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 mecanism inside GateIn to add content features. If you want more info about the extension mecanism, 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 as well as 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 action in WCM start from `delivery` folder, so let's 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
```

1.3. Portlets

1.3.1. Content Detail

1.3.1.1. Available preferences

Preference	Type	Default Value	Description
repository	`String`	repository	The current repository name. always is "repository"
workspace	`String`	collaboration	The workspace 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 top of the portlet
ShowDate	`Boolean`	false	Show the content date on top of the portlet
ShowOptionBar	`Boolean`	false	Show a bar with some actions (Print, Back)
ContextEnable	`Boolean`	false	Determine if the portlet will use the parameter on URL as the path to content to display
ParameterName	`String`	content-id	Determine what 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)

1.3.1.2. 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 is some preference is no longer used: for example: ShowPrintAction preference,..

1.3.2. Content List

1.3.2.1. Available preferences

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

Preference	Type	Default Value	Description
			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/UIContentListPresentationDefault.gtmpl	Path to the template used to display the contents in this portlet.
paginatorTemplatePath	`String`	/exo:ecm/views/templates/Content List Viewer/paginators/UIPaginatorDefault.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 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

Preference	Type	Default Value	Description
			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 showing the path of content in url to display when click on "Read more" button of this content.
showClvBy	`String`	folder-id	The name of the parameter showing the path of the folder in url to display its contents.

1.3.2.2. 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>
```

1.3.3. Search

1.3.3.1. Available preferences

Preference	Type	Default value	Description
repository	`String`	repository	The current repository name. always is "repository"
workspace	`String`	collaboration	The workspace where the content is stored
searchFormTemplatePath	`String`	/exo:ecm/views/templates/wcm/Advance Search/search-form/UIDefaultSearchForm.gtmpl	Path to the search form template

Preference	Type	Default value	Description
searchResultTemplatePath	`String`	/exo:ecm/views/templates/WCM Advance Search/search-result/UIDefaultSearchResult.gtmpl	path to the search result template
searchPaginatorTemplatePath	`String`	/exo:ecm/views/templates/WCM Advance Search/search-paginator/UIDefaultSearchPaginator.gtmpl	path to the search paginator template
searchPageLayoutTemplatePath	`String`	/exo:ecm/views/templates/WCM Advance Search/search-page-layout/UISearchPageLayoutDefault.gtmpl	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 used to display the search result

1.3.3.2. 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>
```

1.3.4. Explorer

1.3.4.1. 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 used. The workspace name was included in the Drive(!)
path	`String`	N/A	The path of node. This preference will be used

Preference	Type	Default value	Description
			when you choose the usecase is Parameterize
drive	`String`	N/A	Not in used. Replaced by driveName preference(!)
views	`String`	N/A	Not in used. The views will be displayed based on the Drive which user has access permission(!)
allowCreateFolders	`String`	N/A	Allow to create a folder by type. When you don't 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**
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 repository name where store the deleted nodes
trashWorkspace	`String`	collaboration	The workspace name where store the deleted nodes
editInNewWindow	`Boolean`	false	Allow to edit document with or without window popup. By default the value is false
showTopBar	`Boolean`	true	Allow to show the Top bar or not. By default the

Preference	Type	Default value	Description
			value is true
showActionBar	`Boolean`	true	Allow to show action bar or not. By default the value is true
showSideBar	`Boolean`	true	Allow to show the side bar or not. By default the value is true
showFilterBar	`Boolean`	true	Allow to show the Filter bar or not. By default the value is true

1.3.4.2. 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>
```

1.3.5. Administration

1.3.5.1. Available preferences

Preference	Type	Default	Description
repository	`String`	repository	The current repository name. Default value is "repository"

1.3.5.2. Sample Configuration

```
<portlet-preferences>
```

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

1.3.6. Fast Content Creator

1.3.6.1. Available preferences

Preference	Type	Default Value	Description
mode	`String`	basic	The default mode for fast content creator portlet
repository	`String`	repository	The current repository name. always is "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 show on the dialog form
saveButton	`String`	Save	The custom button Save
saveMessage	`String`	This node has been saved successfully	The custom message when click Save button
isRedirect	`Boolean`	false	Redirect to other page or not
redirectPath	`String`	http://www.google.com.vn	The page will redirect to
isActionNeeded	`Boolean`	true	Is an action needed to save the configuration

1.3.6.2. 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 2. Configuration

2.1. 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.

For shorter, 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 : can we see hidden nodes ? can we create folders in this drive ?

2.1.1. Fields

		org.exoplatform.services.cms.drives.DriveDa
name	`String`	Name of the drive, must be unique inside WCM
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
	example	`*:/platform/users`
icon	`String`	url to the icon
views	`String`	List of views you want to use
	example	`simple-view,admin-view`
viewPreferences	`Boolean`	"User Preference" icon will be visible if true
viewNonDocument	`Boolean`	Non document types will be visible

		org.exoplatform.services.cms.drives.DriveDa
		in the user view if true
viewSideBar	`Boolean`	Show/Hide the left bar (with navigation and filters)
showHiddenNode	`Boolean`	Hidden nodes will be visible if true
allowCreateFolders	`Boolean`	List of nodetype we can create as folders
	example	`nt:folder,nt:unstructured`
allowNodeTypesOnTree	`String`	Allows you to filter nodetypes in the navigation tree. Default value is "*" to show all content types
	example	`*`
		`exo:taxonomy`

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</o
    </init-params>
  </component-plugin>
</external-component-plugins>
```

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

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

2.1.2. 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>
```

2.2. Publication

2.2.1. component configuration

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

2.2.1.1. component-plugin configuration

Configuration name	Data type	Sample value	Description
name	String	Simple publication	Name of component-plugin

Configuration name	Data type	Sample value	Description
set-method	`String`	addPublicationPlugin	Set method to start using plugin
type	`String`	org.exoplatform.services.wcm.publication.lifecycle.simple.Simple	Class and location of plugin
description	`String`	...	Some description

2.2.1.2. 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 infomation will be the object's properties.

Configuration name	Data type	Sample value	Description
name	`String`	cache.config.wcm.composer	
description	`String`		
object	`Object`		This object will be passed to component as a parameter, this object includes fieldset properties

2.2.2. external-component-plugins configuration

Configuration name	Data type	Sample value	Description
target-component	`String`	org.exoplatform.services.wcm.core.WebSchemaConfigService<	External plugin's class location
component-plugin	`Object`		Configuration of component that will be used as a plugin

2.2.3. Example

2.2.3.1. 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>
```

2.2.3.2. 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>
```

2.2.3.3. 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>
```

2.3. Deployment

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

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

2.3.1. Fields

init-params

Element	Type	Default value	Description
object-param	`Object`	(see below)	The parameter which is an Object

object-param

Element	Type	Default value	object-param
name	`String`		The name of this object parameter
description	`String`		The description of this object parameter
object	`Class`	(see below)	The object of this object parameter

object

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

org.exoplatform.services.deployment.DeploymentDescriptor

Name	Type	Default value	Description
target	`Object`	org.exoplatform.services.deployment.DeploymentDescriptor$Target (*)	The target node (will contains the imported node)
sourcePath	`String`		The absolute path of the XML file

org.exoplatform.services.deployment.DeploymentDescriptor$Target

Field	Type	Default value	Description
repository	`String`		The repository of the target node
workspace	`String`		The workspace of the target node
nodePath	`String`		The path of the target node

2.3.2. 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>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

2.4. Taxonomies

Taxonomies are used to sort documents in order 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 from the created documents.

2.4.1. Fields

		org.exoplatform.services.cms.taxonomy.impl.
permissions	`java.util.ArrayList<TaxonomyTreeDefaultUserPermission>`	List of default user permission to access the Taxonomy Tree

		org.exoplatform.services.cms.taxonomy.impl.
identity	`String`	user name or user group.
read	`Boolean`	permission to read taxonomy tree.
addNode	`Boolean`	permission to add a node in taxonomy tree.
setProperty	`Boolean`	permission to set properties for node in taxonomy tree .
remove	`Boolean`	permission to remove node in taxonomy tree.

		org.exoplatform.services.cms.taxonomy.impl.
autoCreateInNewRepository	`Boolean`	Enable/ Disable the creation of the taxonomies in new created repository.
repository	`String`	name of the repository where taxonomies are created.
workspace	`String`	name of the workspace where taxonomies are created.
treeName	`String`	name of the taxonomy tree to be created.
permission.configuration	`org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig`	access permission for the whole taxonomy tree
predefined.actions	`org.exoplatform.services.cms.taxonomy.impl.ActionsConfig`	predefined actions for the

		org.exoplatform.services.cms.taxonomy.impl.
		taxonomy tree root node.
taxonomy.configuration	`org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig`	access permissions for each taxonomy in tree.

		org.exoplatform.services.cms.taxonomy.impl.
taxonomies	`java.util.ArrayList<TaxonomyConfig.Taxonomy>`	list of taxonomy to be configured permission.

		org.exoplatform.services.cms.taxonomy.impl.
name	`String`	name of the taxonomy.
path	`String`	path to the taxonomy in taxonomy tree.
permissions	`java.util.ArrayList<TaxonomyTreeListener.PermissionConfig.Permission>`	list of permission for user/group to access the taxonomy

		org.exoplatform.services.cms.actions.impl.Ac
actions	`java.util.ArrayList<ActionConfig.TaxonomyAction>`	list of actions which are created for the taxonomy tree root node.

		org.exoplatform.services.cms.actions.impl.Ac
type	`String`	type of the action.
name	`String`	name of the action.
description	`String`	description of the action.
homePath	`String`	location of node where the action node is stored.
targetWspace	`String`	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 workspace is specified in this field.
targetPath	`String`	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	`java.util.ArrayList<String>`	the life cycle phases, in which the

		org.exoplatform.services.cms.actions.impl.Ac
		action is activated.
roles	`String`	the users or groups only with whom the action is activated.
mixins	`java.util.ArrayList<ActionConfig.Mixin>`	list of node types that are affected by this action.

		org.exoplatform.services.cms.actions.impl.Ac
name	`String`	mixin node type name that will be added, whose responsibility is to store the affected node types.
properties	`String`	name of the properties that stores all the node types affected by the action.

2.4.2. Sample Configuration

