for which we seek the categories @throws RepositoryException
getAllCategories	`List<Node>`	`getAllCategories(Node node, boolean system) throws RepositoryException;`	Returns all the paths of the categories which have been associated to the given node @param node: The node for which we seek the categories @param system: check system provider or not @throws RepositoryException
removeCategory	`void`	`removeCategory(Node node, String taxonomyName, String categoryPath) throws RepositoryException;`	Removes a category to the given node @param node: The node for which we remove the category @param taxonomyName: The name of the taxonomy

Method	Return	Prototype	Description
			@param categoryPath: The path of the category relative to the root node of the given taxonomy @throws RepositoryException: if the category cannot be removed
removeCategory	`void`	`removeCategory(Node node, String taxonomyName, String categoryPath, boolean system) throws RepositoryException;`	Removes a category to the given node @param node: The node for which we remove the category @param taxonomyName: The name of the taxonomy @param categoryPath: The path of the category relative to the root node of the given taxonomy @param system: check system provider or not @throws RepositoryException: if the category cannot be removed
addCategories	`void`	`addCategories(Node node, String taxonomyName, String[] categoryPaths) throws RepositoryException;`	Adds several categories to the given node @param node: The node for which we add the categories @param taxonomyName:

Method	Return	Prototype	Description
			The name of the taxonomy @param categoryPaths: An array of category paths relative to the given taxonomy @throws RepositoryException: if the categories cannot be added
addCategories	`void`	`addCategories(Node node, String taxonomyName, String[] categoryPaths, boolean system) throws RepositoryException;`	Adds several categories to the given node @param node: The node for which we add the categories @param taxonomyName: The name of the taxonomy @param categoryPaths: An array of category paths relative to the given taxonomy @param system: check system provider or not @throws RepositoryException: if the categories cannot be added
addCategory	`void`	`addCategory(Node node, String taxonomyName, String categoryPath) throws RepositoryException;`	Adds a new category path to the given node @param node: the node for which we add the

Method	Return	Prototype	Description
			category @param taxonomyName: The name of the taxonomy @param categoryPath: The path of the category relative to the given taxonomy @throws RepositoryException: if the category cannot be added
addCategory	`void`	`addCategory(Node node, String taxonomyName, String categoryPath, boolean system) throws RepositoryException;`	Adds a new category path to the given node @param node: the node for which we add the category @param taxonomyName: The name of the taxonomy @param categoryPath: The path of the category relative to the given taxonomy @param system: check system provider or not @throws RepositoryException: if the category cannot be added
getTaxonomyTreeDefaultUserPermission	`Map<String, String[]>`	`getTaxonomyTreeDefaultUserPermission();`	get default permission for the user in taxonomy tree

Method	Return	Prototype	Description
addTaxonomyPlugin	`void`	`addTaxonomyPlugin(ComponentPlugin plugin);`	Add a new taxonomy plugin to the service @param plugin: The plugin to add
init	`void`	`init(String repository) throws Exception;`	Initialize all taxonomy plugins that have been already configured in .xml files @param repository: The name of repository @see TaxonomyPlugin @throws Exception

7.2. LinkManager

Supply API to work with the linked node or the link included in a node.

Package org.exoplatform.services.cms.link.LinkManager

Method	Return	Prototype	Description

createLink	`Node`	`createLink(Node parent, String linkType, Node target)`
		
		`throws RepositoryException;`
			Creates a new link, add it to the parent node and returns the link
			
			@param parent: The parent node of the link
			
			@param linkType: The primary node type of the link must be a sub-type of
			
			exo:symlink, the default value is "exo:symlink"

Method	Return	Prototype	Description
			@param target The target of the link @throws RepositoryException: if the link cannot be created for any reason

createLink	`Node`	`createLink(Node parent, Node target)`
		
		`throws RepositoryException;`
			Creates a new node of type exo:symlink, add it to the parent node and
			
			returns the link node
			
			@param parent: The parent node of the link to create
			
			@param target: The target of the link
			
			@throws RepositoryException: if the link cannot be created for any reason

createLink	`Node`	`createLink(Node parent, String linkType, Node target, String linkName)`
		
		`throws RepositoryException;`
			Creates a new link, add it to the parent node and returns the link
			
