ATELIER TECHNIQUE #3

Service WEB de type REST en Java

23 septembre 2014

François ANDRE

SEDOO
OMP

SOMAIRE

REST

SERVICES WEB

- **Objectif** : mise à disposition de **services** applicatifs en utilisant l'infrastructure de **Web**.
- **Standards**
  - **SOA**P
  - REST

REST

- **Architecture** basée sur
  - Identification des ressources distribuées par une URL
    Exemple :

    http://portailrbvws.sedoo.fr/rest/Integration/getById/08451ee9-5286-4715-9ab9-91834b24ab5c
  - Utilisation des commandes du protocole HTTP - GET, POST, PUT, DELETE - pour manipuler ces ressources

REST	CRUD
POST	CREATE
GET	READ
PUT	UPDATE
DELETE	DELETE

REST

- **Raisons du succès**
  - Simplicité de mise en œuvre
  - Simplicité d'intégration dans les clients Web.
  - Mécanisme simple de communication entre applications (Web, Shell,...)
- **Limites**
  - Limites des protocoles Web : Same Origin Policy (SOP), Sécurité

DANS LE MONDE JAVA

- **JAX-RS** (Java API for RESTful Services) - (JSR 311)
- Implémentation de référence: **Jersey** 
  - Version : 2.12
  - Contributeur : Oracle
  - Licence : Open-source

MISE EN ŒUVRE DE JERSEY

PREMIERS PAS

GÉNÉRATION SQUELETTE APPLICATIF

Utilisation archetype Maven

```
mvn archetype:generate -DgroupId=fr.sedoo.demo -DartifactId=atelier3
-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

Dépendances (*pom.xml*)

```
...
<dependencies>
    <dependency>
        <groupId>com.sun.jersey</groupId>
        <artifactId>jersey-server</artifactId>
        <version>1.18</version>
    </dependency>
</dependencies>
...
```

CONFIGURATION WEB

Ajout de la servlet Jersey (*web.xml*)

```
...
<servlet>
  <servlet-name>jersey-serlvet</servlet-name>
  <servlet-class>
  com.sun.jersey.spi.container.servlet.ServletContainer
  </servlet-class>
  <init-param>
   <param-name>com.sun.jersey.config.property.packages</param-name>
   <param-value>fr.sedoo.demo.atelier3.service</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>jersey-serlvet</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
...
```

PREMIER SERVICE REST

```
package fr.sedoo.demo.atelier3.service;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

@Path("/first")
public class SecondService {

  @GET
  @Path("/isalive")
  public Response getMsg() {

    String output = "Yes";
    return Response.status(200).entity(output).build();

  }
}
```

RESULTAT

Remarques :

- L'annotation *@Path* n'est pas obligatoire sur une méthode
- Le chemin peut être une expression régulière :
  `@Path("/a:[iI]ntegration")`
- Un service REST retourne un code HTTP et non une exception : 200, 400, 401, 500 - plus rarement 418...

PASSAGE DE PARAMÈTRES

PARAMÈTRES DE CHEMIN : @PATHPARAM

```
@GET
@Path("/iagos/{airport}/{particle}/")
public Response getUserHistory(
            @PathParam("airport") String airport,
            @PathParam("particle") String particle)
    {
        return Response.status(200)
        .entity("Voici la teneur en "+particle+" pour "+airport).build();
  }
```

- Url correspondante : http://localhost:7080/atelier3/rest/first/iagos/BLAGNAC/NO2
- Remarques :
  - L'annotation *@PathParam* peut référencer une partie du chemin de la classe
  - Le type de la variable correspondant peut être :
    - Un type primitif (String, int...)
    - Une classe ayant un constructeur ou une méthode statique *valueOf* ou *fromString* prenant une chaine comme argument
    - Des *PathSegment* pour une analyse plus fine de l'url.

PARAMÈTRES DE REQUÊTE : @QUERYPARAM

