**Alfresco Live Tutorial**

Contents

[Managing Alfresco Lifecycle with Maven 2](#_Toc355804068)

[Prerequisites 2](#_Toc355804069)

[Create Project 3](#_Toc355804070)

[Running Project 4](#_Toc355804071)

[Using command prompt 4](#_Toc355804072)

[Using Eclipse IDE 4](#_Toc355804073)

[Creating Model 7](#_Toc355804074)

[Execute Test Model 7](#_Toc355804075)

[Steps to see custom content type 7](#_Toc355804076)

[Steps to see custom aspects 8](#_Toc355804077)

[**Alfresco Module Package(AMP)** 10](#_Toc355804078)

# Managing Alfresco Lifecycle with Maven

## Prerequisites

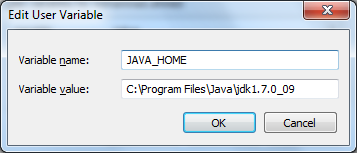
1. Install java 7.0 and set JAVA\_HOME and path as follow

Window 2007

Start->Computer->Right Click -> properties ->Advance System Setting ->new

Variable Name: JAVA\_HOME

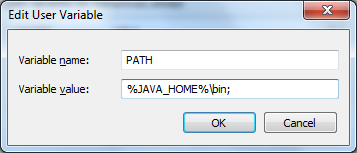
Variable Value: C:\Program Files\Java\jdk1.7.0\_09



Set Path

Variable Name: PATH

Variable Value: %JAVA\_HOME%\bin;



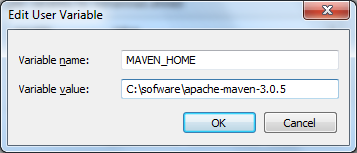
1. Download latest version of [Maven](http://maven.apache.org/download.cgi) and set environmental variable as follow

Window 2007

Start->Computer->Right Click -> properties ->Advance System Setting ->new

Variable Name: MAVEN\_HOME

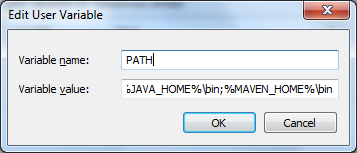
Variable Value: C:\software\apache-maven-3.0.5



In PATH variable append MAVEN\_HOME

Variable Name: PATH

Variable Value: %JAVA\_HOME%\bin;%MAVEN\_HOME%\bin



## Create Project

Follow below mentioned steps for creating alfresco maven project

Open command prompt.

Type below command line

mvn archetype:generate -DarchetypeCatalog=https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-catalog.xml

Select archetype

1: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> org.alfresco:maven-alfresco-extension-archetype (This archetype d

eveloped aims to provide a standardized approach to development, release and dep

loyment of Alfresco extensions (as opposed to

AMP builds, to be released as a different artifact ).)

2: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> org.alfresco:maven-alfresco-amp-archetype (This archetype aims to

provide a standardized approach to development, release and deployment of Alfre

sco AMPs (as opposed to Alfresco

extensions, released as a different artifact ))

3: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> org.alfresco.maven:maven-alfresco-share-module-archetype (-)

4: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> org.alfresco.maven:maven-alfresco-share-archetype (Maven Archetyp

e to allow Maven based customization of Alfresco SHARE)

5: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> org.alfresco.enablement.codecamps.cmis:cmis-master-labs-archetype

(-)

6: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> com.sourcesense.alfresco:maven-alfresco-extension-archetype (This

archetype developed aims to provide a standardized approach to development, rel

ease and deployment of Alfresco extensions (as opposed to

AMP builds, to be released as a different artifact ).)

7: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> com.sourcesense.alfresco:maven-alfresco-amp-archetype (This arche

type aims to provide a standardized approach to development, release and deploym

ent of Alfresco AMPs (as opposed to Alfresco

extensions, released as a different artifact ))

8: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> org.alfresco.maven.archetype:alfresco-amp-archetype (Sample proje

ct with full support for lifecycle and rapid development of AMPs (Alfresco Modul

e Packages))

9: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype-

catalog.xml -> org.alfresco.maven.archetype:alfresco-allinone-archetype (Sample

multi-module project for All-in-One development on the Alfresco plaftorm. Includ

es modules for: Repository, AMP, Share, Solr, Web Quick Start and embedded Jetty

run)

10: https://artifacts.alfresco.com/nexus/content/repositories/releases/archetype

-catalog.xml -> org.alfresco:alfresco-benchmark-client-archetype (-)

Choose a number or apply filter (format: [groupId:]artifactId, case sensitive co

ntains): :

**Note:** select archetype 9 is recommended, which has All-in-One development on the Alfresco plaftorm. Includes modules for: Repository, AMP, Share, Solr, Web Quick Start and embedded Jetty

1. Choose archetype version. It will take few second to download archetype.
2. Now define value for property ‘groupId’, like com.myproject.demo
3. Define value for property ‘artifactId’, like myproject. It will the name of the project.
4. After that, it will ask for confirm property configuration. If you want to change property configuration name than press ‘N’ otherwise press ‘Y’.

It will download all required project related files.

## Running Project

### Using command prompt

Follow below steps to running alfresco project in command prompt

1. Navigate to created project folder, in our case type command “cd myproject” and press enter.
2. Now Type “mvn install -Prun” and press enter. It run all the lifecycle of maven, like

Validate, compile, test and build war files and will deploy in inbuilt jetty server.

**Troubleshoot:**

1. **Issue:** While running project if it complain for Java.lang.OutOfMemoryError: PermGen error.

**Solution:** set memory size as follow.

set MAVEN\_OPTS=-Xmx512M -XX:MaxPermSize=512M

set JVM\_ARGS="-Xmx1024m -XX:MaxPermSize=256m"

1. **Issue:** If you face like below error.

