EBook Gratis

APRENDIZAJE alfresco

Free unaffiliated eBook created from
Stack Overflow contributors.

#alfresco

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: alfresco

It is an unofficial and free alfresco ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official alfresco.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con al aire libre

Observaciones

Ediciones disponibles

Edición	Soporte comercial	Listo para producción	Vendedor	Última fecha de lanzamiento
Alfresco Uno	Sí	Sí	Alfresco Software Inc.	
Edición comunitaria al aire libre	No	No	Alfresco Software Inc.	
LXComunidad ECM	Sí	Sí	Loftux AB	

Examples

Instalación o configuración

Instalaciones de desarrollo y evaluación.

Alfresco proporciona varios instaladores diferentes para diferentes sistemas operativos y plataformas. Sin embargo, estos instaladores no se recomiendan para entornos de producción.

https://www.alfresco.com/alfresco-community-download

https://sourceforge.net/projects/alfresco - más archivos de instalación de Alfresco, y módulos separados disponibles también allí

Instalaciones de producción

Puede instalar Alfresco en las distribuciones de Ubuntu utilizando el script de instalación de Alfresco Ubuntu de Loftux AB.

Mientras realiza la instalación con el script, puede seleccionar entre Alfresco Community Edition (sin soporte comercial) y LXCommunity ECM (con soporte comercial).

https://loftux.com/en/products-and-add-ons/alfresco-utilities

Addons

Además, puede extender Alfresco con complementos creados por la comunidad; están disponibles en https://addons.alfresco.com

Sobre alfresco

Alfresco es un sistema ECM (gestión de contenido empresarial), basado en frameworks de java y javascript. Alfresco (como "repositorio" es el núcleo base de Alfresco como producto).

Proporciona por ejemplo. almacén de contenido cifrado, configuración de permisos específicos, tipos de contenido con propiedades específicas (como nombre, fecha de creación, también se pueden incluir propiedades personalizadas).

Alfresco también proporciona procesos de flujo de trabajo, por separado o con documentos, los flujos de trabajo están especificados por el flujo del diagrama (el motor Activiti y jbpm es compatible; jbpm está en desuso).

El producto también admite en parte Sharepoint (a través de los servicios de oficina de Alfresco "aos", CIFS y otras características. Las características clave son:

- Almacenamiento de contenido
- Recuperación de contenido
- Modelado de contenido
- Interfaz de consulta
- Control de acceso
- Auditoría
- Versiones

El contenido del usuario se envuelve con la capa de seguridad: el contenido binario se guarda en el disco, pero la estructura y los metadatos se guardan en la base de datos.

Este "repositorio" de parte de Alfresco como producto proporciona la API REST para las extensiones como Alfresco Share.

Enlace de instalación de Alfresco para Windows

http://docs.alfresco.com/community/tasks/simpleinstall-community-win.html

Para linux

http://docs.alfresco.com/community/tasks/simpleinstall-community-lin.html

Breve descripción sobre la instalación

http://docs.alfresco.com/community/concepts/install-community-intro.html

Lea Empezando con al aire libre en línea:
https://riptutorial.com/es/alfresco/topic/3748/empezando-con-al-aire-libre

Capítulo 2: Administración

Examples

Comenzando y Parando

Para iniciar Alfresco:

1. Cambiar al usuario al aire libre
2. Cambie al directorio $ ALFRESCO_HOME
3. Ejecutar `./alfresco.sh start`

Para detener Alfresco:

1. Cambiar al usuario al aire libre
2. Cambie al directorio $ ALFRESCO_HOME
3. Ejecutar `./alfresco.sh start`

Explotación florestal

Los registros de Alfresco se encuentran en $ ALFRESCO_HOME / tomcat / logs / catalina.out.

Copias de seguridad

Hay muchas maneras de hacer una copia de seguridad de un sistema Alfresco. Es importante que realice una copia de seguridad de la base de datos, así como del almacén de contenido. También es posible que desee realizar una copia de seguridad de los índices de Solr.

