



Atelier SOA -TP09

Client JAVA pour un service web REST

Objectifs

Consommer un Service Web REST avec un client JAVA

1. Utiliser la classe **WebResource**

- Appeler des méthodes (GET, POST, PUT, DELETE)
- Configurer les paramètres de la requête

2. Utiliser l'API « **Gson** » de Google

- Convertir une représentation JSON en objet JAVA
- Convertir une représentation JSON en un tableau d'objets JAVA

A. Créer un projet JAVA avec Maven

1. Créer un projet **MAVEN** (**maven-archetype-quickstart**) dans **IntelliJ IDEA** nommé **jax-rs-client-java** (sous le dossier **workspace**) ayant les caractéristiques suivantes :

- groupId : **ws.rest**
- artifactId : **jax-rs-client-java**
- version : **1.0-SNAPSHOT**

2. Configurer ce projet pour supporter le développement des services web REST (dépendances de servlet et de JERSEY)

B. Invoquer les méthodes d'un service web REST avec **WebResource**

3. Créer, sous le dossier « **main** », une classe «**Jax_Rc_Client**» ayant le code suivant :

```

package ws.rest;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import javax.ws.rs.core.UriBuilder;
import java.net.URI;

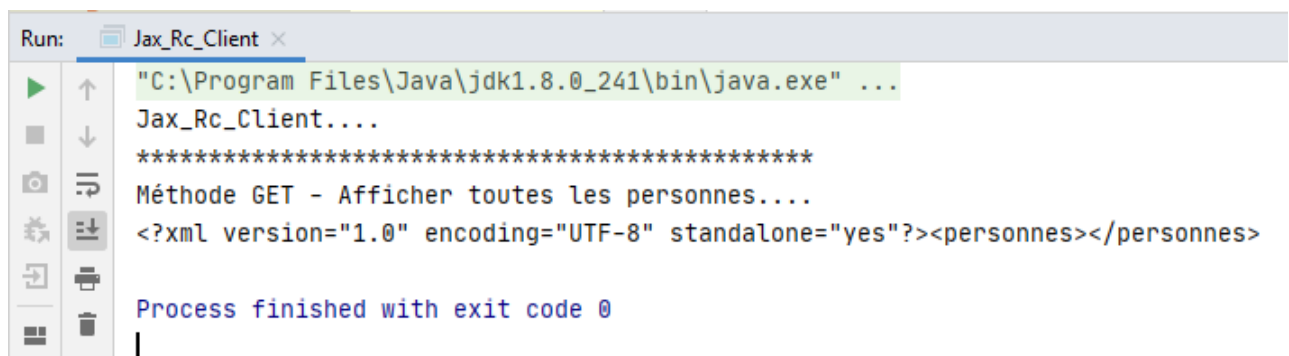
public class Jax_Rc_Client {

    public static void main( String[] args )
    {
        System.out.println("Jax_Rc_Client....");
        // Objet de configuration
        ClientConfig config = new DefaultClientConfig();
        //objet client
        Client client = Client.create(config);
        //créer l'uri
        URI uri =
UriBuilder.fromUri("http://localhost:9999/rest_arc/tp8/person").build();
        //obtenir une ressource correspondante à l'uri du service web
        WebResource service = client.resource(uri);
        //Requête GET
        System.out.println( "*****" );
        System.out.println( "Méthode GET - Afficher toutes les personnes...." );
        //référencer la méthode "getAll"

        WebResource resource1= service.path("getAll");
        //passer la méthode "get"
        String reponse1= resource1.get(String.class);
        //Afficher la réponse textuelle
        System.out.println(reponse1);
    }
}

```

4. L'exécution de cette classe nécessite la publication du service web REST « **person** » du projet « **jax-rs-arc** » avec le serveur « **Tomcat** » sur le port « **9999** ». Initialement, aucune personne n'est créée :



```

Run: Jax_Rc_Client x
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Jax_Rc_Client....
*****
Méthode GET - Afficher toutes les personnes....
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><personnes></personnes>

Process finished with exit code 0