			@param parent: The parent node of the link
			
			@param linkType: The primary node type of the link must be a sub-type of
			
			exo:symlink, the default value is "exo:symlink"

Method	Return	Prototype	Description
			@param target: The target of the link @param linkName: The name of the link @return @throws RepositoryException: if the link cannot be created for any reason

updateLink	`Node`	`updateLink(Node link, Node target)`
		
		`throws RepositoryException;`
			Updates the target node of the given link
			
			@param link: The link node to update
			
			@param target: The new target of the link
			
			@throws RepositoryException: if the link cannot be updated for any reason

getTarget	`Node`	`getTarget(Node link, boolean system) throws ItemNotFoundException,`
		
		`RepositoryException;`
			Gets the target node of the given link
			
			@param link: The node of type exo:symlink
			
			@param system: Indicates whether the target node must be retrieved using a
			
			session system or user session in case we cannot use the same

Method	Return	Prototype	Description
			session as the node link because the target and the link are not in the same workspace @throws ItemNotFoundException: if the target node cannot be found @throws RepositoryException: if an unexpected error occurs while retrieving the target node

getTarget	`Node`	`getTarget(Node link)`
		
		`throws ItemNotFoundException, RepositoryException;`
			Gets the target node of the given link using the user session
			
			@param link: The node of type exo:symlink
			
			@throws ItemNotFoundException: if the target node cannot be found
			
			@throws RepositoryException: if an unexpected error occurs while retrieving
			
			the target node

isTargetReachable	`boolean`	`isTargetReachable(Node link)`	Checks if the target node of the given link can be reached using the user

Method	Return	Prototype	Description
		`throws RepositoryException;`	session @param link: The node of type exo:symlink @throws RepositoryException: if an unexpected error occurs

isTargetReachable	`boolean`	`isTargetReachable(Node link, boolean system)`
		
		`throws RepositoryException;`
			Checks if the target node of the given link can be reached using the user session
			
			@param link: The node of type exo:symlink
			
			@param system
			
			@throws RepositoryException if an unexpected error occurs

isLink	`boolean`	`isLink(Item item)`
		
		`throws RepositoryException;`
			Indicates whether the given item is a link
			
			@param item: the item to test
			
			@return <code>true</code>: if the node is a link, <code>false</code> otherwise
			
			@throws RepositoryException: if an unexpected error

Method	Return	Prototype	Description
			occurs
getTargetPrimaryNodeType	`String`	`getTargetPrimaryNodeType(Node link) throws RepositoryException;`	Give the primary node type of the target @param link: The node of type exo:symlink @return: the primary node type of the target @throws RepositoryException: if an unexpected error occurs

7.3. PublicationManager

PublicationService to manage the publication

Method	Return	Prototype	Description
addLifecycle	`void`	`addLifecycle(ComponentPlugin plugin);`	Add plugins context for the publication service
getLifecycle	`Lifecycle`	`Lifecycle getLifecycle(String name) ;`	Get a specific Lifecycle with the given name @param name: Name of wanted Lifecycle
getLifecycles	`List<Lifecycle>`	`List<Lifecycle> getLifecycles();`	Get all the Lyfecycle which were added to service instance
getContext	`Context`	`Context getContext(String name) ;`	Get a specific context with the given name
getContexts	`List<Context>`	`List<Context> getContexts();`	Get all the context which were added to service instance
getContents	`List<Node>`	`List<Node> getContents(String`	Get all the node

Method	Return	Prototype	Description
		`fromstate, String tostate, String date, String user, String lang , String workspace) throws Exception;`	@param fromstate/tostate: the current range state of node @param @param user: The last user of node @param lang: the node's language @param workspace: the Workspace of node's location
getLifecyclesFromUser	`List<Lyfecycle>`	`List<Lyfecycle> getLifecyclesFromUser(String remoteUser, String state);`	Get all the Lifecycle of a specific user@param remoteUser: current user of publication service @param state: Current state of the node

7.4. WCMComposer

This class is used to get contents inside the WCM product. We should not access contents directly from the jcr on the front side.

In generall, this service stands between publication and cache.