[ERROR] error: error reading C:\Users\mohammad.ahmed\.m2\repository\org\codehaus\groovy\groovy\1.7.5\groovy-1.7.5.jar; invalid LOC header (bad signature)

**Solution:** Navigate to the folder where the file complaining the error and delete all files, then run project again.

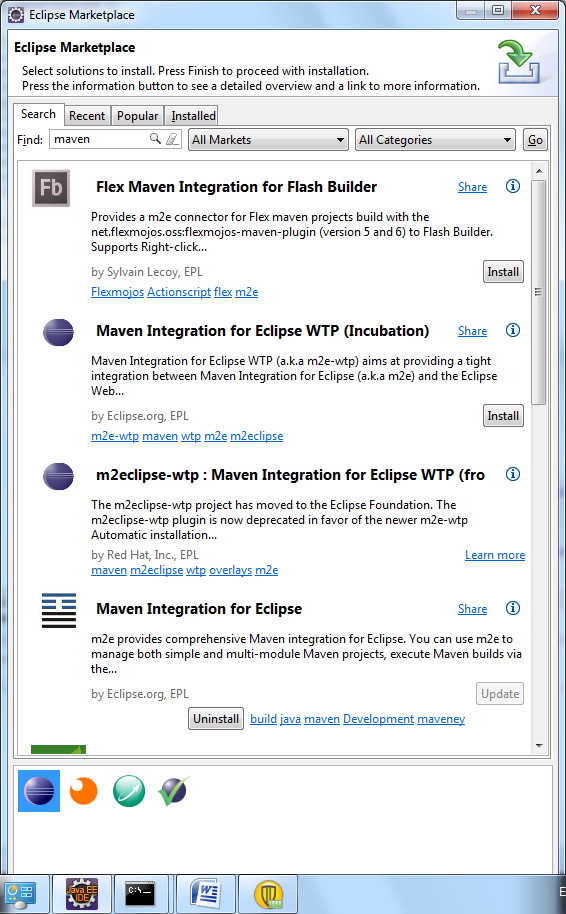
### Using Eclipse IDE

#### Maven Plug-in

Follow the below mentioned steps to install plug-in.

1. Download the eclipse maven plug-in. If you are using Juno version then navigate to

Help->Eclipse Marketplace and type “Maven” in find textbox.

2) Select “Maven Integration for Eclipse” to install maven plug-in.

#### Setting Project

1. Import alfresco project based on maven, in our case project name is “myproject” using “Existing Maven Project” source.
2. Right click on “myproject”->run->maven build...->(in “Goalds” field type “install -Prun”)->run

**Troubleshoot:** if you see below in error in any POM.xml file.

Plugin execution not covered by lifecycle configuration: org.alfresco.maven.plugin:alfresco-maven-plugin:1.0:set-version (execution: default-set-version, phase: initialize)

**Solution:** Add following code in POM.xml

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.eclipse.m2e</groupId>

<artifactId>lifecycle-mapping</artifactId>

<version>1.0.0</version>

<configuration>

<lifecycleMappingMetadata>

<pluginExecutions>

<pluginExecution>

<pluginExecutionFilter>

<!--<groupId>some-group-id</groupId>

<artifactId>some-artifact-id</artifactId>

<versionRange>[1.0.0,)</versionRange>-->

<groupId>org.alfresco.maven.plugin</groupId>

<artifactId>alfresco-maven-plugin</artifactId>

<versionRange>[0.0.0,)</versionRange>

<goals>

<goal>set-version</goal>

</goals>

</pluginExecutionFilter>

<action>

<execute >

<runOnIncremental>false</runOnIncremental>

</execute >

</action>

</pluginExecution>

</pluginExecutions>

</lifecycleMappingMetadata>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

# Creating Model for Explorer (<http://localhost:8080/alfresco>)

**Create Model**

Create “sd-content-model.xml” and placed at “C:\Alfresco\tomcat\shared\classes\alfresco\extension\model\sd”. If folder is not already created then create missing folder for the same.

<?xml version="1.0" encoding="UTF-8"?>

<!-- Definition of new Model -->

<!-- The important part here is the name - Note: the use of the sd: namespace

which is defined further on in the document -->

<model name="sd:sdcontentmodel" xmlns="http://www.alfresco.org/model/dictionary/1.0">

<!-- Optional meta-data about the model -->

<description>SD Content Model</description>

<author>Mohammad Jamil Ahmed</author>

<version>1.0</version>

<!-- Imports are required to allow references to definitions in other models -->

<imports>

<!-- Import Alfresco Dictionary Definitions -->

<import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d"/>

<!-- Import Alfresco Content Domain Model Definitions -->

<import uri="http://www.alfresco.org/model/content/1.0" prefix="cm"/>

</imports>

<!-- Introduction of new namespaces defined by this model -->

<!-- NOTE: The following namespace my.new.model should be changed to reflect your own namespace -->

<namespaces>

<namespace uri="sd.content.model" prefix="sd"/>

</namespaces>

<types>

<!-- Definition of new Content Type: development-->

<type name="sd:development">

<title>Development</title>

<parent>cm:content</parent>

<properties>

<property name="sd:name">

<type>d:text</type>

</property>

</properties>

<mandatory-aspects>

<aspect>sd:common</aspect>

</mandatory-aspects>

</type>

</types>

<aspects>

<aspect name="sd:common">

<title>Common</title>

<properties>

<property name="sd:siteName">

<type>d:text</type>

</property>

<property name="sd:uploadedDate">

<type>d:datetime</type>

</property>

<property name="sd:user">

<type>d:text</type>

<default>User1</default>

<constraints>

<constraint name="sd:userList" type="LIST">

<parameter name="allowedValues">

<list>

<value>User1</value>

<value>User2</value>

<value>User3</value>

</list>

</parameter>

<parameter name="caseSensitive"><value>true</value></parameter>

</constraint>

</constraints>

</property>

</properties>

</aspect>

</aspects>

</model>

**Display Model Component**

Add following code to ‘web-client-config-custom.xml’ file at ‘C:\Alfresco\tomcat\shared\_old\classes\alfresco\extension\web-client-config-custom.xml’ to display Model component

<config evaluator="string-compare" condition="Content Wizards">

<content-types>

<type name="sd:development" />

</content-types>

</config>

<config evaluator="node-type" condition="sd:development">

<property-sheet>

<show-property name="sd:name" />

</property-sheet>

</config>

<config evaluator="string-compare" condition="Action Wizards">

<aspects>

<aspect name="sd:common"/>

</aspects>

</config>

<config evaluator="aspect-name" condition="sd:common">

<property-sheet>

<show-property name="sd:siteName"/>

<show-property name="sd:uploadedDate"/>

<show-property name="sd:user"/>

</property-sheet>

</config>

**Model Localization**

Create “sd-content.properties” and placed at “C:\Alfresco\tomcat\shared\classes\alfresco\extension\model\sd” same as sd-content-model.xml

# Display labels for Content Domain Model

sd\_sdcontentmodel.type.sd\_development.title=Development

sd\_sdcontentmodel.property.sd\_name.title=Name

sd\_sdcontentmodel.property.sd\_siteName.title=Site Name

sd\_sdcontentmodel.property.sd\_uploadedDate.title=Uploaded Date

sd\_sdcontentmodel.property.sd\_user.title=User List

**Register the Model (**sd-content-model.xml**) and Model Localization (**sd-content.properties**) with the Repository**

Create “sd-model-context.xml” and placed at “C:\Alfresco\tomcat\shared\classes\alfresco\extension\”.