```
<component>
  <key>org.exoplatform.services.cms.taxonomy.TaxonomyService</key>
  <type>org.exoplatform.services.cms.taxonomy.impl.TaxonomyServiceImpl</type>
  <init-params>
    <object-param>
      <name>defaultPermission.configuration</name>
      <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission">
        <field name="permissions">
          <collection type="java.util.ArrayList">
            <value>
              <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission$Permiss
                <field name="identity">
                  <string>*:/platform/administrators</string>
                </field>
                <field name="read">
                  <string>true</string>
                </field>
                <field name="addNode">
                  <string>true</string>
                </field>
                <field name="setProperty">
                  <string>true</string>
```

```
              </field>
              <field name="remove">
                <string>true</string>
              </field>
            </object>
          </value>
        </collection>
      </field>
    </object>
  </object-param>
</init-params>
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.taxonomy.TaxonomyService</target-component>
  <component-plugin>
    <name>predefinedTaxonomyPlugin</name>
    <set-method>addTaxonomyPlugin</set-method>
    <type>org.exoplatform.services.cms.taxonomy.impl.TaxonomyPlugin</type>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <value-param>
        <name>workspace</name>
        <value>dms-system</value>
      </value-param>
      <value-param>
        <name>treeName</name>
        <value>System</value>
      </value-param>
      <object-param>
        <name>permission.configuration</name>
        <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig">
          <field name="taxonomies">
            <collection type="java.util.ArrayList">
              <value>
```

```
            <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig$Taxonomy">
              <field name="permissions">
                <collection type="java.util.ArrayList">
                  <value>
                    <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig$Permission">
                      <field name="identity">
                        <string>*:/platform/users</string>
                      </field>
                      <field name="read">
                        <string>true</string>
                      </field>
                      <field name="addNode">
                        <string>true</string>
                      </field>
                      <field name="setProperty">
                        <string>true</string>
                      </field>
                      <field name="remove">
                        <string>false</string>
                      </field>
                    </object>
                  </value>
                </collection>
              </field>
            </object>
          </value>
        </collection>
      </field>
    </object>
  </object-param>
  <object-param>
    <name>predefined.actions</name>
    <description>description</description>
    <object type="org.exoplatform.services.cms.actions.impl.ActionConfig">
      <field name="actions">
        <collection type="java.util.ArrayList">
          <value>
            <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$TaxonomyAction">
              <field name="type">
```

```
      <string>exo:taxonomyAction</string>
    </field>
    <field name="name">
      <string>taxonomyAction</string>
    </field>
    <field name="description">
      <string/>
    </field>
    <field name="homePath">
      <string>dms-system:/exo:ecm/exo:taxonomyTrees/storage/System</string>
    </field>
    <field name="targetWspace">
      <string>collaboration</string>
    </field>
    <field name="targetPath">
      <string>/Documents</string>
    </field>
    <field name="lifecyclePhase">
      <collection type="java.util.ArrayList">
        <value>
          <string>node_added</string>
        </value>
      </collection>
    </field>
    <field name="roles">
      <string>*:/platform/administrators</string>
    </field>
    <field name="mixins">
      <collection type="java.util.ArrayList">
        <value>
          <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin">
            <field name="name">
              <string>mix:affectedNodeTypes</string>
            </field>
            <field name="properties">
              <string>exo:affectedNodeTypeNames=exo:article,exo:podcast,exo:sample,kfx:document,
            </field>
          </object>
        </value>
      </collection>
```

```
            </field>
          </object>
        </value>
      </collection>
    </field>
  </object>
</object-param>
<object-param>
  <name>taxonomy.configuration</name>
  <description>configuration predefined taxonomies to inject in jcr</description>
  <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig">
    <field name="taxonomies">
      <collection type="java.util.ArrayList">
        <!-- cms taxonomy -->
        <value>
          <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig$Taxonomy">
            <field name="name">
              <string>cmsTaxonomy</string>
            </field>
            <field name="path">
              <string>/cms</string>
            </field>
            <field name="permissions">
              <collection type="java.util.ArrayList">
                <value>
                  <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig$Permission">
                    <field name="identity">
                      <string>*:/platform/users</string>
                    </field>
                    <field name="read">
                      <string>true</string>
                    </field>
                    <field name="addNode">
                      <string>true</string>
                    </field>
                    <field name="setProperty">
                      <string>true</string>
                    </field>
                    <field name="remove">
```

```
                              <string>false</string>
                            </field>
                          </object>
                        </value>
                      </collection>
                    </field>
                  </object>
                </value>
              </collection>
            </field>
          </object>
        </object-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
</component>
```

2.5. Nodetypes

Every node has a type. A node's type The names, types and other attributes of its child items. Node types can be used to define complex storage objects consisting of multiple sub nodes and properties, possibly many layers deep.

2.5.1. Fields

		org.exoplatform.services.jcr.core.nodetype.E
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

2.5.2. 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>
```

2.6. Views

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

2.6.1. Fields

org.exoplatform.services.cms.views.ViewConfig

Name	Type	Description
name	`String`	view name, must be unique inside

Name	Type	Description
		WCM
permissions	`String`	Visibility of the view based on eXo Rights
	example	`*:/platform/administrators`
template	`String`	Specify path to the template location
tabList	`java.util.ArrayList`	include a set of view names.

org.exoplatform.services.cms.views.ViewConfig$Tab

Name	Type	Description
tabName	`String`	tab name that is not uniquely
button	`String`	These have to specify a set of view component names
	example	viewNodeType; viewPermissions; viewProperties; showJCRStructure

org.exoplatform.services.cms.views.TemplateConfig

Name	Type	Description
type	`String`	Specify to a name that is truly a class representing all properties of a view.
name	`String`	These have to specify a set of view component names
warPath	`String`	Specify to a template location for viewing
	example	/ecm-explorer/SystemView.gtmpl

2.6.2. 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>
```

2.7. Templates

2.7.1. Application template

A template is an definition of presentation while displaying the saved information

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

2.7.1.1. Available configuration

2.7.1.1.1. init-params

Configuration name	Data type	Default Value	Description
autoCreateInNewRepository	`Boolean`	true	Allow the application automatically import predefined templated at start up of template service
storedLocation	`String`	war:/conf/dms-extension/dms-artifacts/templates	Location of stored templates

Configuration name	Data type	Default Value	Description
repository	`String`	repository	Location of stored templates
object-param	`Structure`		Configuration for each instance.

2.7.1.1.2. object-param

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

Object type define all available template file, using the "collection type" configuration

2.7.1.1.3. object

type: name of each object type, it means the type of template, the further configurations for this type are define by the some specify fileds.

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 displaying of the title for this node

There are 3 further information related to presentation of each template, that is referencedView: Determine how to display to view. referencedDialog: Determine how to display a diaglog to input information. referencedSkin: Determine the style sheet for displaying.

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

Field name	Data type	Desciption
templateFile	String	The location of the file store for the template's presentation
roles	String	Determine who can access this

Field name	Data type	Desciption
		object (View/Dialog/CSS)

2.7.1.2. Example of configuration for an template

This bellow 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>
```

2.7.2. Nodetype template

2.8. Views

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

2.8.1. Fields

org.exoplatform.services.cms.views.ViewConfig

Name	Type	Description
name	`String`	view name, must be unique inside WCM
permissions	`String`	Visibility of the view based on eXo Rights
	example	`*:/platform/administrators`
template	`String`	Specify path to the template location
tabList	`java.util.ArrayList`	include a set of view names.

org.exoplatform.services.cms.views.ViewConfig$Tab

Name	Type	Description
tabName	`String`	tab name that is not uniquely
button	`String`	These have to specify a set of view component names
	example	viewNodeType; viewPermissions; viewProperties; showJCRStructure

org.exoplatform.services.cms.views.TemplateConfig

Name	Type	Description
type	`String`	Specify to a name that is truly a class representing all properties of a view.
name	`String`	These have to specify a set of view component names
warPath	`String`	Specify to a template location for viewing
	example	/ecm-explorer/SystemView.gtmpl

2.8.2. 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 3. Inside WCM Templates

3.1. 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.

3.1.1. 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.

