AMSeT Alfresco SWORD Interface

Developer Guide

November 2009

Table of Contents

What is this document?

This is a deliverable from the JISC Alfresco Management and Security Toolkit (AMSeT) project which was funded from the Repositories: Rapid Innovation strand of the 12/08 JISC Information Environment and e-Research call and ran from 1 April to 30 September 2009.

Who should read this document?

This document is intended for developers who wish to create applications that use a SWORD interface to the Alfresco content management system.

Introduction

This document provides a technical description of the SWORD document deposit interface created for the Alfresco content management system as part of the JISC AMSeT project.

Software Requirements

Java 5.0 or later

NetBeans 6.7.1

Eclipse Java JEE IDE (Galileo).

Alfresco Labs 3.2 SDK

Alfresco Community Edition 3.2 (running on bundled Apache Tomcat 6.0.18)

MySQL 4.x or 5.x.

Development platform details are as follows.

- Hardware: MacBook Pro, Intel Core 2 Duo, 2.8 GHz 4 GB RAM.
- IDE: NetBeans IDE 6.7.1 (Build 200907230233)
- Java: 1.6.0_15; Java HotSpot(TM) 64-Bit Server VM 14.1-b02-90
- System: Mac OS X version 10.6.1.
- GlassFish 2.1

Technologies employed.

- SWORD common core
- Alfresco 3.2 core and repository api
- SpringMVC (Spring 2.0.8)
- JSP 2.1
- JSTL 1.2
- XMLBeans 1.1
- Apache HttpClient 4.0.1
- Content Repository for Java, JSR 170
- SWORD 1.3

AMSet Alfresco SWORD Software

Overview

Although it is possible to insert plain Java into an Alfresco WAR file, the application is wired together using the Spring framework and offers automated installation of extension modules for Spring applications. Hence, the Alfresco SWORD module was written using SpringMVC.

The formal mechanism for adding functionality to Alfresco is to create an Alfresco Module Package (AMP) file and to load this into Alfresco using the special tool provided, the Module Management Tool (MMT). Hence the first step in the development process was to create a SWORD front end Web application using Alfresco jars to provide an API. Then the application was converted to an AMP and installed in the `alfresco.war`.

SWORD is based on the Atom Publishing Protocol and there is a core object hierarchy of `ServiceDocument.Service.Workspace.Collection` where there can be multiple workspaces and collections. The SWORD application must initially return a representation of the `ServiceDocument` object which offers the user a list of details of the collections to which deposits are possible. A document can then be deposited by uploading a file using the HTTP POST operation to the collection URL returned by the service document. AMSeT offers a Web application client for the deposit process. Posting from an application was tested using Apache HttpClient.

The original SWORD project produced a `common-core.jar` with a client JavaServer Pages (JSP) Web application. Although these JSPs were used as HTML templates for the interface the underlying implementation was different in 2 respects: Java Standard Tag Library (JSTL) was used instead of raw Java code in the JSPs; and custom workspace and collection classes were used instead of the `common-core` equivalents due to the the incompatibility of the data structure with a nested bean structure accessible to JSTL expression language.

A `ServiceContext` object is used to hold a `ServiceDetails` object along with the `ServiceDocument`. `ServiceDetails` contains metadata about the service.

Initialization data on the collections available in a service are provided using `ResourceBundle` property files. It is a simple matter to re-implement the interface involved here. For example, a list of collections might be obtained by direct interrogation of the Alfresco repository for folders that are assigned aspects that identify them as SWORD collections. For this project the target SWORD collection was taken to be the `company-home` folder which is created by the default Alfresco installation.

A couple of XML schemas were written to represent the `ServiceDocument` and XMLBeans used to produce the corresponding XML documents.

XMLBeans was also used to create the `entity` document reponse.

Licensing

All AMSeT code is released under the Apache 2 license. Copyright belongs to the University of Leeds.

Main SpringMVC Beans

There are 2 principal functions provided by a SWORD service, to return a service document and to perform a file deposit to a collection. The SpringMVC `DispatcherServlet` passes control to 2 controllers (request handlers) to perform these functions.