Package org.exoplatform.services.wcm.publication.WCMComposer

Method	Return	Prototype	Description

getContent	`Node`	`getContent(String repository, String workspace, String nodeIdentifier, HashMap<String, String> filters, SessionProvider sessionProvider)`
		
		`throws Exception ;`
			returns content at the specified path based on filters.
			
			repository the repository
			
			workspace the workspace
			
			@param path: the path

Method	Return	Prototype	Description
			@param filters: the filters @param sessionProvider: the session provider @return a jcr node @throws Exception: the exception

getContents	`List<Node>`	`getContents(String repository, String workspace, String path, HashMap<String, String> filters, SessionProvider sessionProvider)`
		
		`throws Exception ;`
			returns contents at the specified path based on filters.
			
			@param repository: the repository
			
			@param workspace: the workspace
			
			@param path: the path
			
			@param filters: the filters
			
			@param sessionProvider: the session provider
			
			@return a jcr node
			
			@throws Exception: the exception

updateContent	`boolean`	`updateContent(String repository, String workspace, String nodeIdentifier, HashMap<String, String> filters)`	Update content. repository the repository @param workspace: the

Method	Return	Prototype	Description
		`throws Exception;`	workspace @param path: the path @param filters: the filters @return true, if successful

getAllowedStates	`List<String>`	`getAllowedStates(String mode)`
		
		`throws Exception ;`
			returns allowed states for a specified mode.
			
			@param mode: the mode
			
			@return a jcr node
			
			@throws Exception: the exception

cleanTemplates	`void`	`cleanTemplates()`
		
		`throws Exception ;`
			initialize the templates hashmap
			
			@throws Exception: the exception

isCached	`boolean`	`isCached()`
		
		`throws Exception;`
			Check isCache or not
			
			@return the state of cache
			
			@throws Exception: the exception

7.5. NewFolksonomy

Package org.exoplatform.services.cms.folksonomy.NewFolksonomyService;

Method	Return	Prototype	Description
addPrivateTag	`void`	`addPrivateTag(String[] tagsName, Node documentNode, String repository, String workspace, String userName) throws Exception ;`	Add a private tag to a document. A folksonomy link will be created in a tag node @param tagNames: Array of tag name as the children of tree @param documentNode: Tagging this node by creating a folksonomy link to the node in tag @param repository: Repository name @param workspace: Workspace name @param userName: User name @throws Exception
addGroupsTag	`void`	`addGroupsTag(String[] tagsName, Node documentNode,String repository, String workspace, String[] roles) throws Exception ;`	Add a group tag to a document. A folksonomy link will be created in a tag node @param tagNames: Array of tag name as the children of tree @param documentNode: Tagging this node by creating a folksonomy link to the node in tag @param repository: Repository name @param workspace: Workspace name

Method	Return	Prototype	Description
			@param roles: User roles @throws Exception
addPublicTag	`void`	`addPublicTag(String treePath, String[] tagsName, Node documentNode, String repository, String workspace) throws Exception ;`	Add a public tag to a document. A folksonomy link will be created in a tag node @param treePath: Path of folksonomy tree @param tagNames: Array of tag name as the children of tree @param documentNode: Tagging this node by creating a folksonomy link to the node in tag @param repository: Repository name @param workspace: Workspace name @throws Exception
addSiteTag	`void`	`addSiteTag(String siteName, String[] tagsName, Node node, String repository, String workspace) throws Exception ;`	Add a site tag to a document. A folksonomy link will be created in a tag node @param siteName: Portal name @param treePath: Path of folksonomy tree @param tagNames: Array of tag name as the children of tree

Method	Return	Prototype	Description
			@param documentNode: Tagging this node by creating a folksonomy link to the node in tag @param repository: Repository name @param workspace: Workspace name @throws Exception
getAllPrivateTags	`List<Node>`	`getAllPrivateTags(String userName, String repository, String workspace) throws Exception ;`	Get all private tags @param userName: User name @param repository: repository name @param workspace: Workspace name @return List<Node>
getAllPublicTags	`List<Node>`	`getAllPublicTags(String treePath, String repository, String workspace) throws Exception ;`	Get all public tags @param treePath: Folksonomy tree path @param repository: Repository name @param workspace: Workspace name @return List<Node> @throws Exception
getAllGroupTags	`List<Node>`	`getAllGroupTags(String[] roles, String`	Get all tags by groups