```

5. Pour ajouter une personne, utiliser la méthode « **post** » de la classe « **WebResource** » comme suit :

```
//Requête POST
System.out.println( "*****" );
System.out.println( "Méthode POST - Ajouter une personne...." );
//référencer la méthode "add »
WebResource resource2= service.path("add") ;
//passer la méthode « post »
String reponse2= resource2.post(String.class, new
ws.rest.tp8.Personne(1, "Ben Mohamed",12)) ;
// Afficher la réponse textuelle
System.out.println(reponse2) ;
```

- Remarquer que la classe « **Personne** » n'est pas reconnue.
- Ceci nécessite la déclaration d'une dépendance du projet « **jax-rs-arc** » dans le fichier « **pom.xml** » du projet « **jax-rs-client-java** » :

```
<dependency>
  <groupId>org.soa.tp8</groupId>
  <artifactId>jax-rs-arc</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
```

- Or ce projet est de type « **webapp** », donc il est packagé dans un format « **war** » et il ne peut pas être référencé par maven (maven cherche des fichiers **.jar**)
- Pour remédier à ce problème, aller au projet « **jax-rs-arc** » pour ajouter le plugin « **maven-jar-plugin** » dans le fichier « **pom.xml** ». Ce plugin permet de générer un fichier JAR pour le projet « **jax-rs-arc** » avec le suffixe « **classes** ». Voici la déclaration de ce plugin dans le fichier « **pom.xml** » :

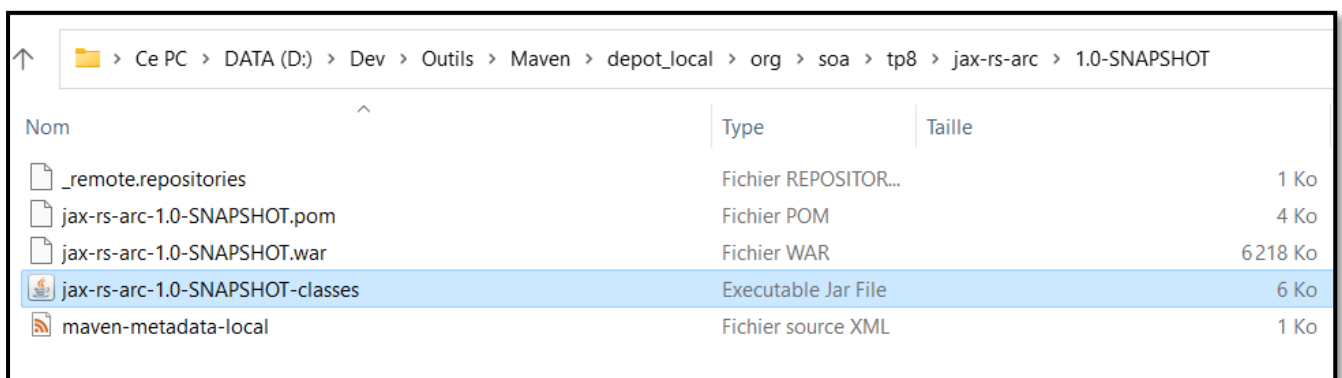
```

<plugin>

<groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.2.0</version>
  <configuration>
    <classifier>classes</classifier>
  </configuration>
  <executions>
    <execution>
      <id>package-jar</id>
      <phase>package</phase>
      <goals>
        <goal>jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>