Suponiendo que instaló utilizando el instalador binario y todo vive en $ ALRESCO_HOME, puede hacer una copia de seguridad de la base de datos de esta manera:

1. Deja de alfresco
2. Cambiar al usuario al aire libre
3. Cambie al directorio $ ALFRESCO_HOME / postgresql / bin
4. `./pg_dump alfresco --user alfresco > $ALFRESCO_HOME/alf_data/db-backup.sql` la base de datos con `./pg_dump alfresco --user alfresco > $ALFRESCO_HOME/alf_data/db-backup.sql` . Es posible que se le solicite una contraseña. Debe ser lo mismo que la contraseña de administrador que proporcionó durante la instalación. Si no, verifique $ ALFRESCO_HOME / tomcat / shared / classes / alfresco-global.properties para la contraseña de la base de datos.

Ahora tienes tu base de datos respaldada. Es importante hacer eso primero. El siguiente paso es hacer una copia de seguridad del almacén de contenido.

1. Cambie al directorio $ ALFRESCO_HOME / alf_data / contentstore.deleted.
2. Eliminar todo aquí. No hay razón para mantener esos archivos y no hay razón para respaldarlos.

3. Cambie al directorio $ ALFRESCO_HOME / alf_data.
4. Arregle todo el asunto, que también incluirá la copia de seguridad de la base de datos que creó en el paso anterior, asumiendo que la colocó en el directorio alf_data. Ejecute `tar czvf ~/alfresco-backup.tar.gz .`.

El contenido de ese archivo TAR ahora tiene todo lo que necesita para restaurar su sistema de trabajo.

Revisión de cuentas

La auditoría es una característica de Alfresco que permite al usuario rastrear y registrar algunos eventos específicos durante el uso de la plataforma ECM.

Habilitar la auditoría

Para habilitar la auditoría, debe agregar algunas líneas de configuración `alfresco-global.properties` archivo `alfresco-global.properties` , que reside en `tomcat/shared/classes/`

```
audit.enabled = true
audit.alfresco-access.enabled=true
```

Debe guardar los cambios en el archivo `alfresco-global.properties` y reiniciar el servidor Alfresco para habilitar la auditoría.

Auditoría de la configuración por defecto

Aquí está la lista completa de las propiedades de configuración que se pueden reemplazar modificando el archivo `alfresco-global.properties` :

```
# Audit configuration

audit.enabled=true
audit.tagging.enabled=true
audit.alfresco-access.enabled=false
audit.alfresco-access.sub-actions.enabled=false
audit.cmischangelog.enabled=false
audit.dod5015.enabled=false

# Setting this flag to true will force startup failure when invalid audit configurations are
detected

audit.config.strict=false

# Audit map filter for AccessAuditor - restricts recorded events to user driven events. In
this case it neglect events issued by a System or a null user, the content or folder path is
under /sys:archivedItem or under /ver: and the node type is not cm:folder, cm:content or
st:site

audit.filter.alfresco-access.default.enabled=false
audit.filter.alfresco-access.transaction.user=~System;~null;.*
audit.filter.alfresco-access.transaction.type=cm:folder;cm:content;st:site
audit.filter.alfresco-access.transaction.path=~/sys:archivedItem;~/ver:;.*

#The default to preserve all cm:auditable data on a node when the process is not directly
```

```
driven by a user action

system.auditableData.preserve=${system.preserve.modificationData}

#Specific control of how the FileFolderService treats cm:auditable data when performing moves

system.auditableData.FileFolderService=${system.auditableData.preserve}

#Specific control of whether ACL changes on a node trigger the cm:auditable aspect

system.auditableData.ACLs=${system.auditableData.preserve}
```

Como es habitual, debe guardar los cambios en el archivo alfresco-global.properties y reiniciar el servidor Alfresco para habilitar estas modificaciones.

Filtros de auditoria

Los filtros de auditoría son propiedades que especifican la estrategia utilizada para filtrar eventos de auditoría mediante el uso de expresiones regulares particulares para incluir o excluir eventos. Los filtros de auditoría personalizados y predeterminados se pueden agregar como anulaciones en el archivo de configuración `alfresco-global.properties` .