Method	Return	Prototype	Description
		`repository, String workspace) throws Exception ;`	@param roles: Roles of user @param repository: Repository name @param workspace: Workspace name @return List<Node> @throws Exception
getAllGroupTags	`List<Node>`	`getAllGroupTags(String role, String repository, String workspace) throws Exception ;`	Get all tags by group @param role: Role of user @param repository: Repository name @param workspace: Workspace name @return List<Node> @throws Exception
getAllSiteTags	`List<Node>`	`getAllSiteTags(String siteName, String repository, String workspace) throws Exception ;`	Get all tags of Site @param siteName: Portal name @param treePath: Folksonomy tree path @param repository: Repository name @param workspace: Workspace name

Method	Return	Prototype	Description
			@return List<Node> @throws Exception
getAllDocumentsByTag	`List<Node>`	`getAllDocumentsByTag(String tagPath, String repository, String workspace, SessionProvider sessionProvider) throws Exception ;`	Get all documents which are stored in a tag @param treeName: Name of folksonomy tree @param tagName : Name of tag @param repository: Repository name @return: List of documents in tag @throws Exception
getTagStyle	`String`	`getTagStyle(String tagPath, String repository, String workspace) throws Exception ;`	Get HTML*STYLE*PROP property in styleName node in repository @param tagPath: Tag path @param workspace Workspace name @param repository: Repository name @return: value of property of styleName node @throws Exception
addTagStyle	`void`	`addTagStyle(String styleName, String tagRange, String htmlStyle, String`	Update the properties TAG*RATE*PROP and HTML*STYLE*PROP,

Method	Return	Prototype	Description
		`repository, String workspace) throws Exception ;`	following the values tagRate, htmlStyle for node in tagPath in repository @param styleName: Style name @param tagRate: The range of tag numbers @param htmlStyle: Tag style @param repository: Repository name @param workspace: Workspace name @throws Exception
updateTagStyle	`void`	`updateTagStyle(String styleName, String tagRange, String htmlStyle, String repository, String workspace) throws Exception ;`	Update the properties TAG*RATE*PROP and HTML*STYLE*PROP, following the value tagRate, htmlStyle for node in tagPath in repository @param styleName: Style name @param tagRate: The range of tag numbers @param htmlStyle Tag style @param repository Repository name

Method	Return	Prototype	Description
			@param workspace: Workspace name @throws Exception
getAllTagStyle	`List<Node>`	`getAllTagStyle(String repository, String workspace) throws Exception ;`	Get all tag style bases of folksonomy tree @param repository: Repository name @param workspace: Workspace name @return List<Node> List tag styles @throws Exception
init	`void`	`init(String repository) throws Exception ;`	Initialize all TagStylePlugin with session in repository name @param repository: repository name
removeTagOfDocument	`void`	`removeTagOfDocument(String tagPath, Node document, String repository, String workspace) throws Exception;`	Remove a tag of a given document @param treeName: Name of folksonomy tree @param tagName: Name of tag @param document: Document which is added a link to tagName @param repository: Repository name

Method	Return	Prototype	Description
			@return @throws Exception
removeTag	`void`	`removeTag(String tagPath, String repository, String workspace) throws Exception;`	Remove tag @param tagPath: Path of tag @param repository: Repository name @param workspace: Workspace name
modifyTagName	`Node`	`modifyTagName(String tagPath, String newTagName, String repository, String workspace) throws Exception;`	Modify tag name @param tagPath: Path of tag @param newTagName: New tag name @param repository: Repository name @param workspace Workspace name @return @throws Exception
getLinkedTagsOfDocument	`List<Node>`	`getLinkedTagsOfDocument(Node documentNode, String repository, String workspace) throws Exception;`	Get all tags linked to given document @param documentNode Document node @param repository: Repository name

Method	Return	Prototype	Description
			@param workspace: Workspace name @return @throws Exception
getLinkedTagsOfDocumentByScope	List<String>	`getLinkedTagsOfDocumentByScope(int scope, String value, Node documentNode, String repository, String workspace) throws Exception;`	Get all tags linked to given document by scope @param documentNode: Document node @param repository: Repository name @param workspace Workspace name @return @throws Exception
removeTagsOfNodeRecursively	void	`removeTagsOfNodeRecursively(Node node, String repository, String workspace, String username, String groups) throws Exception;`	Remove all tags linked to children of given node @param node @param repository @param workspace @throws Exception
addTagPermission	`void`	`addTagPermission(String usersOrGroups);`	Add given users or groups to tagPermissionList @param usersOrGroups
removeTagPermission	`void`	`removeTagPermission(String usersOrGroups);`	Remove given users or groups from