```

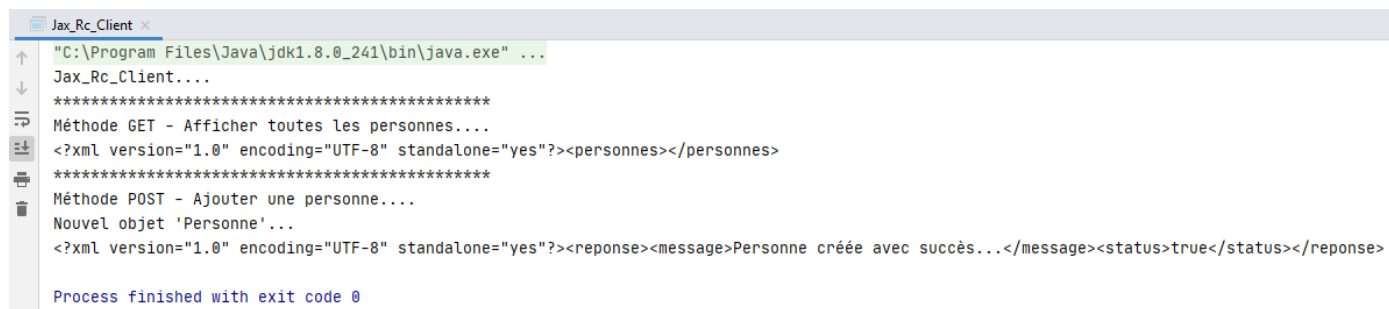
- Lancer la commande **install** de Maven pour publier le projet dans le dépôt local de Maven :



- Il est nécessaire, aussi, d'ajouter dans la déclaration de la dépendance du projet « **jax-rs-arc** » dans le fichier « **pom.xml** » du projet « **jax-rs-client-java** » la balise « **classifier** » comme suit :

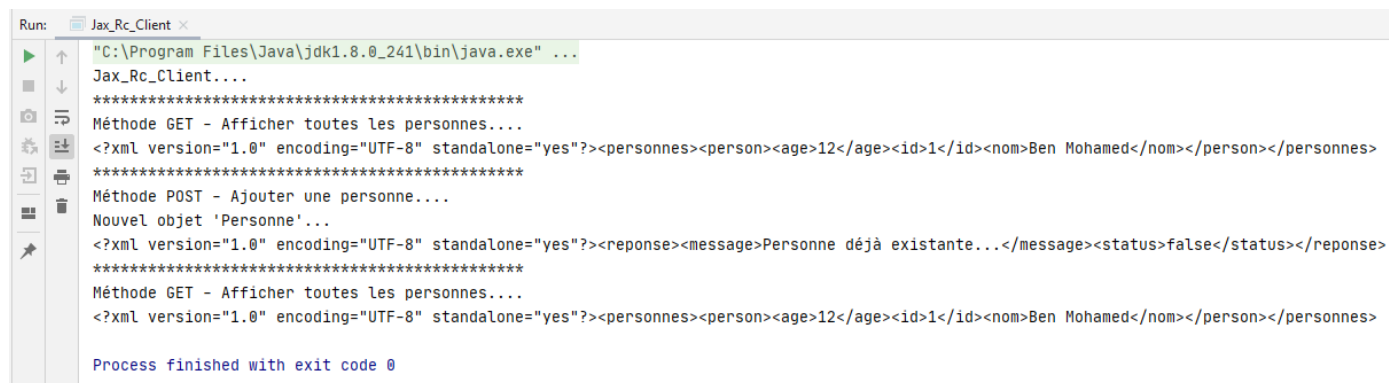
```
<dependency>
  <groupId>org.soa.tp8</groupId>
  <artifactId>jax-rs-arc</artifactId>
  <version>1.0-SNAPSHOT</version>
  <classifier>classes</classifier>
</dependency>
```

- Ainsi la classe « **Personne** » est reconnue. Passer maintenant à l'exécution de la classe « **Jax_Rc_Client** » :



```
Jax_Rc_Client ...
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Jax_Rc_Client...
*****
Méthode GET - Afficher toutes les personnes...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><personnes></personnes>
*****
Méthode POST - Ajouter une personne...
Nouvel objet 'Personne'...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><reponse><message>Personne créée avec succès...</message><status>true</status></reponse>
Process finished with exit code 0
```