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>

<!-- Registration of new models -->

<bean id="example.dictionaryBootstrap" parent="dictionaryModelBootstrap" depends-on="dictionaryBootstrap">

<property name="models">

<list>

<!--value>alfresco/extension/exampleModel.xml</value-->

<value>alfresco/extension/model/sd/sd-content-model.xml</value>

</list>

</property>

<property name="labels">

<list>

<value>alfresco/extension/model/sd/sd-content</value>

</list>

</property>

</bean>

</beans>

## Execute Test Model

Open command prompt, navigate to model file which is to be test and execute below command

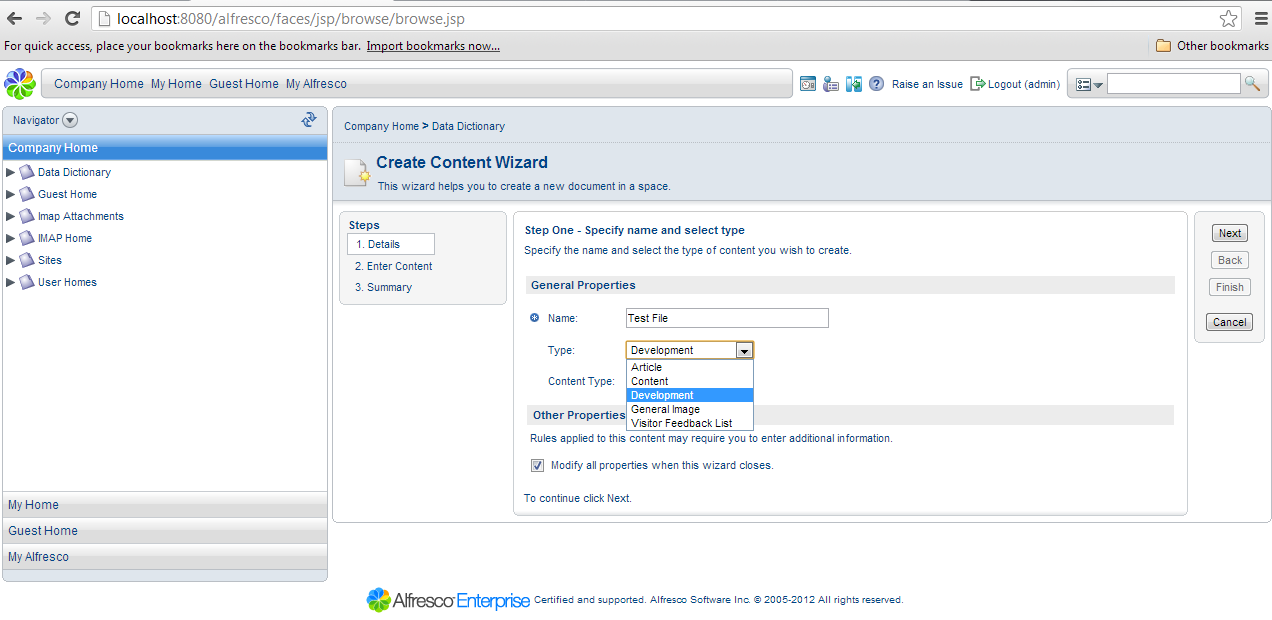
Example: “example-model-context.xml” is model name

java -cp ".;C:\Alfresco\tomcat\webapps\alfresco\WEB-INF\lib\\*;C:\Alfresco\tomcat\webapps\alfresco\WEB-INF\classes" org.alfresco.repo.dictionary.TestModel sd-model-context.xml