A web application route into an Alfresco SWORD collection is handled by the `ServiceDocumentFormController` and the `DepositSubmitFormController`. The latter delegates to a `AlfrescoDepositManager` class which carries out the deposit of the resource and the media link entry. The third handler, `HomePageAbstractController`, populates the home page with some service data.

`DepositSubmitFormController` handles deposit requests that originate from an HTTP form POST (multipart/form-data). A Spring interceptor bean handles non-multipart requests which then skirt the main controller.

AMP Files

The structure of an AMP file is shown below.

```
/
|
|- /config
|
|- /context
   |
   |
   |- /lib
   |
   |- /licenses
   |
   |- /web
   |
   |- /jsp
   |
   |- /css
   |
   |- /images
   |
   |- /scripts
|
|- module.properties
|
|- file-mapping.properties
```

config

The recommended package structure for an Alfresco AMP project is `alfresco.module.projectpackages`. In AMSeT, `alfresco.module.sword` has been used. The directory structure in the `config` directory must follow this package structure. In the directory `config/alfresco/module/sword` is placed the `module-context.xml` file. This contains a Spring beans document that points to the module's Spring configuration files. In this case we have the following.

Listing 1.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'
   'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>
<import
    resource="classpath:alfresco/module/sword/context/service-context.xml" />
</beans>
```

In this `context` directory is the `service-context.xml` file which contains the module's bean definitions:

Listing 2.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'
      'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>

    <!-- SWORD AMP -->

    <bean id="controllerClassNameHandlerMapping"
    class="org.springframework.web.servlet
    .mvc.support.ControllerClassNameHandlerMapping"/>

    <bean id="urlMapping"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
        <property name="mappings">
            <props>
                <prop key="home">homePageAbstractController</prop>
                <prop key="results">resultsController</prop>
            </props>
        </property>
    </bean>

    <bean id="viewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView"/>
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>
    ...
    ...
    ...
```

lib

Jar files for the project.

/web/jsp

Since all the JSP files in this module are hidden under `WEB-INF` this directory is empty. There isn't a default mapping that puts JSP files in `WEB-INF` so a custom mapping has to be introduced by adding an entry in the `file-mapping.properties` file.

module.properties

This is a mandatory file which contains metadata about the project used by the MMT. In this module we have the following.

Listing 3.

```
# SWORD AMP
module.id=sword
module.title=SWORD AMP
module.description=Builds AMP for SWORD interface
module.version=1.0
```

file-mapping.properties

The directory `WEB-INF/jsp` was introduced at the top level of the AMP file and the module jsp files were placed here. In order for the jsps to be placed in the `/WEB-INF/jsp` directory of the application, the following line must be present in the `file-mapping.properties` file.

```
WEB-INF/jsp=/WEB-INF/jsp
```

Loading AMP module with Module Management Tool

The MMT inserts a module into an Alfresco WAR file when the software and associated resources are structured according to the AMP format. [A sample application in the Alfresco Software Development Kit (SDK) is provided whose build file contains a target for automatically converting applications to AMPs and loading modules into a given Alfresco WAR.]

There is a more convenient means of loading an AMP. On installing Alfresco Community Edition 3.2 there is an `amps` directory with an `add-amps-here.txt` file containing the following text.

To help you manage AMP extensions to your Alfresco server, you can place AMP files in this directory and use the apply_amps script to perform the update. Usually, you only need to run the apply_amps script when you add a new AMP to the directory, or when you upgrade your Alfresco server. Whenever the script is run, a backup of the core server file is made (the alfresco.war file).

The MMT tool worked as stated and the Alfresco log (`alfresco.log`) and Spring standard output provided useful error information.

There was one wrinkle in the installation process, however. The module, being a SpringMVC application, requires that the Spring dispatcher servet be declared in the `web.xml` file. The Spring container, seeing the servlet, requires a `dispatcher-servlet.xml` configuration file. At this point, all the AMSeT-SWORD Spring bean definitions were put in this file and placed in the `WEB-INF` directory. Despite some definitions also being present in the `service-context.file` everything worked OK and was left like that.