3.1.1.1. 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)

3.1.1.2. Text Field

3.1.1.2.1. Addition parameters

None

See also: Common parameters

3.1.1.2.2. Example

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

3.1.1.3. Hidden Field

3.1.1.3.1. Addition parameters

None

See also: Common parameters

3.1.1.3.2. Example

3.1.1.4. Text Area Field

3.1.1.4.1. 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

3.1.1.4.2. Example

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

3.1.1.5. Rich Text Field

3.1.1.5.1. 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

3.1.1.5.2. Example

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

3.1.1.6. Calendar Field

3.1.1.6.1. Addition parameters

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

See also: Common parameters

3.1.1.6.2. Example

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

3.1.1.7. Upload Field

3.1.1.7.1. Addition parameters

None

See also: Common parameters

3.1.1.7.2. 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. What if you want to display the upload field if the image is blank, otherwise you display the image and an action allow you to remove ? You can do as follow

```
<%
        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) ;
        }
%>
```

3.1.1.8. Radio Field

3.1.1.8.1. Addition parameters

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

See also: Common parameters

3.1.1.8.2. Example

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

3.1.1.9. 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

3.1.1.9.1. Example

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

3.1.1.10. CheckBox Field

3.1.1.10.1. Addition parameters

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

See also: Common parameters

3.1.1.10.2. Example

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

3.1.1.11. Mixin Field

3.1.1.11.1. Addition parameters

None

See also: Common parameters

3.1.1.11.2. Example

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

3.1.1.12. Action Field

3.1.1.12.1. 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` don't need any special parameters

See also: Common parameters

3.1.1.12.2. Example

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

3.1.1.13. 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`

3.1.1.13.1. Example

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

3.1.1.14. This is an instruction that explains how to add new ECM template with tabs.

To avoid refresh to first tab every action execution, we add new private function to the template with tabs In the template we have to 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, in order to display selected tab, we'll add into style of UITabContent class

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

3.1.2. 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")%>
```

3.2. List of Contents

3.2.1. Folder based Template

3.2.2. Category tree Template

Chapter 4. Inside WCM Explorer

4.1. CSS

- By using WCM, all the stylesheet of each site can be managed easily and online. You don't need to access to the file system to modify and wait for server restarted. About the structure, each site have its own CSS folder, this folder can contains 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 be applied in to the site.

- For example: By default, a WCM demo package have 2 sites ACME and Classic. Classic site has a CSS folder which contains a CSS file called **DefaultStylesheet**. Most of the stylesheet of this site is defined within this stylesheet. Moreover, ACME site has 2 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 disable the blue one (by edit it and set Available equals false) and enable the green one. Now back to the homepage and you will see the magic.

- Note: Please remember clear the cache and refresh the browser first if you don't see the change. Most of the time, this is the main reason the new style is not applied.

4.2. Javascript

4.3. CKEditor

Basically, if you want to add a rich text area into your dialogs, you can use addRichtextField method. But sometimes, you want to add the rich text editor by yourself manually. So first you need to use addTextAreaField method and some addition javascript like this

```
<%
        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>
```

Chapter 5. Extensions

5.1. REST Services

- 1. Rest Overview
- 2. Restful Web Service
  - 2.1 HTTP Method
  - 2.2 Formats
  - 2.3 Rest configuration
  - 2.4 Create a rest service

5.1.1.1. 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 next time the client chooses to initiate a new state transition.

REST was 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 top of it.

5.1.1.2. Restful Web Service

5.1.1.2.1. 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

Method	Definition
DELETE	Delete a Resource

5.1.1.2.2. 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

5.1.1.2.3. Parameters common to every APIs

5.1.1.2.4. 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.

5.1.1.2.5. 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>
```

5.1.1.2.6. 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/")	Specified the URI path that a resource or class

Parameters	Definition
	method will serve requests for.
@PathParam("repository")	Binds 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")	Binds the value **showItems** of a HTTP query parameter to a resource method parameter, resource class field, or resource class bean property.

5.2. UI Extensions

This article's goal to describe how to create an sample of ui extension.

- 1. Overview
- 2. How to add your own tab in ECM Administration
  - 2.1 Add your own UIAction
  - 2.2 Add your action listener
  - 2.3 Deploy your application
  - 2.4 Define label for your actions and register resource bundle
  - 2.5 Run your own ui extension sample

5.2.1.1. 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**.

5.2.1.2. How to add your own tab in ECM Administration

5.2.1.3. Add you 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>2.2.5-GA</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>2.1.1.1</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.ecms</groupId>
      <artifactId>exo-ecms-core-webui-administration</artifactId>
      <version>2.1.1.1</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**

5.2.1.4. 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 follow (ACTION*NAME)****ActionListener*** *Example :* You create your own action listener name **CacheView**, so your action listener then will be named as **CacheViewActionListener**

first thing we will do, add a static inner class called **CacheViewActionListener** that extends **org.exoplatform.ecm.webui.component.admin.listener.UIECMAdminControlPanelActionListener**

See below the expected code

```
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 with the service *org.exoplatform.webui.ext.UIExtensionManager*, as below:

5.2.1.5. 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

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

> **Note**
>
> All resources can be located in the src/main/resource package because we will seperate the resources(**.xml, images, conf file) and the code. This is very useful in hierarchical structure.**

Create ExamplePortlet*en.xml with 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>
```

