Accès nuxeo via REST

Introduction

Nuxeo propose une interface REST. Cf. The Nuxeo Restlet API

Ce mécanisme est particulièrement intéressant dans la mesure où, si les services proposés de base par nuxeo ne sont pas suffisants, il est possible d'en ajouter de nouveaux.

A noter néanmoins que cette document est incomplète et qu'il existe beaucoup de services de bases dans nuxeo. Je vous conseille de rechercher tous les extensions de PluggableRestletService dans les sources de nuxeo pour voir ce qui est disponible. En général le point d'extension est bien documenté. Exemple :

```
<documentation>
  Create a File document via upload
  POST /nuxeo/restAPI/{repoId}/{docId}/{filename}/upload
</documentation>
<restletPlugin
    name="upload"
    class="org.nuxeo.ecm.platform.ui.web.restAPI.UploadRestlet"
    enabled="true"
    useSeam="true">
  <urlPatterns>
    <urlPattern>/{repo}/{docid}/{filename}/upload</urlPattern>
  </urlPatterns>
</restletPlugin>
```

⚠ Il est aussi possible d'utiliser/crée des services REST de/via WebEngine mais ici je me suis limité à utiliser ceux de nuxeo DM

Mise en place du projet

1. **mvn archetype:create -DgroupId=fr.univrennes1.testREST -DartifactId=testREST -DpackageName=fr.univrennes1.testREST**
2. modification su pom.xml pour ajouter les dépendances requises :

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>fr.univrennes1.testCMIS</groupId>
        <artifactId>testREST</artifactId>
        <packaging>jar</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>testCMIS</name>
        <url>http://maven.apache.org</url>
        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.httpcomponents</groupId>
                        <artifactId>httpmime</artifactId>
                        <version>4.0.1</version>
                </dependency>
                <dependency>
                        <groupId>dom4j</groupId>
                        <artifactId>dom4j</artifactId>
                        <version>1.6.1</version>
                </dependency>
                <dependency>
                        <groupId>jaxen</groupId>
                        <artifactId>jaxen</artifactId>
                        <version>1.1.1</version>
                </dependency>
        </dependencies>
</project>
```

3. Création du projet eclipse : **mvn eclipse:eclipse**

Code

```
package fr.univrennes1.testREST;

import java.io.File;
import java.io.IOException;

import org.apache.http.HttpException;
import org.apache.http.HttpHost;
import org.apache.http.HttpRequest;
import org.apache.http.HttpRequestInterceptor;
import org.apache.http.auth.AuthScheme;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.AuthState;
import org.apache.http.auth.Credentials;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.protocol.ClientContext;
import org.apache.http.entity.mime.MultipartEntity;
import org.apache.http.entity.mime.content.FileBody;
import org.apache.http.entity.mime.content.StringBody;
import org.apache.http.impl.auth.BasicScheme;
import org.apache.http.impl.client.BasicResponseHandler;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
```

```
import org.apache.http.protocol.ExecutionContext;
import org.apache.http.protocol.HttpContext;
import org.dom4j.Document;
import org.dom4j.DocumentException;
import org.dom4j.DocumentHelper;

public class Rest
{
    public static void main( String[] args ) throws IOException, DocumentException
    {
            //************* diverses initialisations *************
            Integer port = 8080;
            String host = "localhost";
            String baseURL = "http://" + host + ":" + port + "/nuxeo/restAPI/default/";
            UsernamePasswordCredentials credentials = new UsernamePasswordCredentials("Administrator",
"Administrator");
            AuthScope authScope = new AuthScope(host, port);
        String url = null;
        String rep = null;
        //91ea0088-4545-41cd-a43e-22ce4c1aefda pour le workspace testRB
        String WorkspaceID = "83641c01-c9d4-4bee-a7ff-72c1a9fd051a";

        //************* Suppression de Dossier1 *************
            //GET /nuxeo/restAPI/{repoId}/deleteDocumentByPath?path=/default-domain/workspaces/doc1
        url = baseURL
                    + "deleteDocumentByPath?path=/default-domain/workspaces/testrb/dossier1";
        getUrl(credentials, authScope, url);

        //************* création du dossier "Dossier1" *************
        //GET /nuxeo/restAPI/{repo}/{parentdocid}/createDocument?docType=File
        url = baseURL
                    + WorkspaceID
                    + "/createDocument?docType=Folder&dublincore%3Atitle=Dossier1";
        rep = getUrl(credentials, authScope, url);
                Document document = DocumentHelper.parseText(rep);
                String FolderID = document.selectSingleNode("/document/docRef").getText();