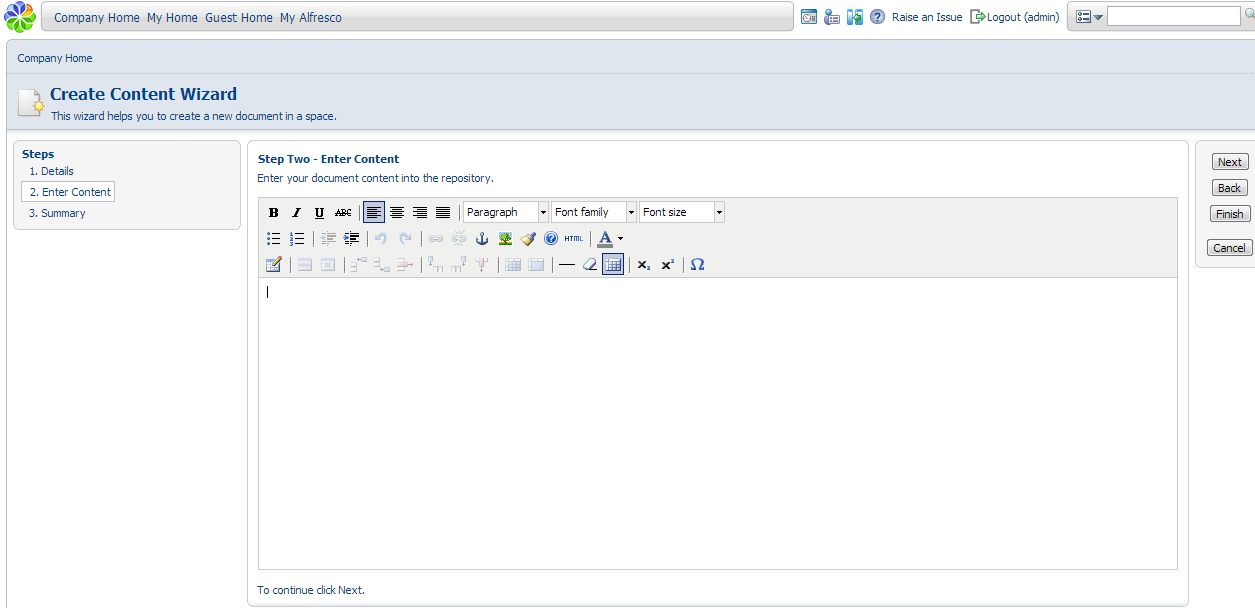
## Steps to see custom content type

**Follow below mentioned steps to see custom “content type” in Alfresco.**

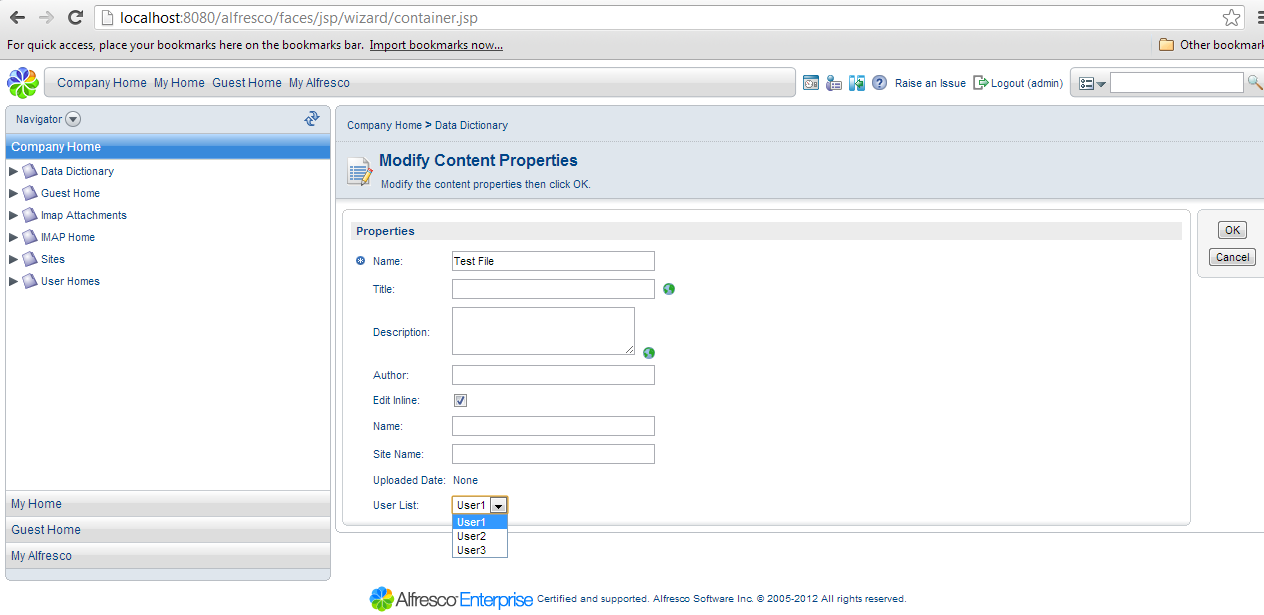
1. Log in to Alfresco with valid credentials.
2. On the dashboard, click on the Company Home hyperlink available on the top left corner.
3. Click on the “Create” Menu available on the top right corner.
4. Select “Create Content” from the drop down menu. A “Create Content Wizard” screen will show up. Here you will be able to see your custom content type in “Type” dropdown as shown below in the screenshot.



1. Enter value in the Name field and click on the “Next” button; it will take you to the alfresco editor (as shown below).



1. Click on the Finish Button, it will take you to the “Modify content properties screen” as shown below.



As we have selected custom type “Global Logic Scanner File” on the Create Content Wizard screen, its properties (Custom File Id, Custom File Name and Custom File Status) are displayed on the above screenshot.

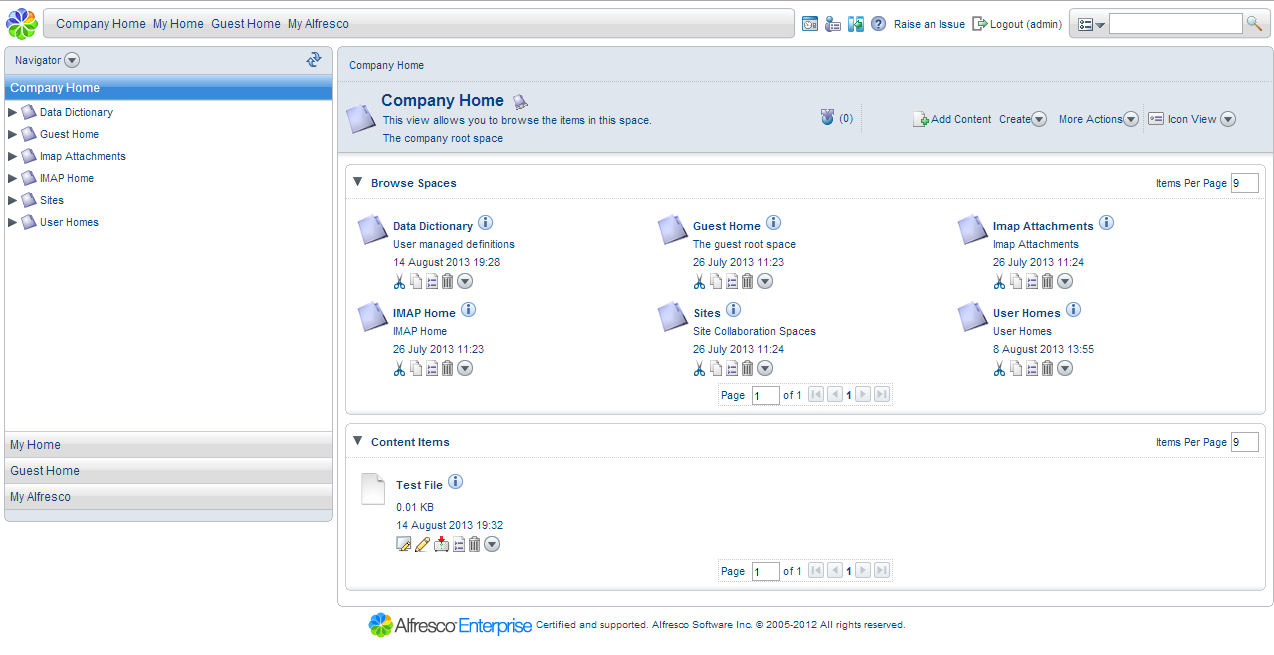
1. Fill the form and click on the “Ok” button.

Now you have successfully created custom content type “Global Logic Scanner File” and have applied it to its related content.