La anatomía de una propiedad de filtro de auditoría es la siguiente:

```
audit.filter.<data_producer>.<path>
```

donde `<data-producer>` es uno de los productores de datos integrados de Alfresco:

1. `alfresco-access` : un amplio grupo de eventos de alto nivel, como inicios de sesión (exitosos y fallidos), actualizaciones de propiedades, CRUD en nodos, lecturas / actualizaciones de contenido, adición y eliminación de aspectos, control de versiones, operaciones de check-in / check-out
2. `alfresco-node`
3. `alfresco-api` : eventos emitidos por la llamada de métodos y servicios API de bajo nivel. Por ejemplo, se puede usar para enumerar los parámetros de la lista de búsqueda de SearchServices, la lista de propiedades usando PropertyServices, las operaciones en nodos que usan NodeServices y así sucesivamente.

y la `path` es el valor de la ruta real para filtrar.

Los nombres de propiedad tienen un prefijo audit.filter. * Y usan '.' como separador donde los componentes de rootPath y las claves en el mapa de auditoría usan '/'.

Las listas se evalúan de izquierda a derecha y, si no se hacen coincidencias al final de la lista, se rechaza el valor. Si no hay una propiedad para un valor determinado o si se define una lista vacía, se acepta cualquier valor.

Cada expresión regular en la lista está separada por un punto y coma (';'). Las expresiones que incluyen un punto y coma pueden escaparse usando un ''.

Tenga en cuenta que si el indicador `audit.config.strict` se establece en verdadero, el inicio de

Alfresco fallará en caso de que las configuraciones de auditoría no sean válidas.

Una expresión que comienza con un `'~'` indica que se debe rechazar cualquier valor coincidente. Si el primer carácter de una expresión debe ser un `'~'` , se puede escapar con un `'\'` .

Agregar `.*` Al final de un filtro incluirá todos los valores que no se hayan excluido específicamente

Los filtros pueden ser uno de los siguientes:

`transaction.user` : especifica qué acciones de los usuarios no se auditarán. Por ejemplo: se auditarán las acciones de todos los usuarios excepto 'Sistema'

`transaction.type` : se `transaction.type` acciones que se realicen contra el tipo de documento especificado.

`default.path` : las acciones que se producen en los documentos dentro de la ruta especificada se auditarán

`transaction.action` : especifica qué acciones se auditarán y cuáles no. Algunos de los eventos de auditoría que pueden habilitarse o inhabilitarse usando esta propiedad son: LEER, MOVIMIENTO, COPIAR, COMPROBAR, COMPROBAR, CANCELAR SALIR, CREAR VERSIÓN, leerContenido, añadirNodoAspectar, eliminarNodoAvisionar, actualizarNodoPropiedades.

Para más información sobre filtros de auditoría:

https://github.com/tsgrp/OpenContent/wiki/Alfresco-Audit-Configuration

http://docs.alfresco.com/5.1/concepts/audit-example-filter.html

Lea Administración en línea: https://riptutorial.com/es/alfresco/topic/6271/administracion

Capítulo 3: Comportamiento y política

Examples

Archivo de versión automática si existe con el mismo nombre

Si el archivo existe con el mismo nombre, se actualizará con una nueva versión.

Para el registro de clase de bean en el archivo `service-context.xml`

```
<bean id="autoVersionByNameBehaviour" class="test.demoamp.AutoVersionByNameBehaviour" init-
method="init">
     <property name="policyComponent" ref="policyComponent"/>
     <property name="nodeService" ref="NodeService"/>
     <property name="contentService" ref="ContentService"/>
     <property name="siteService" ref="SiteService" />
     <property name="fileFolderService" ref="FileFolderService"/>

     <property name="activityService" ref="activityService"/>
     </bean>
```