```
@GET
    @Path("/iagos/{airport}/{particle}/")
    public Response getAirportInfo(
        @PathParam("airport") String airport,
        @PathParam("particle") String particle,
        @QueryParam("from") String from,
        @QueryParam("to") String to)
    {
        return Response.status(200).entity("Voici la teneur en "+particle+
        "pour "+airport+"  de "+from +" a "+ to).build();
    }
```

- Url correspondante : http://localhost:7080/atelier3/rest/first/i-agos/BLAGNAC/NO2?from=22/01/2014&to=25/03/2015

PARAMÈTRES DE REQUÊTE : @QUERYPARAM

- Remarques :
  - L'utilisation de valeurs par défaut est possible :

    ```
    @DefaultValue("2") @QueryParam("step") int step
    ```
  - Le type de la variable correspondant à *@QueryParam* peut être :
    - Un type primitif (String, int...)
    - Une classe ayant un constructeur ou une méthode statique *valueOf* ou *fromString* prenant une chaine comme argument
    - Des listes de ces éléments (répétition du paramètre dans l'URL)

PARAMÈTRES DE FORMULAIRE : @FORMPARAM

```
@POST
@Path("/addOrUpdateWithId")
@Consumes("application/x-www-form-urlencoded")
public Response addOrUpdateWithId(
    @FormParam("src") String src,
    @FormParam("format") String format,
    @FormParam("login") String login,
    @FormParam("password") String password)
{
...
}
```

Remarques : sur le même principe, il existe aussi

- @CookieParam : valeur stockée dans un cookie.
- @HeaderParam : valeur passée dans le header.
- ...

TYPES RETOURNÉS

TYPES RETOURNÉS : @PRODUCES

Il est possible de spécifier une type de retour (Type MIME).

- Par défaut : text/plain
- Autres types
  - XML : @Produces("application/xml")
  - PDF : @Produces("application/pdf")
  - Image : @Produces("image/*")
  - ...

```
@GET
@Path("/getPdfById/{localeCode}/{uuid}")
@Produces("application/pdf")
public Response getPdfById(@PathParam("localeCode") String localeCode, ...)
{
...
}
```

TYPES RETOURNÉS : @PRODUCES

Exemple de retour de fichier

```
return Response.ok(
new ByteArrayInputStream(FileUtils.readFileToByteArray(tmpFile)))
.build();
```

POINTS DIVERS

WEB APPLICATION DESCRIPTION LANGUAGE (WADL)

URL du WADL (généré automatiquement)

`http://localhost:7080/atelier3/rest/application.wadl`

```
<application xmlns="http://research.sun.com/wadl/2006/10">
  <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 1.8 06/24/2011 12:17 PM" />
▼ <resources base="http://localhost:7080/atelier3/rest/">
    <resource path="/second">
      <resource path="/isalive">
        <method id="getMsg" name="GET">
          <response>
            <representation mediaType="*/*" />
          </response>
        </method>
      </resource>
    </resource>
  ▼ <resource path="/first">
    ▼ <resource path="/isalive">
      ▼ <method id="getMsg" name="GET">
        ▼ <response>
            <representation mediaType="*/*" />
          </response>
        </method>
      </resource>
      <resource path="/iagos/{airport}/{particle}/">
        <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="particle" style="template" type="xs:string" />
        <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="airport" style="template" type="xs:string" />
        <method id="getAirportInfo" name="GET">
          <request>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="from" style="query" type="xs:string" />
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="to" style="query" type="xs:string" />
          </request>
          <response>
            <representation mediaType="*/*" />
          </response>
        </method>
      </resource>
    </resource>
  </resources>
</application>
```

AUTHENTIFICATION

- L'authentification peut être mise en place au moyen de filtres

Paramètre supplémentaire dans la servlet Jersey (web.xml)

```
...
<init-param>
  <param-name>com.sun.jersey.spi.container.ContainerRequestFilters</param-name>
  <param-value>fr.sedoo.demo.atelier3.service.AuthFilter</param-value>
</init-param>
..
```

- Plusieurs schémas d'authentification sont utilisables :
  - HTTP Basic authentication,
  - Oauth2,
  - ...

AUTHENTIFICATION

Filtre d'authentification (AuthFilter.java)

```
...
@Provider
public class AuthFilter implements ContainerRequestFilter
{
  /**
   * Exemple de filtre de sécurité (HTTP Basic Authentication)
   * Si la requête contient le mot "secured" seuls les logins contenant "good" sont acceptés.
   */
  public ContainerRequest filter(ContainerRequest request)
  {
    if (request.getRequestUri().toString().contains("secured")) {
      String auth = request.getHeaderValue("Authorization");
     //Pas d'authentification
 if (auth == null) {
   throw new WebApplicationException(Status.UNAUTHORIZED);
 }
 else {
       String credential = new String(Base64.decodeBase64(auth.getBytes()));
       if (credential.contains("good")) {
         return request;
    }
    else {
    throw new WebApplicationException(Status.UNAUTHORIZED);
    }
 }
    }
else {
  return request;
}
  }
}
..
```

TEST UNITAIRES

DÉPENDANCES SUPPLÉMENTAIRES

Dépendances (*pom.xml*)