We have to 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 our resource bundle from the original system, this is so clearly useful, we got 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>
```

5.2.1.7. Run your own ui extension sample

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

We've already add a new extension to ECM Administrator

Chapter 6. Java Services

6.1. TaxonomyService

TaxonomyService is used for working with taxonomies. In this service there are many functions which allow you to add, find, delete taxonomies from 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;	Removes 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;`	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 @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;	Initial all taxonomy plugins that have been already configured in xml files @param repository The name of repository @see TaxonomyPlugin @throws Exception

6.2. LinkManager

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

Method	Return	Prototype	Description
			exo:symlink, the default value is "exo:symlink" @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

Method	Return	Prototype	Description
			exo:symlink, the default value is "exo:symlink" @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

Method	Return	Prototype	Description
			session in case we cannot use the same 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`		

Method	Return	Prototype	Description

		`isTargetReachable(Node link)`
		
		`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
			
			@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

Method	Return	Prototype	Description
			@throws RepositoryException if an unexpected error 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

6.3. 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`	Get a specific context with the	

Method	Return	Prototype	Description	
		`name) ;`	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 fromstate, String tostate, String date, String user, String lang , String workspace) throws Exception;`	Get all the node @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	q

6.4. WCMComposer

This class is responsible of getting contents inside the WCM product. We shouldn't access directly contents from the jcr on front side.

In a general manner, 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
			
			@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

Method	Return	Prototype	Description
updateContent	boolean	updateContent(String repository, String workspace, String nodeIdentifier, HashMap<String, String> filters) throws Exception;	Update content. repository the repository @param workspace the workspace @param path the path @param filters the filters @return true, if successful
updateContents	boolean	updateContents(String repository, String workspace, String path, HashMap<String, String> filters) throws Exception;	Update contents. @param repository the repository @param workspace the 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

Method	Return	Prototype	Description
			@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

6.5. CMS

CmsService is used for working with nodes. In this service there are many functions which allow you to add, edit, move node.

Package org.exoplatform.services.cms.CmsService;

Method	Return	Prototype	Desription
storeNode	String	storeNode(String workspace, String nodetypeName, String storePath, Map inputProperties,String repository) throws Exception;	Returns path to saved node @param workspace: name of workspace @param nodetypeName: NodeType's name @param storePath: Path to store node @param inputProperties:

Method	Return	Prototype	Desription
			Map of node's property including (property name, value) @param repository: Repository's name @throws Exception: throws exception
storeNode	String	storeNode(String nodetypeName, Node storeHomeNode, Map inputProperties, boolean isAddNew,String repository) throws Exception;	Returns path to saved node @param nodetypeName: NodeType's name @param storeHomeNode: Parent node, where node is stored @param inputProperties: Map of node's property including (property name, value) @param isAddNew: flag to decide whether this situation is adding node or updating node @param repository: Repository's name @throws Exception: throws exception
storeEditedNode	String	storeEditedNode(String nodetypeName, Node storeNode, Map inputProperties, boolean isAddNew,String repository) throws Exception;	Returns path to saved node @param nodetypeName: NodeType's name @param storeNode: Node

Method	Return	Prototype	Desription
			is stored @param inputProperties: Map of node's property including (property name, value) @param isAddNew: flag to decide whether this situation is adding node or updating node @param repository: Repository's name @throws Exception: throws exception
storeNodeByUUID	String	storeNodeByUUID(String nodetypeName, Node storeNode, Map inputProperties, boolean isAddNew,String repository) throws Exception;	Returns return UUID of saved node @param nodetypeName: NodeType's name @param storeNode: Node is stored @param inputProperties: Map of node's property including (property name, value) @param isAddNew: flag to decide whether this situation is adding node or updating node @param repository: Repository's name @throws Exception: throws exception
moveNode	void	moveNode(String nodePath, String	@param nodePath: Path

Method	Return	Prototype	Desription
		srcWorkspace, String destWorkspace, String destPath, String repository)	to node in source workspace @param srcWorkspace: Source workspace name @param destWorkspace: Destination of workspace name @param destPath: Destination of node path @param repository: Repository's name @throws Exception: throws exception

6.6. ActionServiceContainer

This service is used to manage DMS actions (with nodetype `exo:action`)

Package org.exoplatform.services.cms.actions.ActionServiceContainer;

Method	Return	Prototype	Description
`getActionPluginNames`	`Collection<String>`	`getActionPluginNames()`	Collection of String @return collection of ActionPlugin names
`getActionPlugin`	`ActionPlugin`	`getActionPlugin(String actionServiceName)`	Get ActionPlugin following ActionSeriveName @param actionServiceName name of action service @return ActionPlugin

Method	Return	Prototype	Description
getActionPluginForActionType	ActionPlugin	getActionPluginForActionType(String actionTypeName)	Get ActionPlugin following action type name @param actionTypeName name of action type @return ActionPlugin
createActionType	void	createActionType(String actionTypeName, String parentActionTypeName, String executable, List<String> variableNames, boolean isMoveType, String repository) throws Exception	Create NodaeTypeValue is in kind of ActionType following action type name @param actionTypeName name of action type @param parentActionTypeName name of parent action @param executable String value of execuable @param variableNames List name of variable @param isMoveType is moved or not @param repository repository name @throws Exception
getCreatedActionTypes	Collection<NodeType>	getCreatedActionTypes(String repository) throws Exception	Get all created node with nodetype = exo:action @param repository repository name

Method	Return	Prototype	Description
			@return Collection of NodeType @throws Exception
getAction	**Node**	`getAction(Node node, String actionName) throws Exception`	get node by using actionName as relative path with current node @param node current processing node @param actionName name of action @return Node @throws Exception
hasActions	`boolean`	`hasActions(Node node) throws Exception`	Check node type is exo:actionable or not @param node @return true: NodeType is exo:actionable false NodeType is not exo:actionable @throws Exception
getActions	`List<Node>`	`getActions(Node node) throws Exception`	Get list of child node with NodeType = exo:action @param node current node @return list of node @throws Exception
getCustomActionsNode	`List<Node>`	`getCustomActionsNode(Node node, String lifecyclePhase)`	Get list of node that have same level with current

Method	Return	Prototype	Description
		`throws Exception`	node, exo:lifecyclePhase = lifecyclePhase @param node current node @param lifecyclePhase exo:lifecyclePhase value @return list of node @throws Exception
`getActions`	`List<Node>`	`getActions(Node node, String lifecyclePhase) throws Exception`	Get list of child node with exo:lifecyclePhase = lifecyclePhase @param node current node @param lifecyclePhase exo:lifecyclePhase value @return list of node @throws Exception
`removeAction`	`void`	`removeAction(Node node, String repository) throws Exception`	Remove all action registered in node @param node @param repository @throws Exception
`removeAction`	`void`	`removeAction(Node node, String actionName, String repository) throws Exception`	Remove all relative node of current node with node type = exo:actionable @param node current node

Method	Return	Prototype	Description
			@param actionName relative path = exo:actionable / actionName @param repository repository name @throws Exception
addAction	void	addAction(Node node, String repository, String type, Map mappings) throws Exception	Add mixintype = exo:actionable to current node Add new node to current node with nodetype = type @param node current node @param repository current repository @param type nodetype name @param mappings value of property for adding to new node @throws Exception
addAction	void	addAction(Node node, String repository, String type, boolean isDeep, String uuid, String nodeTypeNames, Map mappings) throws Exception	Add mixintype = exo:actionable to current node Add new node to current node with nodetype = type @param node current node

Method	Return	Prototype	Description
			@param repository current repository @param type nodetype name @param isDeep affect to child node of node @param uuid affect only to parent node of event having given uuid @param nodeTypeNames affect to parent node of event having nodetype in nodeTypeNames @param mappings value of property for adding to new node @throws Exception
executeAction	void	executeAction(String userId, Node node, String actionName, Map variables, String repository) throws Exception	Execute action following userId, node, variables, repository @param userId user identify @param node current node @param actionName name of action @param variables Map with variables and value @param repository current repository

Method	Return	Prototype	Description
			@throws Exception
executeAction	void	executeAction(String userId, Node node, String actionName, String repository) throws Exception	Execute action following userId, node, repository, initiated variables @param userId user identify @param node current node @param actionName name of action @param repository current repository @throws Exception @see {@link #executeAction(String, Node, String, Map, String)}
initiateObservation	void	initiateObservation(Node node, String repository) throws Exception	Add action listener for all action child node of current node in repository @param node current node @param repository Repository name @throws Exception
init	void	init(String repository) throws Exception	init service with repository name @param repository repository name

Method	Return	Prototype	Description
			@throws Exception

6.7. Records

Package org.exoplatform.services.cms.records.RecordService;

Method	Return	Prototype	Description
`bindFilePlanAction`	`void`	`bindFilePlanAction(Node filePlan, String repository) throws Exception;`	Add action for filePlan node in repository @param filePlan Node to process @param repository Repository name @throws Exception
`addRecord`	`void`	`addRecord(Node filePlan, Node record) throws RepositoryException;`	Set property for filePlan node which is get from record node @param filePlan filePlan Node @param record record Node @throws RepositoryException
`computeCutoffs`	`void`	`computeCutoffs(Node filePlan) throws RepositoryException;`	Determine if the next phase is a hold, transfer or destruction @param filePlan @throws RepositoryException
`computeHolds`	`void`	`computeHolds(Node`	

Method	Return	Prototype	Description
		`filePlan) throws RepositoryException;`	Process transfer or destruction @param filePlan filePlan node@throws RepositoryException
`computeTransfers`	`void`	`computeTransfers(Node filePlan) throws RepositoryException;`	Copy record node in filePlan node to path which value is rma:transferLocation property of filePlan Node @param filePlan @throws RepositoryException
`computeAccessions`	`void`	`computeAccessions(Node filePlan) throws RepositoryException;`	Copy record node in filePlan node to path which value is rma:accessionLocation property of filePlan Node @param filePlan @throws RepositoryException
`computeDestructions`	`void`	`computeDestructions(Node filePlan) throws RepositoryException;`	Remove record node in filePlan node @param filePlan filePlan node @throws RepositoryException
`getRecords`	`List<Node>`	`getRecords(Node filePlan) throws RepositoryException;`	Get list of node by query statement