y la clase de java

```
import java.net.URLEncoder;

import org.alfresco.model.ContentModel;
import org.alfresco.repo.node.NodeServicePolicies;
import org.alfresco.repo.policy.Behaviour;
import org.alfresco.repo.policy.JavaBehaviour;
import org.alfresco.repo.policy.PolicyComponent;
import org.alfresco.service.cmr.activities.ActivityService;
import org.alfresco.service.cmr.model.FileFolderService;
import org.alfresco.service.cmr.model.FileInfo;
import org.alfresco.service.cmr.repository.ChildAssociationRef;
import org.alfresco.service.cmr.repository.ContentReader;
import org.alfresco.service.cmr.repository.ContentService;
import org.alfresco.service.cmr.repository.ContentWriter;
import org.alfresco.service.cmr.repository.NodeRef;
import org.alfresco.service.cmr.repository.NodeService;
import org.alfresco.service.cmr.site.SiteInfo;
import org.alfresco.service.cmr.site.SiteService;
import org.apache.commons.io.FilenameUtils;
import org.json.JSONStringer;
import org.json.JSONWriter;

public class AutoVersionByNameBehaviour
implements NodeServicePolicies.OnCreateNodePolicy {
    private PolicyComponent policyComponent;
    private NodeService nodeService;
    private ContentService contentService;
    private ActivityService activityService;
    private SiteService siteService;
    private FileFolderService fileFolderService;

    public void init() {
```

```
        this.policyComponent.bindClassBehaviour(NodeServicePolicies.OnCreateNodePolicy.QNAME,
ContentModel.TYPE_CONTENT, (Behaviour)new JavaBehaviour((Object)this, "onCreateNode",
Behaviour.NotificationFrequency.TRANSACTION_COMMIT));
    }

    public void onCreateNode(ChildAssociationRef childAssocRef) {
        NodeRef previouslyExistentDoc;
        NodeRef uploadedNodeRef = childAssocRef.getChildRef();
        if (this.nodeService.exists(uploadedNodeRef) && this.isContentDoc(uploadedNodeRef) &&
(previouslyExistentDoc = this.existedPreviousDocument(uploadedNodeRef)) != null) {
            ContentReader reader = this.contentService.getReader(uploadedNodeRef,
ContentModel.PROP_CONTENT);
            ContentWriter writer = this.contentService.getWriter(previouslyExistentDoc,
ContentModel.PROP_CONTENT, true);
            writer.putContent(reader);
            this.nodeService.addAspect(uploadedNodeRef, ContentModel.ASPECT_HIDDEN, null);
            this.postActivityUpdated(previouslyExistentDoc);
            this.nodeService.deleteNode(uploadedNodeRef);
        }
    }

    private void postActivityUpdated(NodeRef nodeRef) {
        SiteInfo siteInfo = this.siteService.getSite(nodeRef);
        String jsonActivityData = "";
        try {
            JSONWriter jsonWriter = new JSONStringer().object();
            jsonWriter.key("title").value((Object)this.nodeService.getProperty(nodeRef,
ContentModel.PROP_NAME).toString());
            jsonWriter.key("nodeRef").value((Object)nodeRef.toString());
            StringBuilder sb = new StringBuilder("document-details?nodeRef=");
            sb.append(URLEncoder.encode(nodeRef.toString(), "UTF-8"));
            jsonWriter.key("page").value((Object)sb.toString());
            jsonActivityData = jsonWriter.endObject().toString();
        }
        catch (Exception e) {
            throw new RuntimeException(e);
        }
        FileInfo fileInfo = this.fileFolderService.getFileInfo(nodeRef);
        this.activityService.postActivity("org.alfresco.documentlibrary.file-updated",
siteInfo == null ? null : siteInfo.getShortName(), siteInfo == null ? null :
"documentLibrary", jsonActivityData, null, fileInfo);
    }

    private boolean isContentDoc(NodeRef nodeRef) {
        return
this.nodeService.getType(this.nodeService.getPrimaryParent(nodeRef).getParentRef()).isMatch(ContentMode

    }

    private NodeRef existedPreviousDocument(NodeRef currentNodeRef) {
        String fileName =
AutoVersionByNameBehaviour.cleanNumberedSuffixes(this.nodeService.getProperty(currentNodeRef,
ContentModel.PROP_NAME).toString());
        if (!fileName.equals(this.nodeService.getProperty(currentNodeRef,
ContentModel.PROP_NAME).toString())) {
            NodeRef folder = this.nodeService.getPrimaryParent(currentNodeRef).getParentRef();
            return this.nodeService.getChildByName(folder, ContentModel.ASSOC_CONTAINS,
fileName);
        }
        return null;
    }
```

```
    public static String cleanNumberedSuffixes(String fileName) {
        String cleanedFileName = fileName;
        String baseName = FilenameUtils.getBaseName((String)fileName);
        if (baseName.indexOf("-") != -1 &&
AutoVersionByNameBehaviour.isInteger(baseName.substring(baseName.lastIndexOf("-") + 1,
baseName.length()))) {
            return baseName.substring(0, baseName.lastIndexOf("-")) +
FilenameUtils.EXTENSION_SEPARATOR_STR + FilenameUtils.getExtension((String)fileName);
        }
        return cleanedFileName;
    }

    public static boolean isInteger(String s) {
        boolean isValidInteger = false;
        try {
            Integer.parseInt(s);
            isValidInteger = true;
        }
        catch (NumberFormatException var2_2) {
            // empty catch block
        }
        return isValidInteger;
    }

    public void setPolicyComponent(PolicyComponent policyComponent) {
        this.policyComponent = policyComponent;
    }

    public void setNodeService(NodeService nodeService) {
        this.nodeService = nodeService;
    }

    public void setContentService(ContentService contentService) {
        this.contentService = contentService;
    }

    public void setActivityService(ActivityService activityService) {
        this.activityService = activityService;
    }

    public void setSiteService(SiteService siteService) {
        this.siteService = siteService;
    }

    public void setFileFolderService(FileFolderService fileFolderService) {
        this.fileFolderService = fileFolderService;
    }
```

