Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'informatique
Année Universitaire : 2023/2024

2ème Année Master (SIA – GL) (S3) Module : **Ontologie et (Services Web)**

TP 5 : Développement d'un Service Web REST

L'objectif de ce TP est de créer des services web REST avec JAX-RS, les de ployer et écrire les clients qui appellent ces services.

1. Présentation

Jusqu'à présent, vous avez vu la création et le déploiement de services Web basés sur le protocole SOAP. Mais une autre approche des services Web et qui devient de plus en plus populaire existe : REST.

Le concept introduit en 2000 ne fait qu'utiliser les principes fondamentaux du Web. Utilisant le protocole HTTP, il permet l'envoi de messages sans enveloppe SOAP et dans un encodage libre (XML, JSON, binaire, simple texte). Il est actuellement très utilisé par les sites communautaires (ou réseaux sociaux) leur permettant de proposer à leurs clients une API facile à utiliser. Des sites comme Flickr, Facebook, Last.fm, Amazon proposent ainsi de telles API évitant à leurs clients de devoir passer par la case SOAP.

2. Architecture

L'information de base, dans une architecture REST, est appelée ressource.

On accède à une ressource (par son URI unique) pour procéder à diverses opérations (GET lecture / POST écriture / PUT modification / DELETE suppression), opérations supportées nativement par HTTP.

3. Création d'un premier service web REST

Les services REST sont spécifiés par le JCP (Java Community Process) sous le nom JAX-RS1(Java API for RESTful Services). Cette spécification précise ce que peut ou doit faire une implémentation, comme pour toutes ces spécifications. L'implémentation de référence que l'on utilise est Jersey2. Jersey est installé en standard dans un serveur JEE (tel que Glassfish ou JBoss), et peut s'installer dans un serveur Tomcat.

Vous allez créer un premier Service Web REST. Pour ce service Web, il s'agit simplement de montrer l'utilisation de celui-ci à partir d'un navigateur Web.

Vous allez suivre les e tapes suivantes :

- Créer un nouveau projet : Java Web→ Web Application que vous nommerez « *WebServiceRest* », vous choisirez *Glassfish* comme serveur.
- Créer un nouveau package que vous nommerez comme vous le souhaitez.
- Créer un service web REST en cliquant sur New→Other→WebService→RESTful WebServices from Pattern. Sélectionner "*Simple Root Resource*". Dans la fenêtre suivante, il faut rentrer les informations permettant de configurer le service Web REST, à savoir :
  - ✓ Un nom de package (Resource Package)
  - ✓ Un chemin d'accès qui permet d'accéder au service Web lorsqu'il est déployé (Path) et qui est à ajouter à l'URL donnée lors du déploiement de l'application Web.
  - ✓ Un nom de class (Class Name)

✓ Un type MIME pour le format de réponse du service Web. Modifier MIME Type en sélectionnant « *text/plain* ».

- Netbeans s'est chargé de la gestion des ressources en générant une sous classe de « *javax.ws.rs.core.Application* ». Il faut savoir que vous avez également la possibilité de réaliser vous-même la gestion des ressources en configurant Jersey.
- Une nouvelle classe est créée par défaut « *GenericResource* ». Vous allez supprimer le code par défaut et le remplacer par le code suivant :

```
import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.PathParam;
import javax.ws.rs.Consumes;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
@Path("Mypath")
public class GenericResource {
@GET
@Produces("text/plain")
public String getText() {
return "Mon premier service REST";
}
}
```

- Il s'agit d'un service REST simple qui invoque la méthode getText qui retourne une réponse sous format texte "*Mon premier service REST*".
- L'annotation @Path3 désigne d'une part l'URI d'accès au service et à ses sous-services, et d'autre part la présence de paramètres dans cette URI.
- Cinq annotations sont définies, qui peuvent annoter certaines méthodes d'un service REST : @GET, @POST, @PUT, @DELETE, et @HEAD. Elles correspondent aux cinq méthodes HTTP qui portent le même nom. On ne peut poser chaque annotation qu'une seule fois sur une unique méthode dans une classe donnée, pour un chemin d'accès donné. La liste des annotations est disponible à l'adresse suivante : http://docs.oracle.com/javaee/6/tutorial/doc/gilik.html.

4. **Déploiement et vérification de la disponibilité du service Web**

Après avoir créé le service Web, l'URL peut être testée sachant que la première partie de celle-ci permet d'accéder à la web application et que la deuxième permet d'accéder aux ressources de l'application. La troisième et dernière partie de l'URL donne le chemin d'accès à la classe où se trouve le web service. C'est dans cette classe que l'on surcharge les méthodes HTTP.

URL : http://localhost:8080/WebServiceRest/webresources/Mypath

- Faire un clean and build de votre projet et faire un deploy.
- Faire un clic droit sur le service « *GenericResource* » et sélectionner « *Test Resource Uri* ». Quel résultat obtenez-vous ? Pourquoi ?
- Vous allez maintenant faire clic droit sur le projet→Test Restful Web services, ensuite vous sélectionnez « *web test client in the project* » et sélectionnez votre projet. Vous allez ouvrir le fichier WADL et décrire son contenu.
- Vous allez modifier le format de réponse du service de telle sorte qu'il retourne le message sous format XML. Tester votre service.
- Vous allez ajouter une nouvelle méthode permettant de calculer le produit de deux entiers. Le chemin de la ressource doit être ../produit et le type de MIME sera soit du JSON soit du XML permettant d'afficher un message de type « le produit de deux entiers est ». Vos arguments seront annotés avec @QueryParam.

- Vous allez ajouter des valeurs par défaut grâce à @DefaultValue et testez.
- Utiliser maintenant l'annotation @PathParam.
- Tester l'exemple suivant4 et expliquer le résultat obtenu. A quoi servent @Context et la classe URIInfo.

```
@GET
  @Path("param")
public String getResource(@Context  UriInfo uriInfo) {
  return "The client used this URI to reach this resource method: " +
uriInfo.getAbsolutePath().toASCIIString();
                        }
```

Travail à faire :

A la lumière de l'activité précédente, créer un service web REST contenant deux opérations addition (a : entier, b : entier) → a+b : entier

Multiplication (a :réel, b : réel) → a*b : réel

Tester ce service