```
...
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-client</artifactId>
  <version>1.18</version>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-grizzly2</artifactId>
  <version>1.18</version>
  <scope>test</scope>
 </dependency>
...
```

PREMIER TEST UNITAIRE

Jersey-client

```
...
@Test
public void testGetAirportInfo()
{
  Client client = Client.create();
  String url =
  "http://localhost:7080/atelier3/rest/first/iagos/BLAGNAC/NO2?from=22/01/2014&to=25/03/2015";
  WebResource webResource = client.resource(url);
  ClientResponse response = webResource.accept("text/plain").get(ClientResponse.class);
  Assert.assertTrue("Le code réponse doit être 200",response.getStatus() == 200);
  String output = response.getEntity(String.class);
  Assert.assertTrue("La réponse doit contenir BLAGNAC", output.contains("BLAGNAC"));
}
...
```

- L'exécution du test nécessite le lancement du serveur au préalable

SERVEUR EMBARQUÉ

Lancement/Arrêt du serveur par le test

```
...
final static URI baseUri = UriBuilder.fromUri( "http://localhost/").port( 7080 ).build();
final static String restPath = "rest/";
HttpServer server;

@Before
public void startServer() throws IOException{
  ResourceConfig rc = new PackagesResourceConfig("fr.sedoo.demo.atelier3.service");
  server = GrizzlyServerFactory.createHttpServer(baseUri + restPath, rc);
}

@After
public void stopServer() {
  server.stop();
}
...
```

SERVEUR EMBARQUÉ

Code du test modifié