}

Lea Comportamiento y política en línea:
https://riptutorial.com/es/alfresco/topic/9696/comportamiento-y-politica

Capítulo 4: Modelo al aire libre con lista dinámica.

Examples

Ejemplo basico

content-model.xml :

```
....
<constraints>
    <constraint name="my:aConstraintList"
type="x.y.z.project.model.constraint.AConstraintList">
    </constraint>
....
<property name="my:aValue">
    <title>My Value</title>
    <type>d:text</type>
    <constraints>
          <constraint ref="my:aConstraintList"></constraint>
    </constraints>
 </property>
```

y la clase de Java declarada antes:

```
public class AConstraintList extends ListOfValuesConstraint implements Serializable {

    ....
    @Override
    public final List<String> getAllowedValues() {
        // Return here the list of values. Enum, call a webservice, etc.
    }

     @Override
     public final String getDisplayLabel(final String value, final MessageLookup
messageLookup) {
         // Return here the label for the value
     }
}
```

Lea Modelo al aire libre con lista dinámica. en línea:
https://riptutorial.com/es/alfresco/topic/6272/modelo-al-aire-libre-con-lista-dinamica-

Capítulo 5: Script T-SQL para crear la base de datos Alfresco en SQLServer 2008 - 2014

Introducción

Es útil tener una secuencia de comandos T-SQL para crear y configurar una nueva base de datos y un usuario para la instalación de Alfresco. Es aburrido y lleva mucho tiempo saltar muchas páginas en MSDN.

A continuación presento el guión:

Observaciones

Las pautas para preparar Alfresco Instalation para una base de datos SQLServer, se pueden leer aquí:

- [http://docs.alfresco.com/3.4/tasks/sqlserver-config.html◆◆1]