Method	Return	Prototype	Description
			@param filePlan filePlan node @throws RepositoryException
`getVitalRecords`	`List<Node>`	`getVitalRecords(Node filePlan) throws RepositoryException;`	Get list of node by query statement with constraint concerning @rma:vitalRecord @param filePlan filePlan node @throws RepositoryException
`getObsoleteRecords`	`List<Node>`	`getObsoleteRecords(Node filePlan) throws RepositoryException;`	Get list of node by query statement with constraint concerning @rma:isObsolete @param filePlan filePlan node @throws RepositoryException
`getSupersededRecords`	`List<Node>`	`getSupersededRecords(Node filePlan) throws RepositoryException;`	Get list of node by query statement with constraint concerning @rma:superseded @param filePlan filePlan node @throws RepositoryException
`getCutoffRecords`	`List<Node>`	`getCutoffRecords(Node filePlan) throws`	Get list of node by query

Method	Return	Prototype	Description
		`RepositoryException;`	statement with constraint concerning @rma:cutoffExecuted @param filePlan filePlan node @throws RepositoryException
`getHolableRecords`	`List<Node>`	`getHolableRecords(Node filePlan) throws RepositoryException;`	Get list of node by query statement with constraint concerning @rma:holdExecuted @param filePlan filePlan node @throws RepositoryException
`getTransferableRecords`	`List<Node>`	`getTransferableRecords(Node filePlan) throws RepositoryException;`	Get list of node by query statement with constraint concerning @rma:transferExecuted @param filePlan filePlan node @throws RepositoryException
`getAccessionableRecords`	`List<Node>`	`getAccessionableRecords(Node filePlan) throws RepositoryException;`	Get list of node by query statement with constraint concerning @rma:accessionExecuted @param filePlan filePlan node @throws

Method	Return	Prototype	Description
			RepositoryException
`getDestroyableRecords`	`List<Node>`	`getDestroyableRecords(Node filePlan) throws RepositoryException;`	Get list of rma:destroyable node by query statement @param filePlan filePlan node @param filePlan filePlan node @throws RepositoryException

6.8. MultiLanguage

MultiLanguageService is used for working with language of node. In this service there are many functions which allow you to add, edit, delete language of node.

Method	Return	Prototype	Desription
getSupportedLanguages	List<String>	getSupportedLanguages(Node node) throws Exception	Returns value of exo:language property of the node @param node: The current node @throws Exception: throws exception
setDefault	void	setDefault(Node node, String language, String repositoryName) throws Exception	Set data for current node @param node: The current node @param language: The language name @param repositoryName: The name of repository @throws Exception:

Method	Return	Prototype	Desription
			throws exception
addLanguage	void	addLanguage(Node node, Map inputs, String language, boolean isDefault) throws Exception	Add new language for current node @param node: The current node @param inputs: Map includes key and value of property @param language: The name of language @param isDefault: flag to define default language is used or not @throws Exception: throws exception
addLanguage	void	addLanguage(Node node, Map inputs, String language, boolean isDefault, String nodeType) throws Exception	Add new language for current node @param node: The current node @param inputs: Map includes key and value of property @param language: The name of language @param isDefault: flag to define default language is used or not @param nodeType: The name of NodeType @throws Exception:

Method	Return	Prototype	Desription
			throws exception
addFileLanguage	void	addFileLanguage(Node node, String fileName, Value value, String mimeType, String language, String repositoryName, boolean isDefault) throws Exception	Add newLanguageNode node, then add new file to newLanguageNode @param node: The current node @param: fileName: The name of file @param value: The value of file @param mimeType: mimiType @param language: The name of language @param repositoryName: The name of repository @param isDefault: flag to use new language or default language @throws Exception: throws exception
addFileLanguage	void	addFileLanguage(Node node, String language, Map mappings, boolean isDefault) throws Exception	Add newLanguageNode node, then set property in mapping to newLanguageNode @param node: The current node @param language: The name of language @param mappings: Map

Method	Return	Prototype	Desription
			includes property and value @param isDefault: flag to use new language or default language @throws Exception: throws exception
addLinkedLanguage	void	addLinkedLanguage(Node node, Node translationNode)	Add newLanguageNode node with a symlink, based on exo:language targetNode property @param node: The current node @param translationNode: target translation node @throws Exception: throws exception
addFolderLanguage	void	addFolderLanguage(Node node, Map inputs, String language, boolean isDefault, String nodeType, String repositoryName) throws Exception	Add new language node as a folder @param node: The current node @param inputs: Map includes key and value of property @param language: The name of language @param isDefault: flag to define default language is used or not @param nodeType: The name of nodeType

Method	Return	Prototype	Desription
			@throws Exception: throws exception
getDefault	String	getDefault(Node node)	Returns Get value of property exo:language in current node @param node: The current node @throws Exception: throws exception
getLanguage	String	getLanguage(Node node, String language) throws Exception	Returns Get node following relative path = "languages/" + language @param node: The current node @param language: The name of language @throws Exception: throws exception

6.9. ManageDrive

This service is used to manage DMS drives

Package org.exoplatform.services.cms.drives.DriveData;

Method	Return	Prototype	Description
addDrive	void	addDrive(String name, String workspace, String permissions, String homePath, String views, String icon, boolean viewReferences, boolean viewNonDocument, boolean viewSideBar,	Register a new drive to workspace or update if the drive is existing @param name drive name @param workspace the workspace name where

Method	Return	Prototype	Description
		`boolean showHiddenNode, String repository, String allowCreateFolder, String allowNodeTypesOnTree)throws Exception`	will store the drive @param permissions specify who can access to this drive @param homePath specify the location of drive @param views include all views can see in drive @param icon the drive icon which can see in drive browser @param viewReferences the boolean to set default for drive can view references node or not @param viewNonDocument the boolean to set default for drive can view non document node or not @param viewSideBar the boolean to set default for drive can view side bar or not @param showHiddenNode the boolean to set default for drive can see hidden node or not @param repository the string contain repository name @param allowCreateFolder the

Method	Return	Prototype	Description
			string to specify which type of folder can add in the drive @throws Exception
getDriveByName	`DriveData`	`getDriveByName(String driveName, String repository) throws Exception`	Return an DriveData Object @param driveName the string contain the drive name @param repository the repository name @see DriveData @return DriveData with specified drive name and repository @throws Exception
getAllDriveByPermission	`List<DriveData>`	`getAllDriveByPermission(String permission, String repository) throws Exception`	Return the list of DriveData This method will look up in all workspaces of repository to find DriveData with specified permission @param permission the string contain the permission @param repository name of repository @return list of DriveData with specified repository and permission

Method	Return	Prototype	Description
			@see DriveData @throws Exception
removeDrive	void	removeDrive(String driveName, String repository) throws Exception	Remove drive with specified drive name and repository @param driveName drive name @param repository repository name @throws Exception
getAllDrives	List<DriveData>	getAllDrives(String repository) throws Exception	This method will look up in all workspaces of repository to find DriveData @param repository repository name @return list of DriveData with specified repository @throws Exception
isUsedView	boolean	isUsedView(String viewName, String repository) throws Exception	This method will check to make sure the view is not in used before remove this view @param viewName view name @param repository repository name @return the status of current view is in used or not

Method	Return	Prototype	Description
			@throws Exception
init	void	init(String repository) throws Exception	Register all drive plugins to repository @param repository the string contain repository name @throws Exception
getDriveByUserRoles	List<DriveData>	getDriveByUserRoles(String repository, String userId, List<String> roles) throws Exception	Get all drives by user roles @param repository Repository name @param userId User name @param roles Roles of user @return List<DriveData> @throws Exception
getMainDrives	List<DriveData>	getMainDrives(String repository, String userId, List<String> userRoles) throws Exception	Get all main drives @param repository Repository name @param userId Name of user @param userRoles Roles of user @return List<DriveData> @throws Exception
getPersonalDrives	List<DriveData>	getPersonalDrives(String	

Method	Return	Prototype	Description
		`repository, String userId, List<String> userRoles) throws Exception`	Get all personal drives @param repository Repository name @param userId Name of user @param userRoles Roles of user @return List<DriveData> @throws Exception
`getGroupDrives`	`List<DriveData>`	`getGroupDrives(String repository, String userId, List<String> userRoles, List<String> groups) throws Exception`	Get all group drives @param repository Repository name @param userId Name of user @param userRoles Roles of user @param groups Groups of user @return List<DriveData> @throws Exception

6.10. NewFolksonomy

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

Method	Return	Prototype	Description
`addPrivateTag`	`void`	`addPrivateTag(String`	Add a private tag to a

Method	Return	Prototype	Description
		`tagsName, Node documentNode, String repository, String workspace, String userName) throws Exception ;`	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 create a folksonomy link to 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 create a folksonomy link to 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 create a folksonomy link to 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 create a folksonomy link to 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 @return List<Node>

Method	Return	Prototype	Description
			@throws Exception
`getAllDocumentsByTag`	`List<Node>`	`getAllDocumentsByTag(String tagPath, String repository, String workspace, SessionProvider sessionProvider) throws Exception ;`	Get all document which storing in 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 repository, String workspace) throws Exception ;`	Update property TAG*RATE*PROP, HTML*STYLE*PROP following value tagRate, htmlStyle for node in tagPath in

Method	Return	Prototype	Description
			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 property TAG*RATE*PROP, HTML*STYLE*PROP following 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 @param workspace Workspace name

Method	Return	Prototype	Description
			@throws Exception
getAllTagStyle	List<Node>	getAllTagStyle(String repository, String workspace) throws Exception ;	Get all tag style base 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 ;	Init 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 tag of given document @param treeName Name of folksonomy tree @param tagName Name of tag @param document Document which added a link to tagName @param repository Repository name @return @throws Exception
removeTag	void	removeTag(String	

Method	Return	Prototype	Description
		`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 @param workspace Workspace name @return

Method	Return	Prototype	Description
			@throws Exception
getLinkedTagsOfDocumentByScope	List<Node>	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 tagPermissionList @param usersOrGroups
getTagPermissionList	List<String>	getTagPermissionList();	Returns tagPermissionList

Method	Return	Prototype	Description
canEditTag	boolean	canEditTag(int scope, List<String> memberships);	Can edit tag or not? @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 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 possible

6.11. Script

This service is used to manage groovy script resources

Package org.exoplatform.services.cms.scripts.ScriptService;

Method	Return	Prototype	Description
getECMScriptHome	Node	getECMScriptHome(String repository,SessionProvider provider) throws Exception	This method will get node for ECM Explorer Scripts by giving the following params