                //************* création du fichier "Fichier1" *************
        url = baseURL
                    + FolderID
                    + "/createDocument?docType=File&dublincore%3Atitle=Fichier1";
        rep = getUrl(credentials, authScope, url);
                document = DocumentHelper.parseText(rep);
                String FileID = document.selectSingleNode("/document/docRef").getText();

                //************* upload du contenu *************
        //POST /nuxeo/restAPI/{repoId}/{docId}/{filename}/upload
        url = baseURL + FileID + "/metro.pdf/upload";
        File targetFile = new File( "/A_GARDER_UN_PEU/nuxeo-dev/workspace/testCMIS/src/main/resources/metro.
pdf" );
        FileBody bin = new FileBody(targetFile);
        StringBody comment = new StringBody("metro.pdf");
        MultipartEntity reqEntity = new MultipartEntity();
        reqEntity.addPart("bin", bin);
        reqEntity.addPart("comment", comment);
        reqEntity.addPart("name", comment);
        HttpPost filePost = new HttpPost(url);
        filePost.setEntity(reqEntity);
        DefaultHttpClient httpclient = new DefaultHttpClient();
        httpclient.getCredentialsProvider().setCredentials(authScope, credentials);

        BasicHttpContext localcontext = new BasicHttpContext();
        // Generate BASIC scheme object and stick it to the local
        // execution context
        BasicScheme basicAuth = new BasicScheme();
        localcontext.setAttribute("preemptive-auth", basicAuth);
        // Add as the first request interceptor
        httpclient.addRequestInterceptor(new PreemptiveAuth(), 0);

        ResponseHandler<String> responseHandler = new BasicResponseHandler();
        rep = httpclient.execute(filePost, responseHandler, localcontext);
```

```
            System.out.println("**********************************");
            System.out.println("url = " + url);
            System.out.println(rep);
    httpclient.getConnectionManager().shutdown();
}

    /**
     * @param credentials
     * @param authScope
     * @param url
     * @throws IOException
     * @throws ClientProtocolException
     */
    private static String getUrl(UsernamePasswordCredentials credentials,
                    AuthScope authScope, String url) throws IOException,
                    ClientProtocolException {
            String ret = null;
            HttpGet httpget = new HttpGet(url);
    DefaultHttpClient httpclient = new DefaultHttpClient();
    httpclient.getCredentialsProvider().setCredentials(authScope, credentials);

    BasicHttpContext localcontext = new BasicHttpContext();
    // Generate BASIC scheme object and stick it to the local
    // execution context
    BasicScheme basicAuth = new BasicScheme();
    localcontext.setAttribute("preemptive-auth", basicAuth);
    // Add as the first request interceptor
    httpclient.addRequestInterceptor(new PreemptiveAuth(), 0);

    ResponseHandler<String> responseHandler = new BasicResponseHandler();
    ret = httpclient.execute(httpget, responseHandler, localcontext);
            System.out.println("**********************************");
            System.out.println("url = " + url);
    System.out.println(ret);
    httpclient.getConnectionManager().shutdown();
    return ret;
    }

static class PreemptiveAuth implements HttpRequestInterceptor {

    public void process(
            final HttpRequest request,
            final HttpContext context) throws HttpException, IOException {

        AuthState authState = (AuthState) context.getAttribute(
                ClientContext.TARGET_AUTH_STATE);

        // If no auth scheme avaialble yet, try to initialize it preemptively
        if (authState.getAuthScheme() == null) {
            AuthScheme authScheme = (AuthScheme) context.getAttribute(
                    "preemptive-auth");
            CredentialsProvider credsProvider = (CredentialsProvider) context.getAttribute(
                    ClientContext.CREDS_PROVIDER);
            HttpHost targetHost = (HttpHost) context.getAttribute(
                    ExecutionContext.HTTP_TARGET_HOST);
            if (authScheme != null) {
                Credentials creds = credsProvider.getCredentials(
                        new AuthScope(
                                targetHost.getHostName(),
                                targetHost.getPort()));
                if (creds == null) {
                    throw new HttpException("No credentials for preemptive authentication");
                }
                authState.setAuthScheme(authScheme);
                authState.setCredentials(creds);
            }
        }

    }

}
```

```
}
```

Remarques

Fichier1 est bien créé dans **Dossier1**. Par contre, **metro.pdf** n'est pas mis en tant que contenu de **Fichier1** mais comme un nouveau fichier (de nom **metr**
) dans le dossier **Dossier1**. Ceci ne semble pas cohérent avec ce que prévoit le service REST. A ce jour, une rapide analyse du code nuxeo ne m'a pas permis de comprendre pourquoi... Visiblement un pb de "Could not instantiate Seam component: pictureBookManager"

la classe **PreemptiveAuth** est utilisée pour passer le user/password dès la première requête. Ceci évite de passer les autre mécanismes d'authentification de nuxeo (invité et cas par exemple).