6. Appeler une autre fois la méthode « **getAll** » pour afficher la personne nouvellement créée :



```
Run: Jax_Rc_Client ...
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Jax_Rc_Client...
*****
Méthode GET - Afficher toutes les personnes...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><personnes><person><age>12</age><id>1</id><nom>Ben Mohamed</nom></person></personnes>
*****
Méthode POST - Ajouter une personne...
Nouvel objet 'Personne'...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><reponse><message>Personne déjà existante...</message><status>false</status></reponse>
*****
Méthode GET - Afficher toutes les personnes...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><personnes><person><age>12</age><id>1</id><nom>Ben Mohamed</nom></person></personnes>
Process finished with exit code 0
```

7. Pour afficher la réponse au format JSON, appeler la méthode « **accept** » comme suit :

```
//passer la méthode "get" avec format « JSON »
reponse1=
resource1.accept(MediaType.APPLICATION_JSON).get(String.class);
// Afficher la réponse textuelle
System.out.println(reponse1);
```

8. Relancer l'exécution pour avoir une réponse au format « **JSON** » :

```
*****  
Méthode GET - Afficher toutes les personnes (format JSON)...  
{"person":{"age":"12","id":"1","nom":"Ben Mohamed"}}  
  
Process finished with exit code 0
```

9. Réaliser les appels suivants :

- Ajouter deux personnes :
 1. (5 , "Ben Omar", 50)
 2. (10, "Ayyadi", 23)
- Afficher la liste des personnes
- Modifier l'age de "Ben Omar" à 40
- Afficher toutes les personnes
- Supprimer "Ayyadi"
- Afficher toutes les personnes

C. Utiliser l'API « Gson » de Google pour convertir un objet Java dans sa représentation JSON et vice versa

10. Ajouter la dépendance suivante :

```
<!-- Google JSON API-->  
<dependency>  
  <groupId>com.google.code.gson</groupId>  
  <artifactId>gson</artifactId>  
  <version>2.2.4</version>  
</dependency>
```

11. Passer le code suivant pour récupérer la liste des personnes et les afficher :

```
//passer la méthode "get"
reponse1= resource1.accept(MediaType.APPLICATION_JSON).get(String.class);
// Afficher la réponse textuelle
System.out.println(reponse1);

// Récupérer des objets "Personne" en utilisant l'API gson de Google
Gson gson = new GsonBuilder().create();

JsonObject jo = new JsonParser().parse(reponse1).getAsJsonObject();
JsonArray jsonArray = jo.getAsJsonArray("person");

Personne[] listePers = gson.fromJson(jsonArray, Personne[].class);
System.out.println( "Liste des Personnes (API gson)....");
for (Personne p: listePers)
{
    System.out.println(p);
}
```

12. Ce code est convenable en cas de l'existence de plusieurs personnes (càd en cas d'envoi d'un tableau de personne (**array**)). En cas d'absence d'objet « **Personne** » ou l'existence d'un seul objet « **Personne** », le code génère une exception « **CastException** ». Pour remédier à ce problème, adopter cette nouvelle version du code :

```
// Récupérer des objets "Personne" en utilisant l'API gson de Google
Gson gson = new GsonBuilder().create();
if (!reponse1.equals("null")) // s'il existe au moins un objet "Personne"
{
    JsonObject jo = new JsonParser().parse(reponse1).getAsJsonObject();

    if (jo.get("person").isArray()) // en cas de plusieurs personnes
    {
        JsonArray jsonArray = jo.getAsJsonArray("person");

        Personne[] listePers = gson.fromJson(jsonArray, Personne[].class);
        System.out.println("Liste des Personnes (API gson)....");
        for (Personne p : listePers) {
            System.out.println(p);
        }
    }
    else
    { // en cas d'un seul objet "Personne"
        JsonObject jsonObject = jo.getAsJsonObject("person");
    }
}
```

```
        Personne personne = gson.fromJson(jsonObject, Personne.class);
        System.out.println("Une seule personne (API gson)....");
        System.out.println(personne);
    }
}
else
{
    System.out.println("Aucune personne n'est trouvée..");
}
```