Method	Return	Prototype	Description
			@param repository The name of repository @param provider @see Node @see NodeHierarchyCreator @see SessionProvider @return Node @throws Exception
getCBScriptHome	`Node`	`getCBScriptHome(String repository,SessionProvider provider) throws Exception`	This method will get node for Content Browser Scripts by giving the following params @param repository The name of reporsitory @param provider @see Node @see NodeHierarchyCreator @see SessionProvider @return Node @throws Exception
getECMActionScripts	`List<Node>`	`getECMActionScripts(String repository,SessionProvider provider) throws Exception`	This method will get all node for ECM Action Scripts by giving the following params

Method	Return	Prototype	Description
			@param repository The name of reporsitory @param provider @see Node @see NodeHierarchyCreator @see SessionProvider @return Node @throws Exception
getECMInterceptorScripts	List<Node>	getECMInterceptorScripts(String repository,SessionProvider provider) throws Exception	This method will get all node for ECM Interceptor Scripts by giving the following params @param repository The name of reporsitory @param provider @see Node @see NodeHierarchyCreator @see SessionProvider @return List<Node> @throws Exception
getECMWidgetScripts	List<Node>	getECMWidgetScripts(String repository,SessionProvider provider) throws Exception	This method will get all node for ECM Widget Scripts by giving the following params

Method	Return	Prototype	Description
			@param repository The name of reporsitory @param provider @see Node @see NodeHierarchyCreator @see SessionProvider @return List<Node> @throws Exception
getScript	`CmsScript`	`getScript(String scriptPath, String repository) throws Exception`	This method will get script by giving the following params @param scriptPath The path of script @param repository The name of repository @see CmsScript @return CmsScript @throws Exception
getBaseScriptPath	`String`	`getBaseScriptPath() throws Exception`	This method will get base path of script @see NodeHierarchyCreator @return String@throws Exception
addScript	`void`	`addScript(String name, String text,`	This method will add

Method	Return	Prototype	Description
		`String repository,SessionProvider provider) throws Exception`	script by giving the following params @param name The name of script @param text @param repository The name of repository @param provider @see InputStream @see Node @see SessionProvider @throws Exception
`removeScript`	`void`	`removeScript(String scriptPath, String repository, SessionProvider provider) throws Exception`	This method will remove script by giving the following params @param scriptPath The path of script @param repository The name of repository @param provider @see Node @see SessionProvider @throws Exception
`getScriptNode`	`Node`	`getScriptNode(String scriptName, String repository,SessionProvider`	This method will get script node by giving the

Method	Return	Prototype	Description
		`provider) throws Exception`	following params @param scriptName The name of script @param repository The name of repository @param provider SessionProvider @see Node @see SessionProvider @return Node @throws Exception
`initRepo`	`void`	`initRepo(String repository) throws Exception`	This method will init repository by giving the following params @param repository The name of repository @see ManageableRepository @see ObservationManager @see Session @throws Exception

6.12. Relations

This service is used to manage document relationship

Package org.exoplatform.services.cms.relations.RelationsService;

Method	Return	Prototype	Description
hasRelations	boolean	hasRelations(Node node) throws Exception	Returns true is the given node has relation @param node Specify the node wants to check relation @see Node @throws Exception
getRelations	List<Node>	getRelations(Node node, String repository, SessionProvider provider) throws Exception	Gets all node that has relation to the given node @param node Specify the node wants to get all node relative to it @param repository The name of repository @param provider The SessionProvider object is used to managed Sessions @see Node @see SessionProvider @throws Exception
removeRelation	void	removeRelation(Node node, String relationPath, String repository) throws Exception	Removes the relation to the given node by specified the relationPath params @param node Specify the node wants to remove a relation @param relationPath The path of relation

Method	Return	Prototype	Description
			@param repository The name of repository @see Node @throws Exception
addRelation	void	addRelation(Node node, String relationPath, String workspaceName, String repository) throws Exception	Inserts a new relation is specified by relationPath params to the given node @param node Specify the node wants to insert a new relation @param relationPath The path of relation @param workspaceName The name of workspace @param repository The name of repository @see Node @throws Exception
init	void	init(String repository) throws Exception	Initial the root of relation node and its sub node @param repository The name of repository @throws Exception

6.13. RSS

RSSService is used for working with generate RSS. In this service there are some functions.

Method	Return	Prototype	Desription
storeNode	String	generateFeed(Map context)	Create a Feed file (feed type is RSS or Podcast) @param context: Map Consist of among information @throws Exception: throws exception

6.14. Voting

This service is used to manage the vote feature in WCM

Package org.exoplatform.services.cms.voting.VotingService;

Method	Return	Prototype	Description
`vote`	`void`	`vote(Node document, double rate, String userName, String language) throws Exception`	Voting the document is specified by the node by giving the rate, username, and language params. Any language belongs to this document can be voted. This method uses variables to store values which are voted from user for all kind languages of this document @param document The node document for voting @param rate The number rate for voting @param userName The username of current user is voting. @param language The language of this

Method	Return	Prototype	Description
			document for voting @see Node @throws Exception
getVoteTotal	long	getVoteTotal(Node node) throws Exception	Gets total voting for all kind languages of this document is specified by node @param node The node document is specified to get total voting @see Node @return @throws Exception
isVoted	boolean	isVoted(Node node, String userName, String language) throws Exception	Check the document is voted or not @param node The document @param userName The voted user @param language the voted language

6.15. Comments

CommentsService is used for working with comment node. In this service there are some functions which allow you to add, edit, delete, get comment.

Package org.exoplatform.services.cms.comments.CommentsService;

Method	Return	Prototype	Desription
addComment	void	addComment(Node document, String commentor, String email, String site, String comment, String language) throws Exception	Comment the document is specified by the node by giving the commentor, email, site, comment and language params @param document: The node document is commented @param commentor: The name of current user @param email: The email of current user @param site: The site of current user @param comment: The comment's content @param language: The language of this document is commented @throws Exception: throws exception
updateComment	void	updateComment(Node commentNode, String newComment) throws Exception	Update comment for document: set new comment for node @param commentNode: The node document is commented @param newComment: The comment's content @throws Exception: throws exception

Method	Return	Prototype	Desription
deleteComment	void	deleteComment(Node commentNode) throws Exception	Delete comment of document by given comment node @param commentNode: The node document is commented @throws Exception: throws exception
getComments	List<Node>	getComments(Node document, String language)	Gets all comments from the specified node @param commentNode: The node document is commented @param language: The name of language @throws Exception: throws exception

6.16. Query

This service is used to manage the search feature in WCM back-end

Package org.exoplatform.services.cms.queries.QueryService;

Method	Return	Prototype	Description
`getRelativePath`	`String`	`getRelativePath()`	Get the relative path
`getQueries`	`List<Query>`	`getQueries(String userName, String repository, SessionProvider provider) throws Exception`	Get queries by giving the following params @param userName String Can be <code>null</code> @param repository String

Method	Return	Prototype	Description
			The name of repository @param provider SessionProvider @return queries List<Query> @see Query @see SessionProvider @throws Exception
execute	`QueryResult`	`execute(String queryPath, String workspace, String repository, SessionProvider provider, String userId) throws Exception`	Execute query by giving the following params @param queryPath String The path of query @param workspace String The name of workspace @param repository String The name of repository @param provider SessionProvider @param userId String The id of current user @return queries QueryResult @see QueryResult @see SessionProvider @throws Exception
addQuery	`void`	`addQuery(String`	

Method	Return	Prototype	Description
		`queryName, String statement, String language, String userName, String repository) throws Exception`	Add new query by giving the following params @param queryName String The name of query @param statement String The statement query @param language String The language is requested @param userName String Can be <code>null</code> @param repository String The name of repository @throws Exception
`removeQuery`	`void`	`removeQuery(String queryPath, String userName, String repository) throws Exception`	Remove query by giving the following params @param queryPath String The path of query param userName String Can be <code>null</code> @param repository String The name of repository @throws Exception
`addSharedQuery`	`void`	`addSharedQuery(String queryName, String statement, String language, String permissions, boolean cachedResult, String repository) throws Exception`	Add new shared query by giving the following params @param queryName String The name of query

Method	Return	Prototype	Description
			@param statement String The statement query @param language String The language is requested @param permissions String @param cachedResult boolean Choosen for caching results @param repository String The name of repository @throws Exception
addSharedQuery	void	addSharedQuery(String queryName, String statement, String language, String permissions, boolean cachedResult, String repository, SessionProvider provider) throws Exception	Add new shared query by giving the following params @param queryName String The name of query @param statement String The statement query @param language String The language is requested @param permissions String @param cachedResult boolean Choosen for caching results @param repository String The name of repository @param provider Session provider

Method	Return	Prototype	Description
			@throws Exception
getSharedQuery	Node	getSharedQuery(String queryName, String repository, SessionProvider provider) throws Exception	Get shared queries by giving the following params @param userId String The id of current user @param repository String The name of repository @param provider SessionProvider @return sharedQueries List<Node> @see Node @see SessionProvider @throws Exception
removeSharedQuery	void	removeSharedQuery(String queryName, String repository) throws Exception	Remove share query by giving the following params @param queryName String The name of query @param repository String The name of repository @throws Exception
getSharedQueries	List<Node>	getSharedQueries(String repository, SessionProvider provider) throws Exception	Get shared queries by giving the following params @param repository String The name of repository

Method	Return	Prototype	Description
			@param provider SessionProvider @return sharedQueries List<Node> @see Node @see SessionProvider @throws Exception
getQueryByPath	`Query`	`getQueryByPath(String queryPath, String userName, String repository, SessionProvider provider) throws Exception`	Get query with path by giving the following params @param queryPath String The path of query @param userName String The name of current user @param repository String The name of repository @param provider SessionProvider @return query Query @see Node @see Query @see SessionProvider @throws Exception
getSharedQueries	`List<Node>`	`getSharedQueries(String userId, String repository, SessionProvider provider) throws`	Get shared queries by giving the following params

Method	Return	Prototype	Description
		`Exception`	@param userId String The id of current user @param repository String The name of repository @param provider SessionProvider @return sharedQueries List<Node> @see Node @see SessionProvider @throws Exception