Compliance with SWORD Profile 1.3

This section follows the structure of the SWORD Profile 1.3 document and provides information on the AMSeT implementation.

Section A. 1.1.

The value of all `q` attributes for the `acceptPackaging` elements have been set to 1. Although Alfresco can accept any type of file as deposit, no package processing will be possible unless custom actions are created.

Section A. 2 Mediated Deposit.

This implementation does not support mediated deposits. This is not required by the 1.3 profile. The `On-Behalf-Of` header/parameter is detected by the code, but not used.

Section A. 2.1.

As required, a mediated element is included in the service document and this is set to *false* to indicate that mediated deposit is not supported. If a non-empty `X-On-Behalf-Of` header or form parameter is received, then an exception is thrown.

Section A. 2.2.

No mediation, so not applicable.

Section A. 3 Developer Features

3.1, 3.2 and 3.3 below are extensions recommended in the SWORD profile v 1.3 in order to ease the development process.

Section A. 3.1. NoOp (Dry run)

`NoOp` is not implemented, but a line or two will do it.

Section A. 3.2. Verbose

The `X-Verbose` header/parameter is processed and a verbose boolean is present in the initialization property file. In order to implement a verbose return, the verbose description will need to be obtained from the property file at initialisation.

Section A. 3.3. Client and Server Identity

`User-Agent` and `Generator` headers implemented and values are obtained from the initialisation properties files.

A 6. Nested Service Description

Nested service urls can be supplied from the initialisation property files.

B 5.1. X-On-Behalf-Of

The `X-On-Behalf-Of` header is processed although the service is not implemented. The service document returns a `mediation = false` statement. This is set at initialization.

B 5.3. Creating a Resource

HTTP response code `201 Created` is returned after a successful deposit together with a `Location` header.

B 5.4. Editing a Resource

Not implemented. Not required by profile.

B 5.5. HTTP Response Codes

Response codes are supplied according to the recommendations of this section. Human readable explanations are also supplied.

7. Category Documents

Not required, not implemented.

8. Service Documents

SWORD parameters `version`, `verbose`, `noOp` and `maxUploadSize` are presented in the service document.

8.1. Workspaces

According to the profile, `sword:collectionPolicy`, `sword:mediation`, `sword:treatment`, `sword:acceptPackaging`, and `sword:service` and `dcterms:abstract` elements are included in the `Collection` element.

9.2. Creating Resources with POST

Content-MD5: implemented, not tested.

User-Agent: the web client sets a user agent header.

Content-Disposition: filename : implemented. If a filename is not present in the `Content-Disposition` header, the `Slug` header will be used if present.

X-On-Behalf-Of: the presence of a value here will provoke a 400 Bad Request response.

Future Work

A few tasks will be be required before pressing AMSeT SWORD into live service.

Compliance Testing

Some unit testing is present, and the return XML validates according to the validator provided by the original SWORD project, but time for testing has been limited.

NoOp and Verbose

The code is there, more or less, but not been switched on.

Authentication and Authorization

Implementation of either Basic Authentication or Java Authentication and Authorisation Specification (JAAS) and aligning this with the Alfresco permission structures. There is a Spring interceptor bean in place for handling authentication. If further security is required, deposit authorization will require a process to retrieve folder permissions for the authenticated user.

Look and Feel

The SWORD JSPs should look like they belong to Alfresco.

XML generation

This is a bit scrappy and could be improved and tied in with either the common-core or with Abdera, the Atom implementation project.

Spring/Alfresco Configuration Files

A bit of rationalisation required involving the contents of the `application-context`, `servlet-context` and `module-context` files. More ugly than dangerous.

Links

AMSeT Wiki: http://amset.leeds.ac.uk:8080/amsetwiki/

AMSeT Blog: http://www.socketelf.org:8080/roller/amset/

AMSeT SourceForge site: http://sourceforge.net/projects/amset/

About this document

This document was created using the Oxygen XML Editor 10.3 using its DocBook 5.0 schema and bundled HTML and PDF XSLT transformations.

Brian P. Clark, November 2009

Please see project wiki [http://amset.leeds.ac.uk:8080/amsetwiki/] for contact details.