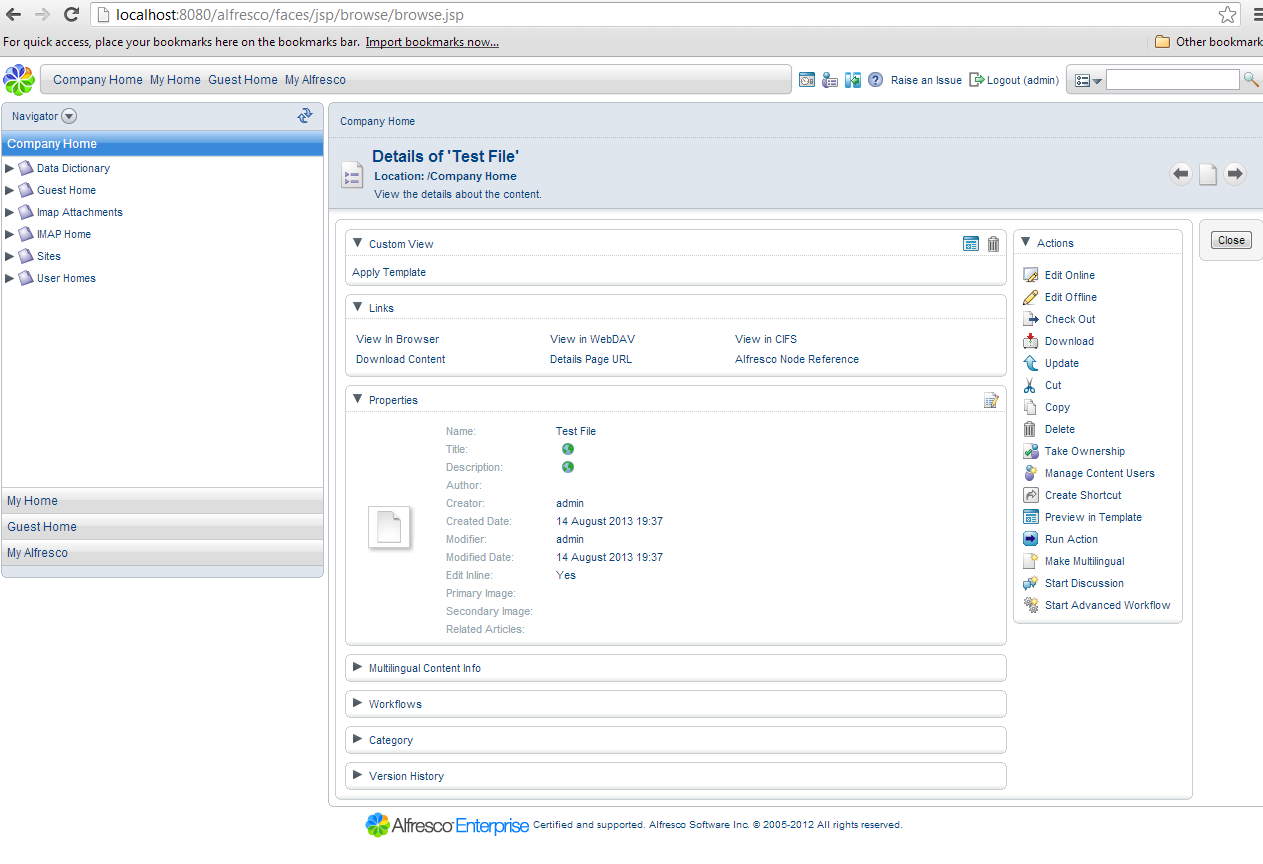
## Steps to see custom aspects

**Follow below mentioned steps to see custom “Aspect” in Alfresco.**

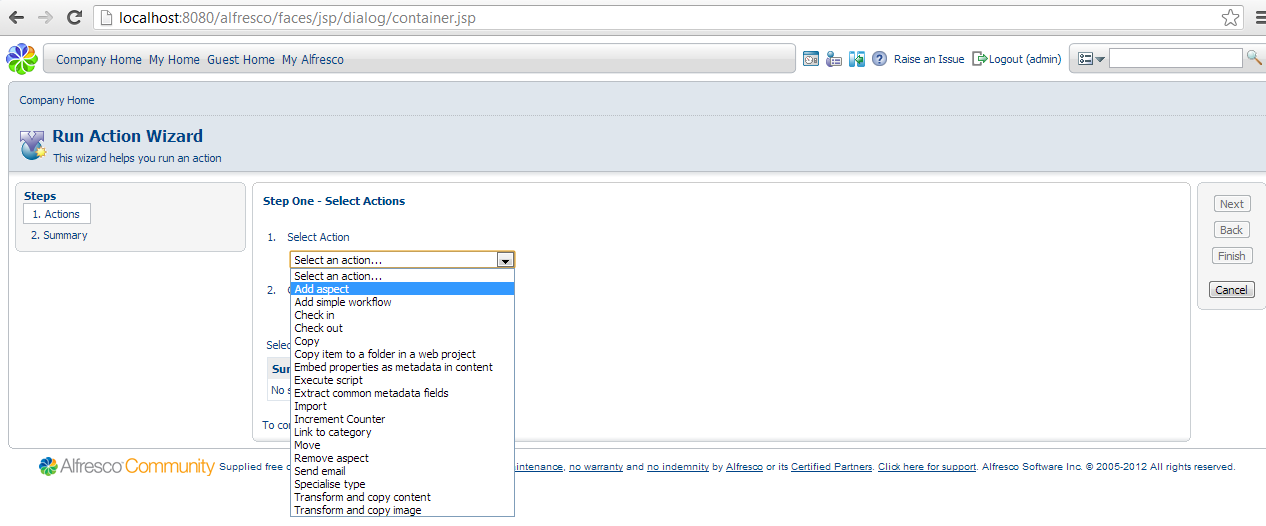
1. Click on the “View details” Action of the ‘Test File’ Content Item as shown below.