`getSharedQueries`	`List<Node>`	`getSharedQueries(String queryType, String userId, String repository, SessionProvider provider) throws Exception`	Get shared queries by giving the following params @param queryType String The type of query @param userId String The id of current user @param repository String The name of repository @param provider SessionProvider @return sharedQueries List<Node> @see Node @see SessionProvider @throws Exception

Method	Return	Prototype	Description
init	void	init(String repository) throws Exception	Init all query plugin by giving the following params @param repository String The name of repository @see QueryPlugin @throws Exception

6.17. PageMetadata

This service is used to manage the page's metadata.

Package org.exoplatform.services.wcm.metadata.PageMetaDataService;

Method	Return	Prototype	Description
extractMetadata	HashMap	extractMetadata(Node node) throws Exception	Extract metadata information from node. @param node the node @return the hash map< string, string> @throws Exception the exception
getPortalMetadata	HashMap	getPortalMetadata(SessionProvider sessionProvider, String uri) throws Exception	Retrieves the portal metadata information for each request uri. @param uri the uri @param sessionProvider the session provider @return the portal

Method	Return	Prototype	Description
			metadata @throws Exception the exception

6.18. ManageView

ManageViewService is used for working with views. In this service there are many functions which allow you to add, edit, delete, get views.

Method	Return	Prototype	Desription
addView	void	addView(String name, String permissions, String template, List<?> tabs, String repository)throws Exception	Inserts a new view by giving the following params @param name: The name of view @param permissions: who can access the view @param template: The name of template @param tabs: list of tabs @param repository: The name of repository @throws Exception: throws exception
getViewByName	Node	getViewByName(String viewName, String repository, SessionProvider provider) throws Exception	Return specify view depend on Name by giving the following params @param viewName: The name of view @param repository: The

Method	Return	Prototype	Desription
			name of repository @param provider: The SessionProvider object is used to managed Sessions @throws Exception: throws exception
getButtons	List<?>	getButtons() throws Exception	Return all string of buttons @throws Exception: throws exception
removeView	void	removeView(String viewName, String repository) throws Exception	Removes the view by giving the following params @param viewName: The name of view @param repository: The name of repository @throws Exception: throws exception
getAllViews	List<ViewConfig>	getAllViews(String repository) throws Exception	Return all views of the repository is configed in XML file by giving the following params @param repository: The name of repository @throws Exception: throws exception
hasView	boolean	hasView(String name, String repository) throws Exception	Returns true is the given repository has view by giving the following params

Method	Return	Prototype	Desription
			@param name: The name of view @param repository: The name of repository @throws Exception: throws exception
getTemplateHome	Node	getTemplateHome(String homeAlias, String repository, SessionProvider provider) throws Exception	Get teamplate Node that has path by giving the following params @param homeAlias @param repository: The name of repository @param provider: The SessionProvider object is used to managed Sessions @throws Exception: throws exception
getAllTemplates	List<Node>	getAllTemplates(String homeAlias, String repository,SessionProvider provider) throws Exception	Gets all node that has template path to the given node @param homeAlias: Alias of template home @param repository: The name of repository @param provider: The SessionProvider object is used to managed Sessions @throws Exception: throws exception
getTemplate	Node	getTemplate(String path, String	Return node that has path of the repository

Method	Return	Prototype	Desription
		repository,SessionProvider provider) throws Exception	@param path: The path of template @param repository: The name of repository @param provider: SessionProvider @throws Exception: throws exception
addTemplate	String	addTemplate(String name, String content, String homePath, String repository)throws Exception	Inserts a new template for node by specified path @param name: The name of new template @param content: The property of template @param homePath: The path of specified node @param repository: The name of repository @throws Exception: throws exception
removeTemplate	void	removeTemplate(String templatePath, String repository) throws Exception	Removes the template to the given node by specified the templatePath params @param templatePath: The path of template @param repository: The name of repository @throws Exception: throws exception

Method	Return	Prototype	Desription
addTab	void	addTab(Node view, String name, String buttons) throws Exception	Insert new tab to the givin view node by specified the following params @param view: Specify the node wants to add a tab @param name: The name of tab @param buttons: The buttons of tab @throws Exception: throws exception
init	void	init(String repository) throws Exception	Get all template that is configed in XML file of specified repository @param repository: The name of repository @throws Exception: throws exception

6.19. ApplicationTemplateManager

This class is 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

Method	Return	Prototype	Description
getAllManagedPortletNames	List<String>	getAllManagedPortletNames(String repository) throws Exception ;	Retrieves the all portlet names that have dynamic groovy templates are managed by service. repository the repository @throws Exception the exception
getTemplatesByApplication	List<Node>	getTemplatesByApplication(String repository, String portletName, SessionProvider provider) throws Exception ;	Retrieves 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;	Retrieves the templates node by category. @param repository the repository @param portletName the portlet name @param category the category @param sessionProvider the session provider

Method	Return	Prototype	Description
			@return the templates by category @throws Exception the exception
getTemplateByName	Node	getTemplateByName(String repository, String portletName, String category, String templateName, SessionProvider sessionProvider) throws Exception;	Retrieves 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 ;	Gets the template by path. @param repository the repository @param templatePath the template path @param sessionProvider the session provider

Method	Return	Prototype	Description
			@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 ;	Removes 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

6.20. NodeFinder

NodeFinder is used to find node with given path. If path to node contains sub-paths to exo:symlink nodes then

find real link node.

Method	Return	Prototype	Desription
getNode	Node	getNode(Node ancestorNode, String relativePath) throws PathNotFoundException, RepositoryException;	Returns the node at relPath relative 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 node exists at the specified path. @throws RepositoryException if another error occurs.
getNode	Node	getNode(Node ancestorNode, String relativePath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	Returns the node at relPath relative 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: Indicates if the target must be returned in case

Method	Return	Prototype	Desription
			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;	Returns 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;	Returns the item at the specified absolute path. @param repository: The name of repository @param workspace: The name of workspace @param absPath: An absolute path.

Method	Return	Prototype	Desription
			@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;	Returns 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 workspace, String absPath, boolean	Returns 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

Method	Return	Prototype	Desription
		giveTarget, boolean system) throws PathNotFoundException, RepositoryException;	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;	Returns the item at the specified absolute path. @param session: The session to u se in order to get the item @param absPath: An absolute path. @throws PathNotFoundException if the specified path cannot be found.

Method	Return	Prototype	Desription
			@throws RepositoryException if another error occurs.
getItem	Item	getItem(Session session, String absPath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	Returns 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;	Returns 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

Method	Return	Prototype	Desription
			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;	Returns <code>true</code> if an item exists at absPath; otherwise returns <code>false</code> Also returns <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 RepositoryException if

Method	Return	Prototype	Desription
			an error occurs.

6.21. DMSConfiguration

DMSConfiguration is used to manage dms-workspace in repository

Package org.exoplatform.services.cms.impl.DmsConfiguration;

Method	Return	Prototype	Desription
getConfig	DMSRepositoryConfiguration	getConfig(String repository)	Get DMS configuration with specific repository. @param repository: repository name. @return: DMSRepositoryConfiguration
addPlugin	void	addPlugin(ComponentPlugin plugin)	This method will add more plugin. @param plugin: plugin name
initNewRepo	void	initNewRepo(String repository, DMSRepositoryConfiguration plugin)	This method will create new repository @param repository: repository name @param plugin: plugin name

6.22. DocumentType

DocumentTypeService get all documents by mime types.

Method	Return	Prototype	Desription
getAllSupportedType	List<String>	getAllSupportedType();	

Method	Return	Prototype	Desription
			Get all supported document type @return List<String>
getAllDocumentsByDocumentType	List<Node>	getAllDocumentsByDocumentType(String documentType, String workspace, String repository, SessionProvider sessionProvider) throws Exception;	Get all documents by kind of document type @param documentType Kind of document(Images, Video) @param workspace The name of workspace will be used to get documents @param repository The name of repository will be used to get documents @param sessionProvider @param mimeType The mime type of node(For example image/jpg) @return List<Node> all documents by kind of document type @throws Exception
getAllDocumentsByType	List<Node>	getAllDocumentsByType(String workspace, String repository, SessionProvider sessionProvider, String mimeType) throws Exception;	Get all document by mimetype @param workspace The name of workspace will be used to get documents @param repository The name of repository will be used to get documents

Method	Return	Prototype	Desription
			@param sessionProvider @param mimeType The mime type of node(For example image/jpg) @return List<Node> all documents by mime type @throws Exception
getAllDocumentsByUser	List<Node>	getAllDocumentsByUser(String workspace, String repository, SessionProvider sessionProvider, String//mimeTypes, String userName) throws Exception;	Get all document type by user @param workspace The name of workspace will be used to get documents @param repository The name of repository will be used to get documents @param sessionProvider @param mimeTypes The array of mimetype(For example image/jpg, image/png) @param userName The name of current use @return List<Node> all documents by mime type @throws Exception
isContentsType	boolean	isContentsType(String documentType);	Check the document is content type or not @param documentType @return

Method	Return	Prototype	Desription
getAllDocumentByContentsType	List<Node>	ggetAllDocumentByContentsType(String documentType, String workspace, String repository, SessionProvider sessionProvider, String userName) throws Exception;	Get all contents type document @param documentType Contents type @param workspace The name of workspace will be used to get documents @param repository The name of repository will be used to get documents @param sessionProvider @return List<Node> all contents type document @param userName @return @throws Exception
getMimeTypes	String//	getMimeTypes(String documentType);	Get mime types by document type @param documentType @return array of string

6.23. Favorite

FavoriteService this service used to manage all favorited content adding by user.

Method	Return	Prototype	Desription
addFavorite	void	addFavorite(Node node, String userName) throws	Add favorite to node

Method	Return	Prototype	Desription
		Exception;	@param node Add favorite to this node @param userName The user added favorite @throws Exception The exception will be raised if the node can not add mixin
