Cours Génie Logiciel
Philippe Dugerdil
Assistant: Sunny Avry (sunny.avry@unige.ch)

UNIVERSITÉ DE GENÈVE

TP09

WEB SERVICES AVEC REST (1)

A rendre pour la semaine suivant la séance d'exercices.

# 1 Installations

- Télécharger **Eclipse IDE for Enterprise Java Developers** : https://www.eclipse.org/downloads/packages/

Une fois Eclipse installé, aller dans Eclipse (ou Window) > Preferences, taper encoding et vérifier que l'encoding CSS, HTML et JSP est bien en UTF-8

- Télecharger **Tomcat 8.5** : https://tomcat.apache.org/download-80.cgi

  Aller dans Download > Tomcat 8

  **Windows :** dans core, télécharger un .zip et le décompresser sur votre disque dur, par exemple à la racine de votre disque dur

  **Mac/Linux :** dans core, télécharger un .tar.gz puis ouvrir une console et aller dans le répertoire de téléchargement

  Décompresser le tar.gz : tar -xzvf apache-tomcat-8.5.47.tar.gz

  Créer un nouveau dossier : sudo mkdir -p /usr/local

  Mettre tomcat dans ce dossier : sudo mv ~/Downloads/apache-tomcat-8.5.47 /usr/local

  Rendre les scripts BASH de ce dossier exécutables : sudo chmod +x /usr/local/bin/*.sh

# 2 Configuration

- Création d'un **nouveau projet** :

  Dans Eclipse : new > Dynamic web project
  Dans Project Name, indiquer JerseyDemo

  Cliquer sur New Runtime et choisir Apache Tomcat v8.5

  Cocher Create a new local server

  Cliquer sur Next et dans Browse, indiquer le dossier où est installé Tomcat

  Cliquer sur Finish. Tomcat doit maintenant se trouver dans Target runtime

Cliquer à nouveau sur Next puis encore Next puis cocher générer le fichier web.xml par défaut

Faire un clic droit sur JerseyDemo > Configure > Convert to Maven Project

Group Id: com.tp9.demo
Artifact Id: JerseyDemo
Version: 0.0.1-SNAPSHOT
Packaging : war

- Chargement des **dépendances** :

Dans pom.xml : copier-coller le code ci-après permettant de charger les dépendances nécessaires au projet (JAX-RS et Jersey). Les fichiers .jar téléchargés se retrouvent dans Java Resources > Librairies > Maven Dependencies

```
<properties>
    <jersey2.version>2.19</jersey2.version>
    <jaxrs.version>2.0.1</jaxrs.version>
  </properties>

  <dependencies>

    <!-- JAX-RS -->
    <dependency>
      <groupId>javax.ws.rs</groupId>
      <artifactId>javax.ws.rs-api</artifactId>
      <version>${jaxrs.version}</version>
    </dependency>

    <!-- Jersey 2.19 -->
    <dependency>
      <groupId>org.glassfish.jersey.containers</groupId>
      <artifactId>jersey-container-servlet</artifactId>
      <version>${jersey2.version}</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.core</groupId>
      <artifactId>jersey-server</artifactId>
      <version>${jersey2.version}</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.core</groupId>
      <artifactId>jersey-client</artifactId>
      <version>${jersey2.version}</version>
    </dependency>
  </dependencies>
```

Faire un clic droit sur JerseyDemo > Properties > Deployement Assembly > Add… > Java Build Path Entries > Next > Maven Dependencies > Finish > Apply and Close. Cette manipulation permet d'associer les dépendences téléchargées par Maven au répertoire WEB-INF/lib

# 3 Mise en place du service REST

- Configuration de la **servlet** :

  Dans le fichier web.xml, copier-coller le code ci-après permettant de spécifier la servlet utilisée (dans ce cas la servlet de Jersey) et de l'associer à toutes les urls (/*)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation=http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd version="3.1">

<display-name>Jersey Demo</display-name>
  <servlet>
    <servlet-name>JerseyDemo</servlet-name>

    <!-- Define ServletContainer of Jersey -->

    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

    <!-- Define the package to search for classes -->

    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>com.tp9.demo.services</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>

  </servlet>

  <!-- Map all the URLs to the Jersey ServletContainer -->

  <servlet-mapping>
    <servlet-name>JerseyDemo</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>

</web-app>
```

- Création du service **REST** :

  Dans Java Resources > src, créer le package **com.tp9.demo.services.** Dans ce package créer la class **HelloWorldService** et y copier-coller le code ci-après :

```
package com.tp9.demo.services;

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

@Path("/helloworld")
public class HelloWorldService {

   @GET
   @Path("/{name}")
   public Response helloworld(@PathParam("name") String msg) {
      String output = "Hello, " + msg + "!";
      return Response.status(200).entity(output).build();
   }

}
```

Sur le nom du projet, faire clic droit > Run As > Run on Server. Dans la fenêtre du navigateur interne à Eclipse ou dans un navigateur externe, entrer l'url suivante :

http://localhost:8080/ JerseyDemo/helloworld/<votreprenom>

Le texte, Hello, <votreprenom> doit s'afficher !

# 4 Travail Maison

Réaliser le déploiement manuel en exportant le .war de votre projet.

Le projet Eclipse complet (sans erreur de syntaxe) avec la classe implémentant le WebService décrit dans ce TP. Le fi chier zip (exportation d'archive depuis Eclipse) doit être déposé directement sur l'espace Moodle du cours