****

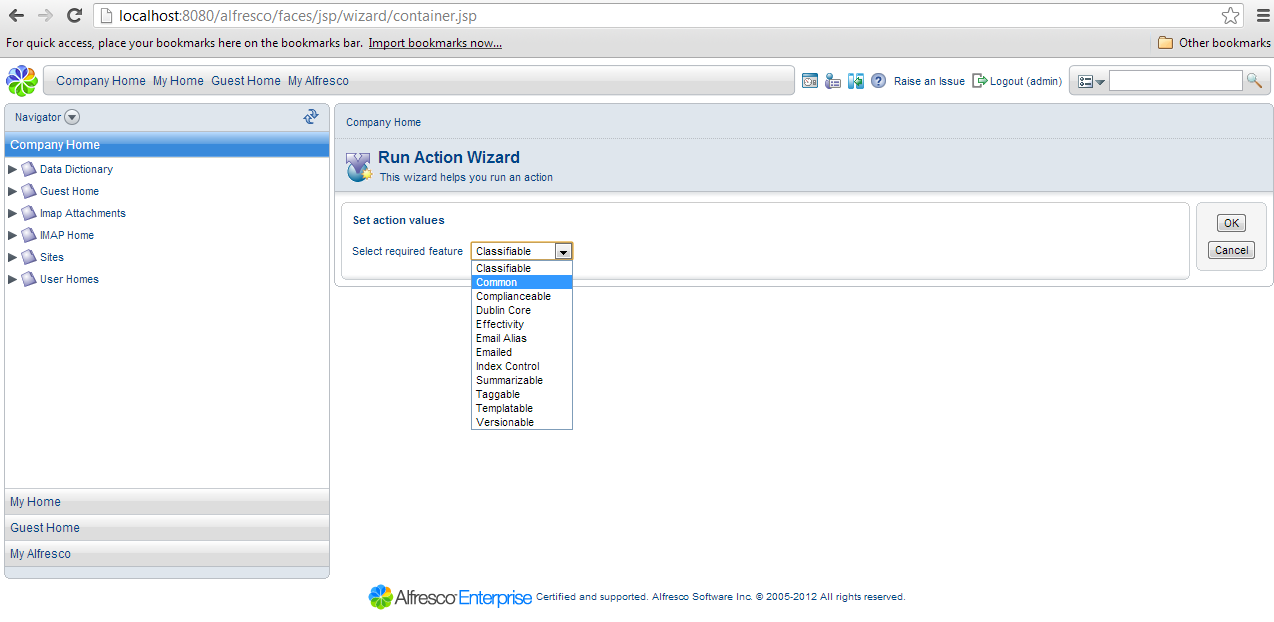
1. It will take you to the content view details screen as shown below.



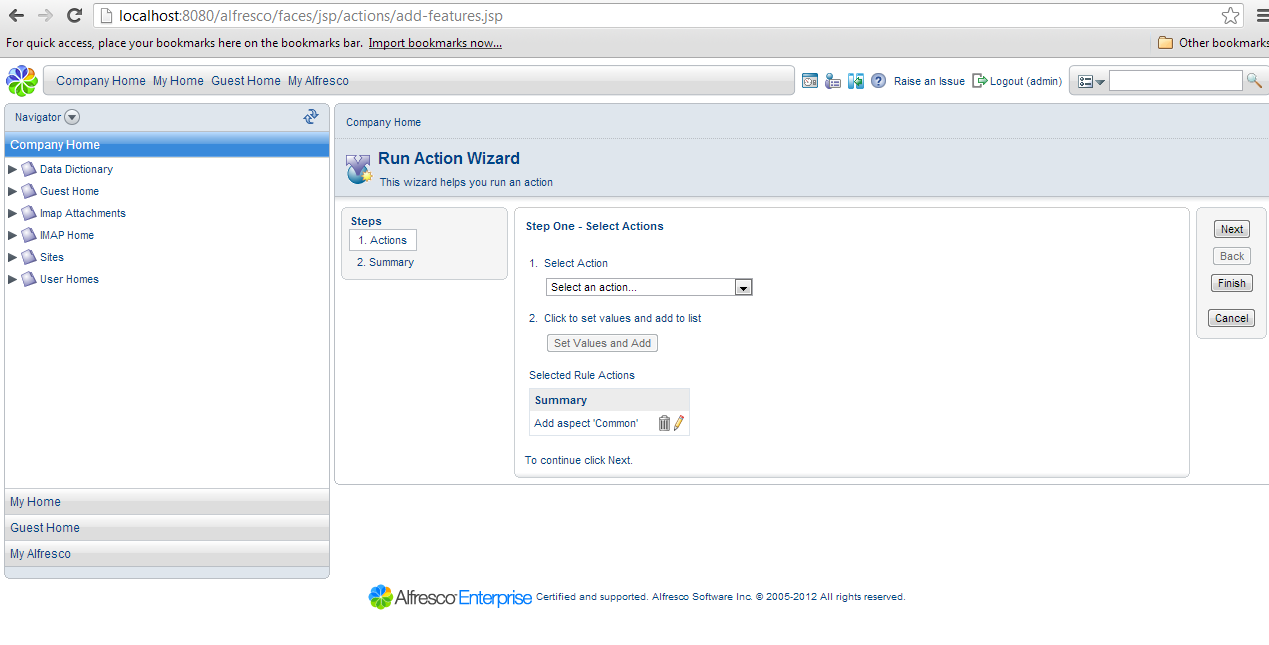
1. Click on the “Run Action” link on Action Menu. It will take you the “Run Action Wizard” screen as shown below.



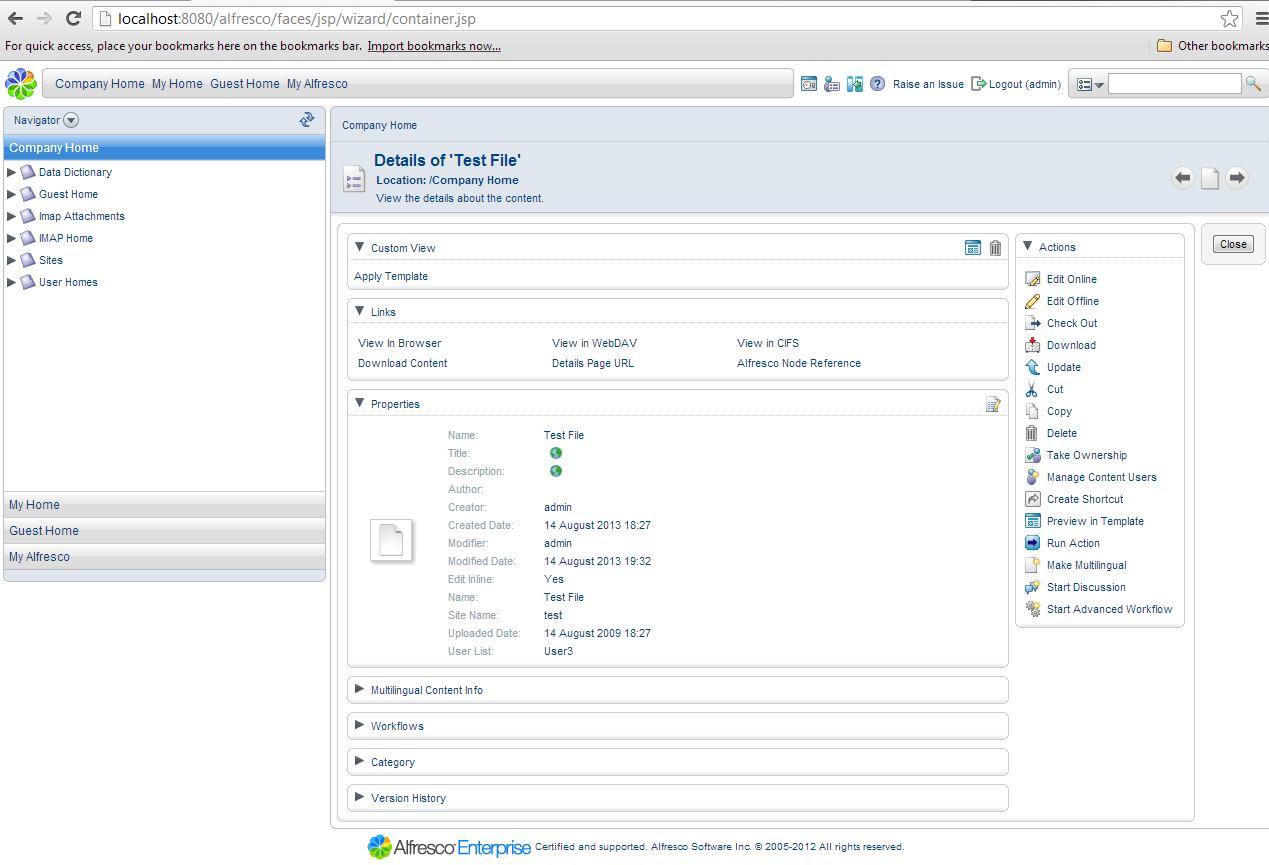
1. Select Action as “Add Aspect” in the dropdown and click on “Click to set values and add to list” button. It will take you to the set action value page where you can see your custom aspect created with name as “Common” in dropdown list as shown below.



