STITUT SUPERIEUR DES ETUDES TECHNOLOGIQUES
DE SFAX

Département Technologies de Informatique

Atelier SOA -TP07
Développer et déployer un service web REST avec JERSEY et tomcat

Objectifs

Développer et déployer un Web Service REST développé avec l'API JAX-RS

1. **Configurer le fichier « pom.xml »**
   - Déclarer les dépendances nécessaires pour utiliser JERSEY
   - Configurer le plugin du moteur de servlet (Tomcat)
2. **Ecrire le service web REST**
   - Indiquer les annotations
3. **Déployer le service web**

A. Créer une application web

1. Créer un projet **MAVEN** (**maven-archetype-webapp**) dans **IntelliJ IDEA** nommé **jax-rs** (sous le dossier **workspace**) ayant les caractéristiques suivantes :
   - groupId : **org.soa.tp7**
   - artifactId : **jax-rs**
   - version : **1.0-SNAPSHOT**

B. Déclarer les dépendances utiles pour l'utilisation de JERSEY

2. Editer le fichier « **pom.xml** » du projet pour ajouter les dépendances suivantes :

```
<!--
https://mvnrepository.com/artifact/javax.servlet/
javax.servlet-api -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>provided</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/asm/asm -
->
<dependency>
  <groupId>asm</groupId>
  <artifactId>asm</artifactId>
  <version>3.3.1</version>
</dependency>

<!--
https://mvnrepository.com/artifact/com.sun.jersey
/jersey-bundle -->
<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-bundle</artifactId>
  <version>1.19.2</version>
</dependency>

<!--
https://mvnrepository.com/artifact/org.json/json
-->
<dependency>
  <groupId>org.json</groupId>
  <artifactId>json</artifactId>
  <version>20160810</version>
</dependency>

<!--
https://mvnrepository.com/artifact/com.sun.jersey
```

```
/jersey-server -->
<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-server</artifactId>
  <version>1.19.2</version>
</dependency>

<!--
https://mvnrepository.com/artifact/com.sun.jersey
/jersey-core -->
<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-core</artifactId>
  <version>1.19.2</version>
</dependency>
```

C. Ajouter le plugin tomcat7 à IntelliJ IDEA

3. Editer le fichier « **pom.xml** » du projet pour ajouter le plugin du « **tomcat7** ».

   Ajouter les lignes suivantes à l'intérieur de la balise « **build** » du fichier « **pom.xml** »:

```
<plugins>
  <plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
      <path>/webAppREST</path>
      <port>8888</port>
    </configuration>
  </plugin>

</plugins>
```

D. Configurer la servlet JERSEY dans le fichier « web.xml »

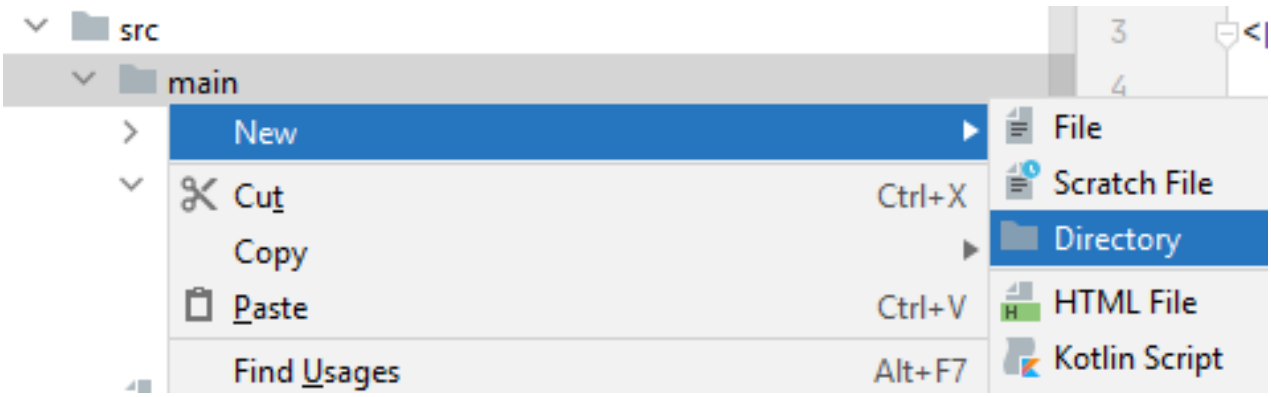
4. Editer le fichier « **web.xml** » (sous **./src/main/webapp/WEB-INF**) du projet pour configurer la servlet de l'API JERSEY.