Examples

T-SQL_script_4_alfresco

```
/* creates a database for Alfresco, on SQLServer 2008- 2014 */
use master;
GO
CREATE DATABASE alfresco;
GO
/* creates a new LOGIN and associated User
use alfresco;
GO
CREATE LOGIN alfresco WITH PASSWORD = 'alfresco';
GO
use alfresco;
go
CREATE USER alfresco FOR LOGIN alfresco;
GO

/* Now try to add alfresco user to the db_owner Role
   NOTICE: coinnect to alfresco database before
   you can also connect as a local Windows user,
   in order to successfully execute the followings:  */
use alfresco;
GO
EXEC sp_addrolemember N'db_owner', N'alfresco';
GO

/* sets Isolation level */
ALTER DATABASE alfresco SET ALLOW_SNAPSHOT_ISOLATION ON;
GO
```

```
/* creates and sets the alfesco schema as the default one */
use alfresco;
go
CREATE SCHEMA alfresco AUTHORIZATION alfresco;
GO
ALTER USER alfresco WITH DEFAULT_SCHEMA = alfresco;
GO

/* tests table creation */
drop table _buttamiVia_;
GO
create table _buttamiVia_
( id int not null );
GO
```

Lea Script T-SQL para crear la base de datos Alfresco en SQLServer 2008 - 2014 en línea: https://riptutorial.com/es/alfresco/topic/9370/script-t-sql-para-crear-la-base-de-datos-alfresco-en-sqlserver-2008---2014

Capítulo 6: Web Scripts

Introducción

Los webscripts son módulos funcionales en Alfresco, que solo pueden mostrar algunas informaciones o también hacer algunas cosas dentro de Alfresco (por ejemplo, ejecutar flujos de trabajo, trabajar con archivos, usuarios, grupos u otras entidades).

Cada webscripts tiene dos partes principales: código (.js, .java) y plantilla de Freemaker (.ftl)

Webscript también puede tener un archivo .properties adicional con cadenas de texto utilizadas en. Las partes se emparejan en el archivo context.xml (lógica de Spring Framework).

Está enlazado a cualquier URL descrita en este archivo.

Examples

Hola World Web Script

Hagamos un script web hola mundo. Los scripts web tienen un descriptor, un controlador y, opcionalmente, una vista. Estos archivos deben seguir una convención de nomenclatura.

Este descriptor se llama helloworld.get.desc.xml.

```
<webscript>
  <shortname>Hello World</shortname>
  <description>Hello world web script</description>
  <url>/example/helloworld?name={nameArgument}</url>
  <authentication>user</authentication>
</webscript>
```

Puede ver que el descriptor declara que este script web se asignará a una URL, "/ example / helloworld", y que requiere la autenticación del usuario. El descriptor también declara un argumento llamado nombre.

Aquí está el controlador. Se llama helloworld.get.js.

```
model.foo = "bar";
```

Este controlador está escrito en JavaScript, pero los controladores también pueden escribirse en Java. Con un poco más de trabajo, también puede escribir controladores en otros idiomas.

Este controlador no hace mucho. Simplemente agrega una nueva variable al modelo llamado "foo" y le da un valor de "barra".

Su controlador tiene acceso a una variedad de variables de ámbito raíz, todas documentadas en la documentación oficial .

Por último, echemos un vistazo a la vista. Se llama helloworld.get.html.ftl

```
<html>
  <body>
    <p>Hello, ${args.name!"name not specified"}!</p>
    <p>Foo: ${foo}</p>
  </body>
</html>
```

Puede ver en el nombre que esta vista se implementa como una plantilla de Freemarker y genera HTML. Esta vista toma el valor de "foo" del modelo y también captura el argumento de nombre que se pasó a la secuencia de comandos web. Si no se especifica un argumento de nombre, la plantilla proporciona un texto predeterminado.

Si desea producir XML o JSON, puede cambiar el nombre y actualizar la implementación de la plantilla en consecuencia.

Despliegue

Los scripts web se pueden implementar en la ruta de clase o cargarse en el repositorio. Por ejemplo, para implementar este script web subiéndolo al repositorio, siga estos pasos:

1. Suba estos tres archivos al diccionario de datos / Extensiones de scripts web
2. Actualice los Web Scripts en http: // localhost: 8080 / alfresco / s / index y haga clic en "Actualizar Web Scripts".
3. Vaya al script web yendo a http: // localhost: 8080 / alfresco / s / example / helloworld? Name = Jeff

Folder Maker: un script web que maneja POST

Hello World Web Script maneja los métodos GET HTTP. ¿Pero qué pasa si quieres crear datos en el servidor? Para eso su script web debe manejar POST.