1. Select custom aspect from dropdown list, click on the “ok” button and then click on the “Next” button. It will take you to the content details screen as shown below.



1. In the below pasted screen shot you would be able to see all the properties of the aspect “Global logic Action” created.



# Creating Model for Share (<http://localhost:8080/share>)

Content model would be same as Explorer site

C:\Alfresco\tomcat\shared\classes\alfresco\web-extension\share-config-custom.xml

**Creating Custom Action in Explorer and Share**

**Overview**

Custom Action is required to perform certain task against file or folder in repository based on business requirement.

**Example**

Assume that Alfresco SDK is configured in your local Machine with Eclipse IDE. If not done, please follow instruction from <http://wiki.alfresco.com/wiki/Alfresco_SDK_4.0>

Here is the example for creating custom action will show how to add a no-parameter action to the web client.

Our Requirement for Alfresco Explorer (http://localhost:8080/alfresco) and Share site (<http://localhost:8080/share>)

*While uploading the file, the site name should be set in site name property of uploaded file.*

**Implementing Action class**

**import** java.io.Serializable;

**import** java.util.List;

**import** org.alfresco.model.ContentModel;

**import** org.alfresco.repo.action.executer.ActionExecuterAbstractBase;

**import** org.alfresco.service.cmr.action.Action;

**import** org.alfresco.service.cmr.action.ParameterDefinition;

**import** org.alfresco.service.cmr.repository.ChildAssociationRef;

**import** org.alfresco.service.cmr.repository.NodeRef;

**import** org.alfresco.service.cmr.repository.NodeService;

**import** org.alfresco.service.cmr.repository.Path;

**import** org.alfresco.service.cmr.repository.Path.ChildAssocElement;

**import** org.alfresco.service.cmr.repository.Path.Element;

**import** org.alfresco.service.namespace.QName;

**import** org.apache.commons.logging.Log;

**import** org.apache.commons.logging.LogFactory;

**public** **class** CustomAction **extends** ActionExecuterAbstractBase {

**private** **static** **final** Log *LOG* = LogFactory.*getLog*(CustomAction.**class**);

**private** **static** **final** QName *SITE\_NAME* = QName.*createQName*("sd.content.model", "siteName");

**private** NodeService nodeService;

**public** NodeService getNodeService() {

**return** nodeService;

}

**public** **void** setNodeService(NodeService nodeService) {

**this**.nodeService = nodeService;

}

@Override

**protected** **void** executeImpl(Action action, NodeRef nodeRef) {

**try** {

**if** (nodeRef != **null**) {

Path path = nodeService.getPath(nodeRef);

Element element = path.get(3);

ChildAssocElement childAssocElement = (ChildAssocElement) element;

ChildAssociationRef childAssociationRef = childAssocElement.getRef();

QName qName = childAssociationRef.getQName();

nodeService.setProperty(nodeRef, *SITE\_NAME*, qName.getLocalName());

*LOG*.info("Properties are set successfully");

}

} **catch** (Exception e) {

*LOG*.error("Error during setting properties",e);

}

}

@Override

**protected** **void** addParameterDefinitions(List<ParameterDefinition> arg0) {

// **TODO** Auto-generated method stub

}

}

As you can see there is two methods are overrides in CustomAction class

1. **protected** **void** executeImpl(Action action, NodeRef nodeRef)

 This will take the instance of the action and the node and perform the work specified. We can write our logic for the task which is going to be performed after executing this Action

1. **void** addParameterDefinitions(List<ParameterDefinition> arg0)

This gives details of the name of the action and details of the parameters that it is expecting.

Create a jar file via ant build.xml and place that jar file into ‘‘C:\Alfresco\tomcat\webapps\alfresco\WEB-INF\lib’

**Registering the Action**

To register CustomAction class to define bean in ‘**customAction-context.xml**’ file, add following below code and place this file at this location ‘C:\Alfresco\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\module\customAction-context.xml’

<bean id=*"customAction"* class=*"com.sd.ecm.CustomAction"* parent=*"action-executer"*>

<property name=*"nodeService"*>

<ref bean=*"nodeService"* />

</property>

</bean>

**Localization**

Create customAction.properties and place at same location at ‘C:\Alfresco\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\module\customAction.properties’

Add following below code in customAction.properties file

customAction.title=Custom Action

Register customAction.properties file in customAction-context.xml

<bean id=*"custom-action"* parent=*"actionResourceBundles"*>

<property name=*"resourceBundles"*>

<list>

<value>alfresco.module.custom-action</value>

</list>

</property>

</bean>

**Alfresco Module Package (AMP)**

An AMP is a bundle of code, content model, content and the directory structure that is used to distribute additional functionality for Alfresco. Use the *Module Management Tool (MMT) to install and manage AMP files.*

*Follow the below mentioned steps to use MMT*.

1. Run following command to install AMP file.

java -jar C:\Alfresco\bin\alfresco-mmt.jar install C:\programs\FullalfrescoSet\amp\target\amp.amp C:/Alfresco/tomcat/webapps/alfresco.war –verbose

usage: install <AMPFileLocation> <WARFileLocation> [options]

valid options:

-verbose : enable verbose output

-directory : indicates that the amp file location specified is a directory.