Method	Return	Prototype	Description
			tagPermissionList @param usersOrGroups
getTagPermissionList	`List<String>`	`getTagPermissionList();`	Returns tagPermissionList
canEditTag	`boolean`	`canEditTag(int scope, List<String> memberships);`	Set the permission for a user to edit tag. @param scope: Scope @param memberships: Memberships @return true If it is possible
getAllTagNames	`List<String>`	`getAllTagNames(String repository, String workspace, int scope, String value) throws Exception;`	Get all tag names which start within a given scope @param repository: Repository @param workspace: Workspace @param scope: scope of tags @param value: value, according to scope, can be understood differently @return true if it is possiblee

7.6. ApplicationTemplateManager

This class is used to manage dynamic groovy templates for ecm-based products.

Package org.exoplatform.services.cms.views.ApplicationTemplateManager;

Method	Return	Prototype	Description
addPlugin	`void`	addPlugin(PortletTemplatePlugin portletTemplatePlugin) throws Exception	Adds the plugin. portletTemplatePlugin the portlet template plugin
getAllManagedPortletNames	`List<String>`	getAllManagedPortletNames(String repository) throws Exception	Retrieve all the portlet names that have dynamic groovy templates managed by service. repository the repository @throws Exception the exception
getTemplatesByApplication	`List<Node>`	getTemplatesByApplication(String repository, String portletName, SessionProvider provider) throws Exception;	Retrieve the templates node by application. @param repository: the repository @param portletName: the portlet name @param provider: the provider @return the templates by application @throws Exception: the exception
getTemplatesByCategory	`List<Node>`	getTemplatesByCategory(String repository, String portletName, String category, SessionProvider sessionProvider) throws Exception;	Retrieve the templates node by category:. @param repository: the repository @param portletName: the portlet name

Method	Return	Prototype	Description
			@param category: the category @param sessionProvider: the session provider @return the templates by category @throws Exception: the exception
getTemplateByName	`Node`	getTemplateByName(String repository, String portletName, String category, String templateName, SessionProvider sessionProvider) throws Exception;	Retrieve the template by name @param repository: the repository @param portletName: the portlet name @param category: the category @param templateName: the template name @param sessionProvider: the session provider @return the template by name @throws Exception: the exception
getTemplateByPath	`Node`	getTemplateByPath(String repository, String templatePath, SessionProvider sessionProvider) throws Exception ;	Get the template by path: @param repository: the repository @param templatePath: the template path

Method	Return	Prototype	Description
			@param sessionProvider: the session provider @return the template by path @throws Exception: the exception
addTemplate	`void`	addTemplate(Node portletTemplateHome, PortletTemplateConfig config) throws Exception;	Adds the template: @param portletTemplateHome: the portlet template home @param config: the config @throws Exception: the exception
removeTemplate	`void`	removeTemplate(String repository, String portletName, String catgory, String templateName, SessionProvider sessionProvider) throws Exception;	Remove the template: @param repository: the repository @param portletName: the portlet name @param catgory: the catgory @param templateName: the template name @param sessionProvider: the session provider @throws Exception: the exception

7.7. NodeFinder

NodeFinder is used to find a node with a given path. If the path to the node contains sub-paths to exo:symlink nodes, find real link node.

Method	Return	Prototype	Desription
getNode	`Node`	getNode(Node ancestorNode, String relativePath) throws PathNotFoundException, RepositoryException;	Return the node at relPath related to ancestor node. @param ancestorNode: The ancestor of the node to retrieve from which we start. @param relativePath: The relative path of the node to retrieve. @throws PathNotFoundException: If "no", the node exists at the specified path. @throws RepositoryException: if another error occurs.
getNode	`Node`	getNode(Node ancestorNode, String relativePath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	Return the node at relPath related to ancestor node. If the node is a link and giveTarget has been set to <code>true</code>, the target node will be returned @param ancestorNode: The ancestor of the node to retrieve from which we start. @param relativePath: The relative path of the node to retrieve. @param giveTarget:

Method	Return	Prototype	Desription
			Indicate if the target must be returned in case the item is a link @throws PathNotFoundException: If no node exists at the specified path. @throws RepositoryException: if another error occurs.
getItem	`Item`	getItem(String repository, String workspace, String absPath) throws PathNotFoundException, RepositoryException;	Return the item at the specified absolute path. @param repository: The name of repository @param workspace: The name of workspace @param absPath: An absolute path. @throws PathNotFoundException: if the specified path cannot be found. @throws RepositoryException: if another error occurs.
getItemSys	`Item`	getItemSys(String repository, String workspace, String absPath, boolean system) throws PathNotFoundException , RepositoryException;	Return the item at the specified absolute path. @param repository: The name of repository @param workspace: The name of workspace @param absPath: An

Method	Return	Prototype	Desription
			absolute path. @throws PathNotFoundException: if the specified path cannot be found. @throws RepositoryException: if another error occurs.
getItem	`Item`	getItem(String repository, String workspace, String : absPath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	Return the item at the specified absolute path. If the item is a link and giveTarget has been set to <code>true</code>, the target node will be returned @param repository: The name of repository @param workspace: The name of workspace @param absPath: An absolute path. @param giveTarget: Indicates if the target must be returned in case the item is a link @throws PathNotFoundException: if the specified path cannot be found. @throws RepositoryException: if another error occurs.
getItemGiveTargetSys	`Item`	getItemGiveTargetSys(String repository, String	Return the item at the specified absolute path. If the item is a link and

Method	Return	Prototype	Desription
		workspace, String absPath, boolean giveTarget, boolean system) throws PathNotFoundException, RepositoryException;	giveTarget has been set to <code>true</code>, the target node will be returned @param repository: The name of repository @param workspace: The name of workspace @param absPath: An absolute path. @param giveTarget: Indicates if the target must be returned in case the item is a link @param system: system provider @throws PathNotFoundException: if the specified path cannot be found. @throws RepositoryException: if another error occurs.
getItem	`Item`	getItem(Session session, String absPath) throws PathNotFoundException, RepositoryException;	Return the item at the specified absolute path. @param session: The session to use in order to get the item @param absPath: An absolute path. @throws PathNotFoundException: if the specified path

Method	Return	Prototype	Desription
			cannot be found. @throws RepositoryException: if another error occurs.
getItem	`Item`	getItem(Session session, String absPath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	Return the item at the specified absolute path. If the item is a link and giveTarget has been set to <code>true</code>, the target node will be returned @param session: The session to use in order to get the item @param absPath: An absolute path. @param giveTarget: Indicates if the target must be returned in case the item is a link @throws PathNotFoundException: if the specified path cannot be found. @throws RepositoryException: if another error occurs.
getItemTarget	`Item`	`getItemTarget(Session session, String absPath, boolean giveTarget, boolean system) throws PathNotFoundException, RepositoryException;`	Return the item at the specified absolute path. If the item is a link and giveTarget has been set to <code>true</code>, the target node will be returned @param session: The session to use in order to get the item

Method	Return	Prototype	Desription
			@param absPath: An absolute path. @param giveTarget: Indicates if the target must be returned in case the item is a link @param system: system provider @throws PathNotFoundException: if the specified path cannot be found. @throws RepositoryException: if another error occurs.
itemExists	`boolean`	itemExists(Session session, String absPath) throws RepositoryException;	Return <code>true</code> if an item exists at absPath; otherwise returns <code>false</code> Also return <code>false</code> if the specified absPath is malformed. @param session: The session to use in order to get the item @param absPath: An absolute path. @return <code>true</code> if an item exists at absPath; otherwise returns <code>false</code>. @throws

Method	Return	Prototype	Desription
			RepositoryException: if an error occurs.

7.8. JodConverter

JodConverter is used to convert documents into different office formats.

Package org.exoplatform.services.cms.jodconverter.JodConverterService

Method	Return	Prototype	Desription
convert	`void`	convert(InputStream input, String formatInput, OutputStream out, String formatOutput) throws Exception;	Convert InputStream in with formatInput format to OutputStream out with formatOutput @param input @param formatInput @param out @param formatOutput @throws Exception

Chapter 8. FAQ

8.1.1. How do I create a new Site ?

8.1.2. How do I create a new Content Type ?

8.1.3. How do I create a new Site ?