   Ajouter les lignes suivantes à l'intérieur de la balise « **web-app** » du fichier « **web.xml** »:

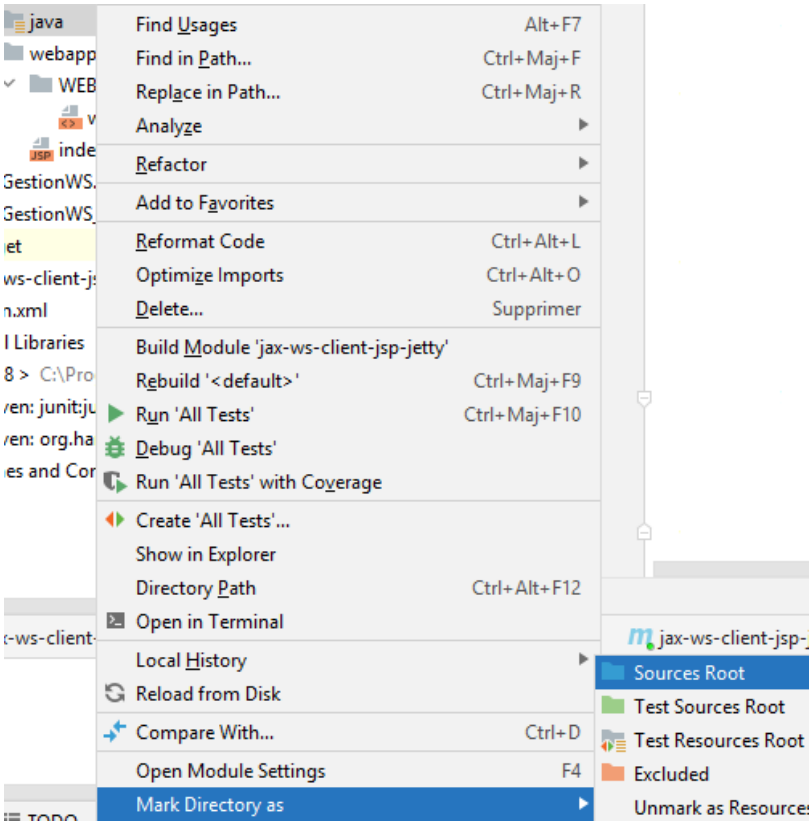
```
<display-name>Archetype Created Web Application</display-name>
 <servlet>
   <servlet-name>jerseyServlet</servlet-name>
   <servlet-
class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
   <init-param>
     <param-name>com.sun.jersey.config.property.packages</param-name>
     <param-value>services</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
 </servlet>
<servlet-mapping>
  <servlet-name>jerseyServlet</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

E. Définir un premier service web de test

5. Créer un dossier « **java** » sous le dossier « **main** » :



6. Rendre le dossier « **java** » nouvellement créé comme dossier de codes sources (**s'il n'est pas déjà un dossier de code sources**):



7. Sous le dossier « **java** », créer un package « **services** ».
8. Créer la classe « **PremierService** » :

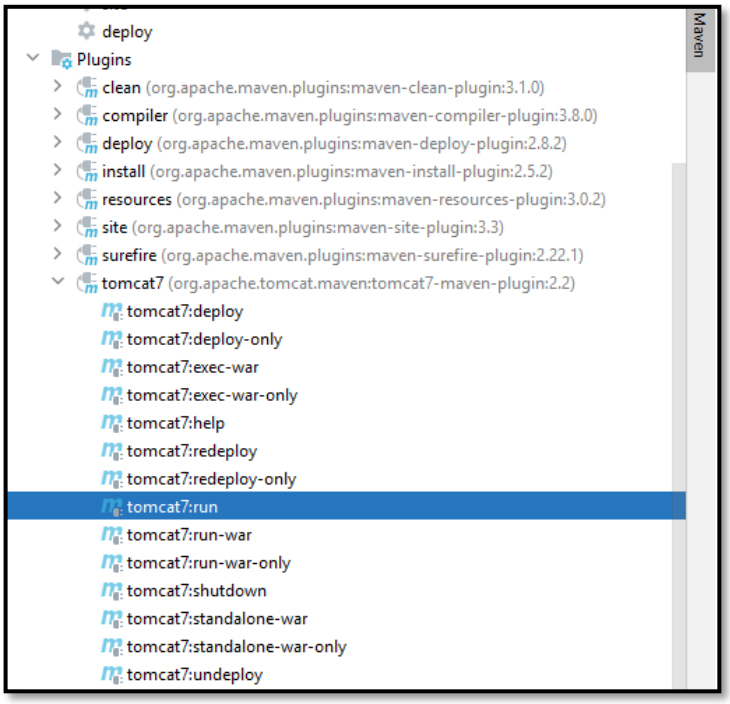
```
package services;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/premier")
public class PremierService {
    @Path("/test")
     @GET
     @Produces(MediaType.TEXT_PLAIN )
     public String message ()
     {
         return "Test du service avec succès...";
     }
}
```

F. Déployer le service web en utilisant le plugin tomcat

9. Dans le volet « **Maven** » double cliquer sue « **tomcat7:run** » :



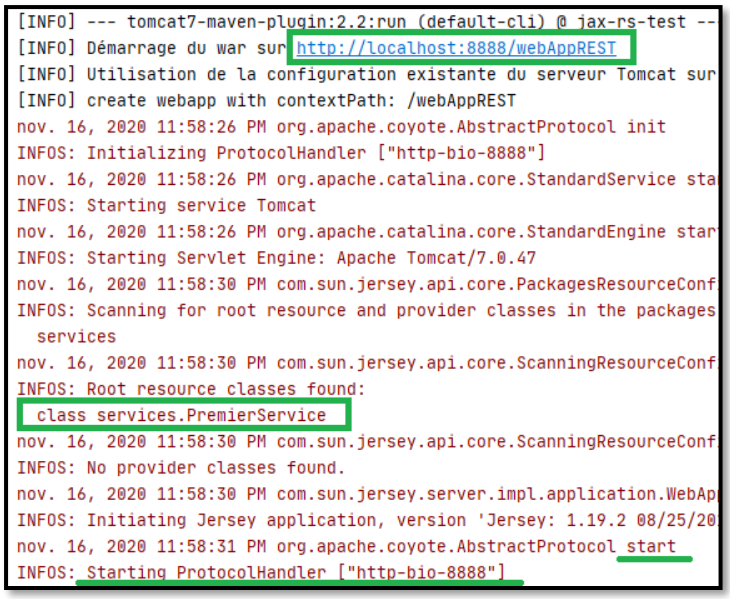
10. Remarquer le déclenchement de l'exception suivante :