All amp files found in the directory and its sub directories are installed.

-force : forces installation of AMP regardless of currently installed module version

-preview : previews installation of AMP without modifying WAR file

-nobackup : indicates that no backup should be made of the WAR

**List**

usage: list <warFile>

Example : java -jar C:\Alfresco\bin\alfresco-mmt.jar list C:/Alfresco/tomcat/webapps/alfresco.war

Lists the details about all the modules currently installed in the WAR file specified. The output is directed to the console.

**Disable**

disable <moduleId> <warFile>

**Enable**

usage: enable <moduleId> <warFile>

**Uninstall**

usage: uninstall<moduleId> <warFile>

java -jar C:\Alfresco\bin\alfresco-mmt.jar uninstall amp C:/Alfresco/tomcat/webapps/alfresco.war

deploy modules in share site

<http://localhost:8080/share/page/modules/deploy>

**Alfresco Cloud**

1. Register in alfresco cloud site to get access.
2. you can register for one or more application. There is no limit for deploying application
3. If user will not use application for one week then refresh token will expire

Alfresco customisation

<http://customizealfresco.blogspot.in/>

For creating alfresco share type template

<http://my-coding-customize-alfresco-share.blogspot.in/2011/09/create-custom-site-type-in-alfresco.html>

Alfresco share customisation

<http://syeoopl.wordpress.com/>

for providing admin access on share sites menu

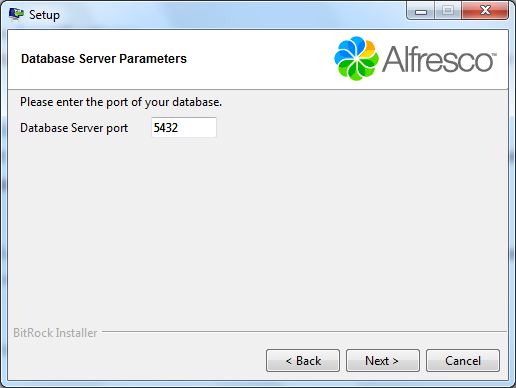
<http://unretoconalfresco.blogspot.in/2012/04/continuando-con-ocultar-la-opcion-crear.html>

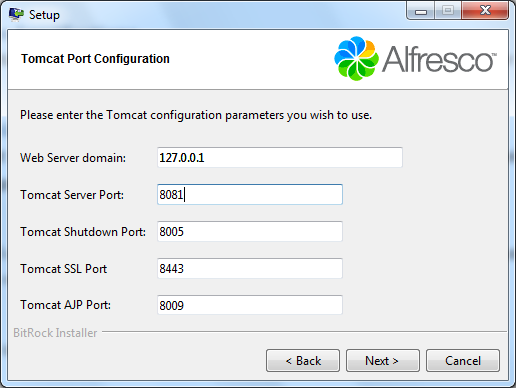
adding document library actions

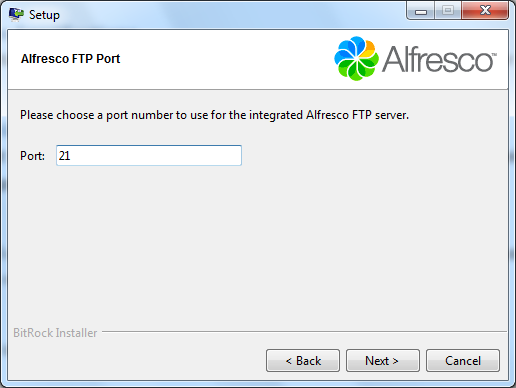
<http://ecmstuff.blogspot.in/2012/04/adding-document-library-actions-in.html>

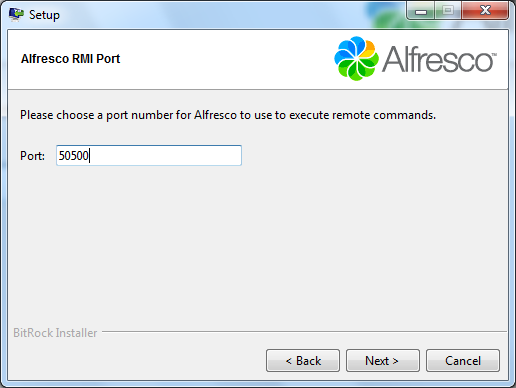
<http://ecmstuff.blogspot.in/2011/02/creating-alfresco-share-sites-with.html>

**Alfresco Installation**

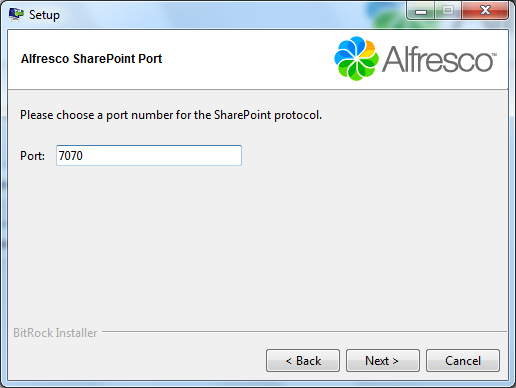
****

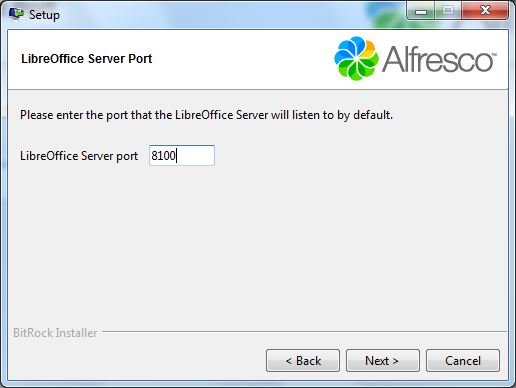
****

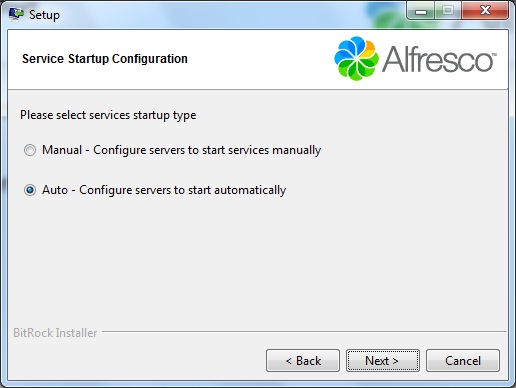
****

****

****

****

****

****