Aquí hay un ejemplo simple que crea nuevas carpetas en el Hogar de la Compañía. Se invoca al hacer una llamada POST con un cuerpo JSON que se parece a:

```
{'name':'testfolder'}
```

Opcionalmente, puede agregar un título o una descripción a la carpeta pasándolos como parte del cuerpo, como:

```
{
  'name':'testfolder',
  'title':'test title',
  'description':'test description'
}
```

El descriptor se llama foldermaker.post.desc.xml:

```
<webscript>
```

```
  <shortname>Folder Maker</shortname>
  <description>Creates folders</description>
  <family>Examples</family>
  <url>/example/folders</url>
  <format default="json"></format>
  <authentication>user</authentication>
</webscript>
```

El elemento opcional "familia" es una forma conveniente de agrupar los scripts web en el índice del script web. El elemento "formato" declara que este script web devuelve JSON.

El controlador se llama foldermaker.post.json.js:

```
var name = title = desc = null;

var name = json.get('name');

try {
  title = json.get('title');
} catch (err) {}

try {
  desc = json.get('description');
} catch (err) {}

var folder = companyhome.createFolder(name);

var needsSave = false;

if (title != null) {
  folder.properties['cm:title'] = title;
  needsSave = true;
}

if (desc != null) {
  folder.properties['cm:description'] = desc;
  needsSave = true;
}

if (needsSave) {
  folder.save();
}

model.id = folder.nodeRef.toString();
model.name = name;
model.title = title;
model.description = desc;
```

Observe que este controlador tiene "json" en su nombre. Esto le dice a Alfresco que espere una carga útil JSON. Alfresco analizará automáticamente el JSON y lo pondrá en una variable raíz llamada "json".

El controlador toma el nombre, el título y la descripción del JSON y crea la carpeta utilizando la variable de ámbito raíz llamada "companyhome".

Si se pasa un título o una descripción en esas propiedades, se guardan.

Los valores que se pasaron, así como la referencia del nodo de la nueva carpeta, se configuran en el modelo antes de entregar el control a la vista.

La vista se llama foldermaker.post.json.ftl:

```
{
  <#if title??>
  "title": "${title}",
  </#if>
  <#if description??>
  "description": "${description}",
  </#if>
  "id": "${id}",
  "name": "${name}"
}
```

Este freemarker simplemente devuelve los valores establecidos en el modelo como JSON. Es posible que el título y la descripción no siempre estén presentes, por lo que la vista utiliza la comprobación nula de Freemarker incorporada en una declaración if para evitar devolverlos si no se configuraron.

Se puede usar cualquier cliente HTTP para probar este script web. Esto es lo que se vería usando curl:

```
jpotts$ curl -uadmin:admin -H "content-type: application/json" -X POST
"http://localhost:8080/alfresco/s/example/folders" -d "{'name':'testfolder','title':'test
title', 'description':'test desc'}"
{
  "title": "test title",
  "description": "test desc"
  "id": "workspace://SpacesStore/cc26a12f-306b-41f1-a859-668f11fc2a54",
  "name": "testfolder"
}
```

Tenga en cuenta que curl está pasando en las credenciales básicas de autenticación. En este caso está utilizando "admin". Debido a que este ejemplo crea elementos en el Hogar de la empresa, debe usar un usuario que tenga los permisos adecuados para hacerlo.

Este script web no tiene comprobación de errores reales. Si no pasa un nombre o si ingresa un nombre que ya ha sido usado, verá un error.

Lea Web Scripts en línea: https://riptutorial.com/es/alfresco/topic/6066/web-scripts

Creditos

S. No	Capítulos	Contributors
1	Empezando con al aire libre	bhagyas, Community, vikash, xxxvodnikxxx
2	Administración	abarisone, Jeff Potts
3	Comportamiento y política	vikash
4	Modelo al aire libre con lista dinámica.	Akah
5	Script T-SQL para crear la base de datos Alfresco en SQLServer 2008 - 2014	ciroBorrelli
6	Web Scripts	Jeff Potts, xxxvodnikxxx