```
SEVERE: StandardWrapper.Throwable
java.lang.TypeNotPresentException Create breakpoint : Type javax.xml.bind.JAXBContext not present <8 internal lines>
    at java.base/java.lang.Class.getGenericInterfaces(Class.java:1211)
```

Pour cela, ajouter les dépendances suivantes de JAXB :

```
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.1</version>
</dependency>
<dependency>
  <groupId>org.glassfish.jaxb</groupId>
  <artifactId>jaxb-runtime</artifactId>
  <version>2.3.1</version>
</dependency>
```

11. Si tout va bien, vous obtenez, au niveau de la console, le message suivant :



12. Le serveur Tomcat 7 est démarré, lancer l'url suivante pour le tester :

http://localhost:8888/webAppREST/rest/premier/test

13. Si tout va bien , vous recevez le résultat suivant :

14. Pour afficher les caractères spéciaux, modifier le code du service web pour spécifier le type de codage « **UTF-8** ». Pour ce faire, changer l'annotation « **@Produces** » comme suit :

```
@Produces(MediaType.TEXT_PLAIN + "; charset=UTF-8")
```

15. Redémarrer le serveur et renvoyer la requête

G. Développer un service web REST qui gère des objets

16. Copier les deux fichiers fournis avec l'énoncé de cet atelier («**Compte.java**» et « **GestionService.java** ») sous le package « **services** ».

17. Redémarrer le serveur **tomcat** pour déployer le service web « **GestionService** » et réaliser les opérations suivantes à travers le navigateur web et interpréter les réponses :

- Appeler la méthode « **getCompte** » du service web.
- Appeler la méthode « **getComptes** » du service web.
- Appeler la méthode « **getListeComptes** » du service web.
- Appeler la méthode « **ajouterCompte** » du service web.
- Appeler la méthode « **supprimerCompte** » du service web.

18. Ecrire une nouvelle méthode « **modifierCompte** » qui permet de changer la valeur du solde d'un compte donné.

19. Lancer l'exécution de cette méthode.