removeFavorite	void	removeFavorite(Node node, String userName) throws Exception;	Remove favorite from node @param node Remove favourite out of this node @param userName Remove the name of current user out of property exo:favouriter @throws Exception
getAllFavoriteNodesByUser	List<Node>	getAllFavoriteNodesByUser(String workspace, String repository, String userName) throws Exception;	Get all favourite nodes by user @param workspace Get all favorite nodes from this workspace @param repository Get all favorite nodes from this repository @param sessionProvider The session provider which will be used to get session @param userName User added favorite to the node @return List<Node> All

Method	Return	Prototype	Desription
			favorite node added by user @throws Exception
isFavoriter	boolean	isFavoriter(String userName, Node node) throws Exception ;	Check if user is in favourite list of node @param node Node to check @param userName The user to check

6.24. Trash

TrashService this service used to move documents to trash foder or restore.

Method	Return	Prototype	Desription
moveToTrash	void	moveToTrash(Node node, String trashPath, String trashWorkspace, String repository, SessionProvider sessionProvider) throws Exception;	Move node to trash location @param node Node will be moved to trash @param trashPath The trash node path @param trashWorkspace The trash workspace @param repository The repository name @param sessionProvider User session provider which will be used to get session @throws Exception

Method	Return	Prototype	Desription
restoreFromTrash	void	restoreFromTrash(Node trashHomeNode, String trashNodePath, String repository, SessionProvider sessionProvider) throws Exception;	Restore node from trash @param trashHomeNode trash home node @param restorePath Restore path which will be used to restore @param restoreWorkspace The workspace name of node which moved to trash @param repository The repository name @param sessionProvider User session provider which will be used to get session @throws Exception
getAllNodeInTrash	List<Node>	getAllNodeInTrash(String trashWorkspace, String repository, SessionProvider sessionProvider) throws Exception;	Get all nodes in trash location @param trashWorkspace @param repository @param sessionProvider @return List<Node> All nodes in trash @throws Exception
getAllNodeInTrashByUser	List<Node>	getAllNodeInTrashByUser(String trashWorkspace, String repository, SessionProvider sessionProvider, String	Get all nodes by user in trash location @param trashWorkspace

Method	Return	Prototype	Desription
		userName) throws Exception;	@param repository @param sessionProvider @param userName @return List<Node> all node in trash which moved by user @throws Exception
removeRelations	List<Node>	removeRelations(Node node, SessionProvider sessionProvider, String repository) throws Exception;	Removes all **relationable** property of nodes that have relation to this node @param node @param sessionProvider @param repository @return remaining nodes in trash @throws Exception

6.25. Timeline

TimelineService is used to get all documents by time frame

Method	Return	Prototype	Desription
getDocumentsOfToday	List<Node>	getDocumentsOfToday(String nodePath, String repository, String workspace, SessionProvider sessionProvider, String userName, boolean	Get all documents of Today @param nodePath Path of current node @param repository

Method	Return	Prototype	Desription
		byUser) throws Exception;	Repository name @param workspace Workspace name @param sessionProvider SessionProvider @param userName Logged in user @param byUser show documents by current user or by all users @return List<Node>
getDocumentsOfYesterday	List<Node>	getDocumentsOfYesterday(String nodePath, String repository, String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception	Get all documents of Yes terday @param nodePath Path of current node @param repository Repository name @param workspace Workspace name @param sessionProvider SessionProvider @param userName Logged in user @param byUser show documents by current user or by all users @return List<Node>
getDocumentsOfEarlierThisWeek	List<Node>		

Method	Return	Prototype	Desription
		getDocumentsOfEarlierThisWeek(String nodePath, String repository, String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Excep tion;	Get all documents earlier this week @param nodePath Path of current node @param repository Repository name @param workspace Workspace name @param sessionProvider SessionProvider @param userName Logged in user @param byUser show documents by current user or by all users @return List<Node>
getDocumentsOfEarlierThisMonth	List<Node>	getDocumentsOfEar lierThisMonth(String nodePath, String repository, String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	Get all documents earlier this month @param nodePath Path of current node @param repository Repository name @param workspace Workspace name @param sessionProvider SessionProvider @param userName Logged in user

Method	Return	Prototype	Desription
			@param byUser show documents by current user or by all users @return List<Node>
getDocumentsOfEarlierThisYear	List<Node>	getDocumentsOfEarlierThisYear(String nodePath, String repository, String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	Get all documents earlier this year @param nodePath Path of current node @param repository Repository name @param workspace Workspace name @param sessionProvider SessionProvider @param userName Logged in user @param byUser show documents by current user or by all users @return List<Node>

6.26. Lock

LockService this service used to manage lock services which enable the user to lock, unlock, assign.

Method	Return	Prototype	Desription
getPreSettingLockList	List<String>	getPreSettingLockList() throws Exception;	Get predefined locked list @throws Exception
getAllGroupsOrUsersForLock	List<String>	getAllGroupsOrUsersForLock() throws Exception;	This method will get

Method	Return	Prototype	Desription
			collection of groups or users which are under locked status @return List of group or user @throws Exception
addGroupsOrUsersForLock	void	addGroupsOrUsersForLock(String groupsOrUsers) throws Exception;	Add group or user to lock priviledge list @param groupsOrUsers Name of group or use @throws Exception
removeGroupsOrUsersForLock	void	removeGroupsOrUsersForLock(String groupsOrUsers) throws Exception;	Method allow to remove a group or user from priviledge list @param groupsOrUsers Name of groups or users @throws Exception

6.27. JodConverter

JodConverter allow to converts documents between different office formats

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

Method	Return	Prototype	Desription
convert	void	convert(InputStream	

Method	Return	Prototype	Desription
		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

6.28. WatchDocument

WatchDocumentService this service used to move documents to trash foder or restore.

Method	Return	Prototype	Desription
watchDocument	void	watchDocument(Node documentNode, String userName, int notifyType) throws Exception;	Watching the document that is specified by the node by giving a userName, notifyType If the document is watching, all thing that changes to it's property will be notified to user specified by the userName @param documentNode Specify the document for watching @param userName he username of current user is votting. It can't be

Method	Return	Prototype	Desription
			<code>null<code> @param notifyType Type of notification. Its can be 0, 1 or 2 0 Notification by email 1 Notification by rss 2 Full notification @see Node @throws Exception
getNotificationType	int	getNotificationType(Node documentNode, String userName) throws Exception;	This method will gets the type of notification for the specify document If that document is not a exo watchable document, the value return is -1 If notification is notified by email, the value return is 1 If notification is notified by rss, the value return is 2 If notification is notified by rss and email, the value return is 0 @param documentNode Specify the document for watching @param userName The username of current user is votting. It can't be

Method	Return	Prototype	Desription
			<code>null<code> @see Node @return 0, 1, 2 or -1 @throws Exception
unwatchDocument	void	unwatchDocument(Node documentNode, String userName, int notifyType) throws Exception;	Watching the document that is specified by the node by giving a userName, notifyType If the document is watching, all thing that changes to it's property will be notified to user specified by the userName @param documentNode Specify the document for watching @param userName he username of current user is votting. It can't be <code>null<code> @param notifyType Type of notification. Its can be 0, 1 or 2 0 Notification by email 1 Notification by rss 2 Full notification @see Node @throws Exception

Method	Return	Prototype	Desription

6.29. MetaData

MetadataService is used to process with meta data for system

Method	Return	Prototype	Desription
getMetadataList	List<String>	getMetadataList(String repository) throws Exception;	Get all NodeType in repository with NodeType=exo:metadata @param repository: repository name @return: ArrayList of NodeType
getAllMetadatasNodeType	List<NodeType>	getAllMetadatasNodeType(String repository) throws Exception ;	Get all NodeType in repository with NodeType = exo:metadata @param repository: repository name @return: ArrayList of NodeType
addMetadata	String	addMetadata(String nodetype, boolean isDialog, String role, String content, boolean isAddNew, String repository) throws Exception;	Add new nodetype and set property EXO*ROLES*PROP, EXO*TEMPLATE*FILE*PROP for dialog template node or view template node if node doesn't exist. Set property *EXO*ROLES*PROP*, *EXO*TEMPLATE*FILE*PROP for dialog template node or view template node if node exists @param nodetype: Node

Method	Return	Prototype	Desription
			name for processing @param isDialog: true for dialog template @param role: permission param content: content of template @param isAddNew: false if nodetype exist in repository, true if not @param repository: repository name @return path to node if node exist, otherwise return null @throws Exception
removeMetadata	void	removeMetadata(String nodetype, String repository) throws Exception;	Remove node named nodetype below baseMetadataPath @param nodetype: name of node @param repository: repository name
getExternalMetadataType	List<String>	getExternalMetadataType(String repository) throws Exception;	Get all NodeType name that contains property that is not autocreated and name of NodeType differs from exo:metadata @param repository: repository name @return: ArrayList of

Method	Return	Prototype	Desription
			metadata type
getMetadataTemplate	String	getMetadataTemplate(String name, boolean isDialog, String repository) throws Exception;	Get content of dial og template node or view template in repository @param name: Node name @param isDialog: true: Get dialog template content false: Get view template content @param repository: repository name @return: content of template
getMetadataPath	String	getMetadataPath(String name, boolean isDialog, String repository) throws Exception;	Get path to dialog template or view tempate node @param name: Node name @param isDialog: true: Get dialog template content false: Get view template content @param repository: repository name @return: path to template node
getMetadataRoles	String	getMetadataRoles(String	

Method	Return	Prototype	Desription
		name, boolean isDialog, String repository) throws Exception;	Get permission of template node @param name: Node name @param isDialog: true: Get dialog template content false: Get view template content @param repository: repository name @return: String of permission
hasMetadata	boolean	hasMetadata(String name, String repository) throws Exception;	Check node with given name exists or not below baseMetadataPath path in repositorys @param name: Node name @param repository: repository name @return true : Exist this node name false: Not exist this node name
init	void	init(String repository) throws Exception;	Call all available in list of TemplatePlugin to add some predefine template to repository @repository: repository name

Method	Return	Prototype	Desription
			@throws Exception

6.30. CompressData

CompressData is used to compress data file

Package org.exoplatform.services.cms.compress.CompressData;

Method	Return	Prototype	Desription
getBase	String	getBase()	
addFile	void	addFile(String entryName, File file)	
addDir	void	addDir(File srcDir)	
addInputStream	void	addInputStream(String entryName, InputStream is) throws Exception	
createZipFile	void	createZipFile(String fileName) throws Exception	
createZip	void	createZip(OutputStream os) throws Exception	
createJarFile	void	createJarFile(String fileName) throws Exception	
createJar	void	createJar(OutputStream os) throws Exception	
cleanDataInstance	void	cleanDataInstance()	

Chapter 7. FAQ

7.1.1. How do I create a new Site ?

7.1.2. How do I create a new Content Type ?

7.1.3. How do I create a new Site ?