```
...
@Test
public void testGetAirportInfo()
{
  Client client = Client.create();
  String airportName = "BLAGNAC";
  String url =
  baseUri.toString()+restPath+"first/iagos/"+airportName+"/NO2?from=22/01/2014&to=25/03/2015";
  WebResource webResource = client.resource(url);
  ClientResponse response = webResource.accept("text/plain").get(ClientResponse.class);
  Assert.assertTrue("Le code réponse doit être "+HttpStatus.OK_200,
  response.getStatus() == HttpStatus.OK_200.getStatusCode());
  String output = response.getEntity(String.class);
  Assert.assertTrue("La réponse doit contenir "+airportName, output.contains(airportName));
}
...
```

EXEMPLES

PORTAIL RBV

EXPERIMENTAL SITE **Observations and measurements made by experimental site CAPESTERRE** *(Added or modified on 2014-09-16)*

This site is located at the outlet of the Capesterre catchment. Anthropogenic influence on this site, located upstream of agricultural areas, is moderate. A series of automatic instruments (a meteorological station, 3 pressure gauges, 2 turbidimeters, a Lisst-Streamsside, a conductivimeter, an automatic water-sampler, 2 temperature probes, a rain water collector) allow the real time monitoring of meteorological parameters, the chemistry of atmospheric deposits, the river flow rate, the suspended load, the chemical composition of the river (pH, conductivity, major elements, dissolved organic and inorganic carbon, ...), of soil solutions and of the suspended load. Regular aerial image and terrestrial lidar acquisition are used to monitor the evolution of the river morphology.

VIEW PRINT XML

OBSERVATORY : OBSERA **Observations and measurements made by observatory OBSERA** *(Added or modified on 2014-09-16)*

The Earth surface evolves under the action of geological, chemical, physical, biological and anthropogenic processes involving a wide range of time and length scales (from the meter and the second up to the thousand of kilometers and the million years). These processes control the evolution of soils, the shape of landscapes and the coupling between climate, tectonics and erosion. Understanding them requires to monitor experimental catchments over durations long enough to capture all the time scales involved. The Observatory of Erosion in the Antilles (ObsErA) is a new observatory created in january 2011 by the CNRS-INSU to address these problematics. It is operated by the Institut de Physique du Globe de Paris (IPGP) and involves a team of 12 permanent technicians, engineers and researchers belonging to 3 different institutes: IPGP and the Observatoire Volcanologique et Sismologique de Guadeloupe (OVSG), the University of Bretagne Occidentale (UBO) and the Laboratoire des Sciences de l[...]

VIEW PRINT XML

▸ *Snapshot(s)*

PORTAIL RBV - WS

Web Service de métadonnées (v 0.0.7)

Un service web a été mis en place pour permettre une alimentation automatique du catalogue RBV. Il est destiné aux observatoires vérifiant les deux conditions ci-dessous :

- Ils possèdent un système d'informations contenant la plupart des métadonnées recensées dans la fiche de métadonnées (potentiellement via un catalogue de métadonnées).
- Ils ne possèdent pas de mécanisme de moissonage de leur catalogue de métadonnées.

Ce web service se base sur un format XML reprenant le contenu de la fiche de métadonnées. Il est particulièrement simplifié par rapport au format ISO19139 et est donc plus facile à générer.

Test du web service

Le web service peut être testé manuellement ici.

Documentation

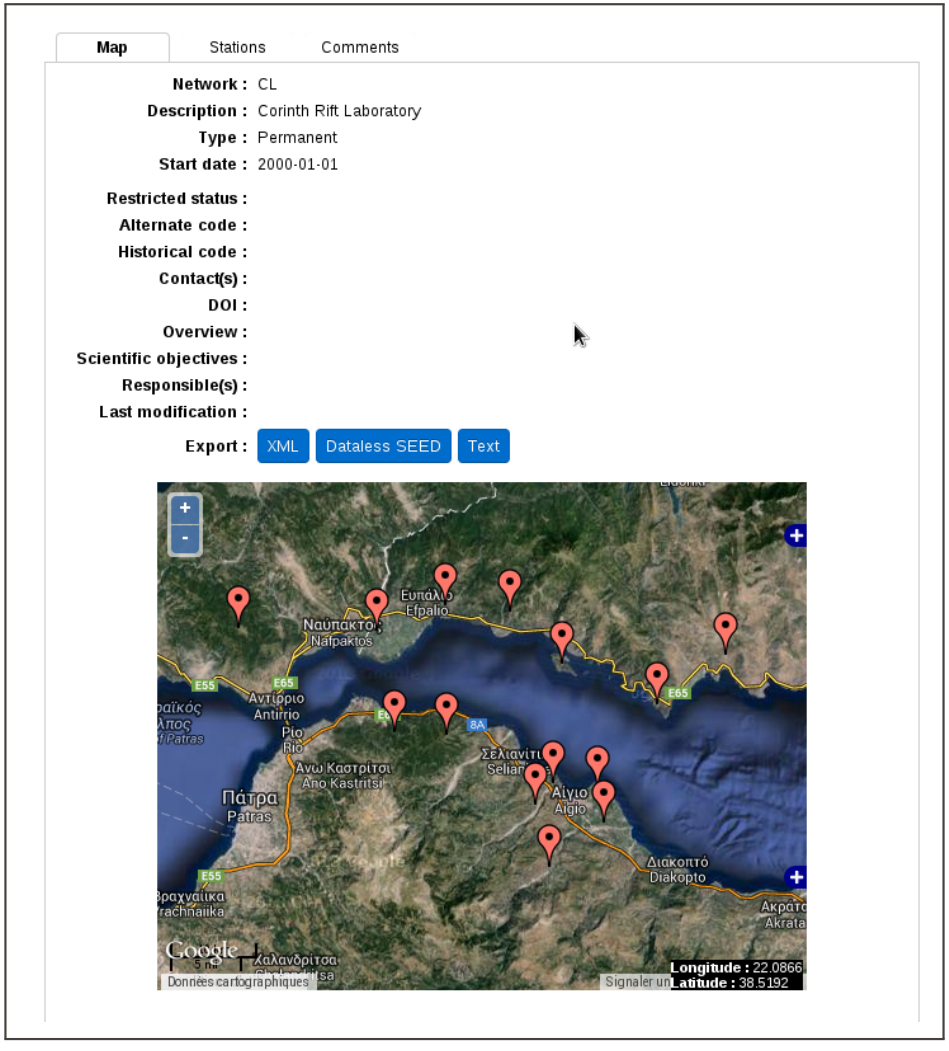
La documentation est accessible ici.

Mode autonome

Le convertisseur RBV vers ISO19139 peut être utilisé de manière autonome. Plus d'infos ici.

OMP / SEDOO

PORTAIL RESIF



LIENS

LIENS

- Références
  - Thèse de R. Fielding : http://opikanoba.org/tr/fielding/rest/
  - Site officiel Jersey : https://jersey.java.net/
- Tutoriels
  - http://www.mkyong.com/tutorials/jax-rs-tutorials/
  - http://draptik.github.io/blog/2013/07/19/unit-testing-restful-services/

QUESTIONS