Web services REST avec JAX-RS

Hyacinthe MENIET

3 août 2019

Développer des web services RESTful en Java est possible depuis les débuts de REST, notamment grâce à l'API Servlet. En effet, cette API est suffisamment proche du protocole HTTP pour permettre à chaque développeur de transférer et de récupérer ce qu'il veut dans une réponse HTTP ou depuis une requête HTTP. Dans la documentation Web services REST avec Java (JAXB) j'explique comment faire. Reportez-vous au même article pour une courte introduction sur les architectures REST.
Ceci dit, malgré son expressivité, l'API Servlet reste loin du niveau de commodité atteint par JAX-WS pour les web services SOAP (voir l'article Web services SOAP avec Java (JAX-WS) pour plus de détails). C'est pourquoi, la Java Specification Request 311 a été déposée auprès du Java Community Process (JCP) et approuvée à l'unanimité. A la suite de quoi, un groupe d'experts a entamé des travaux afin de concevoir une API flexible, facile à utiliser et qui encouragerait les développeurs à monter des architectures REST. Le résultat de ce travail a été finalisé sous le nom d'API Java pour les services web RESTful (JAX-RS) dont la version courante est 2.1.
Dans cet article je vais indiquer comment développer un Web service REST en utilisant JAX-RS 2.1. Puis j'expliquerai comment le consommer. Pour le mapping Objet-Relationnel, j'utiliserai la Java Persistence API (JPA 2.1). Enfin, la sérialisation Java vers XML sera assurée par la Java API for XML Binding (JAXB 2.2).

1. Pré-requis

- Vous êtes familier de Java 8 et de sa syntaxe.
- Vous êtes familier des Web services REST.
- Vous disposez du JDK 8 minimum.
- Vous êtes familier de Tomcat et vous avez installé et configuré sa version 8 minimum.
- Vous êtes familier de MariaDB (**ou MySQL**) et vous avez installé et configuré sa version 10 minimum (ou MySQL 5.7 minimum).
- Vous êtes familier de Maven et vous avez installé et configuré sa version 3.3 minimum.

2. Vue d'ensemble

2.1 Vue d'ensemble de JAX-RS

JAX-RS est une API récente mais déjà bien fournie : Elle propose notamment des annotations spécifiques pour lier une URI et des verbes HTTP à des méthodes d'une classe Java. JAX-RS dispose également d'annotations capables d'injecter et de récupérer des données dans les entêtes d'une réponse ou depuis celles d'une requête. De même l'API JAX-RS fournit ce qu'il faut pour extraire et écrire ce que vous voulez dans le corps d'une requête/réponse HTTP. Enfin, en cas d'exception Java, JAX-RS est capable de produire une réponse avec le code HTTP le plus pertinent.

Comme souvent dans le monde Java, il existe de nombreuses implémentations de JAX-RS. La plupart de ces implémentations sont open source et libres. Dans ce document, je n’en présenterai qu’une : Jersey. Pour rappel, Jersey est un morceau de GlassFish. En particulier, c'est Jersey qui fournit l'implémentation de référence de JAX-RS.

2.2 Vue d'ensemble de JPA

La Java Persistence API (JPA) est une API légère à base de POJO pour la persistance d'objets Java. Cette API a été développée dans le cadre de la JSR 220. JPA est très souple car elle peut être utilisée dans une webapp, dans une application J2EE et même dans une simple application Java SE.
Comme JAX-RS, JPA dispose de nombreuses implémentations. Pour cet article, j'ai choisi l'une des implémentations les plus populaires : **Hibernate**.

2.3 Vue d'ensemble de JAXB

La Java API for XML Binding (JAXB) est un ensemble d'annotations qui aident à la sérialisation/désérialisation d'objets Java en XML. Cette API est extrêmement utile, car elle permet de se passer des représentations abstraites d'un document XML pour se concentrer sur de simples POJO annotés. Dans l'article Sérialisation et désérialisation XML avec Java (JAXB), j'explique plus en détail comment fonctionne JAXB. L'implémentation de référence de JAXB est intégrée au JDK.

2.4 Vue d'ensemble de l'article

Pour rentre cet article le plus didactique possible, je vais parcourir les capacités de JAX-RS à travers le système d'information d'une association de troc de services. Cette association met en relation des personnes qui proposent des services en échange d'autres services. Une fois mis en relation et sous réserve d'un accord mutuel, une personne peut, par exemple, recevoir une leçon de cuisine thaïlandaise en échange d'un dépannage de la plomberie.
Dans cet article, je ne modéliserai que la relation entre une personne et ses services. Concrètement, je réaliserai un Web service REST qui expose les 6 actions suivantes :

URI	HTTP	Description	Sortie	Entrée
/persons/{pId}	GET	Retourne des informations sur la personne	la personne	N/A
/persons/{pId}/services	GET	Retourne les services de la personne	Une collection de services	N/A
/persons/{pId}/services	POST	Ajoute un service à la personne	l'URI du service	Un service
/persons/{pId}/services/{sId}	GET	Retourne des informations sur le service	Un service	N/A
/persons/{pId}/services/{sId}	PUT	Modifie un service	N/A	Un service
/persons/{pId}/services/{sId}	DELETE	Supprime un service	N/A	N/A

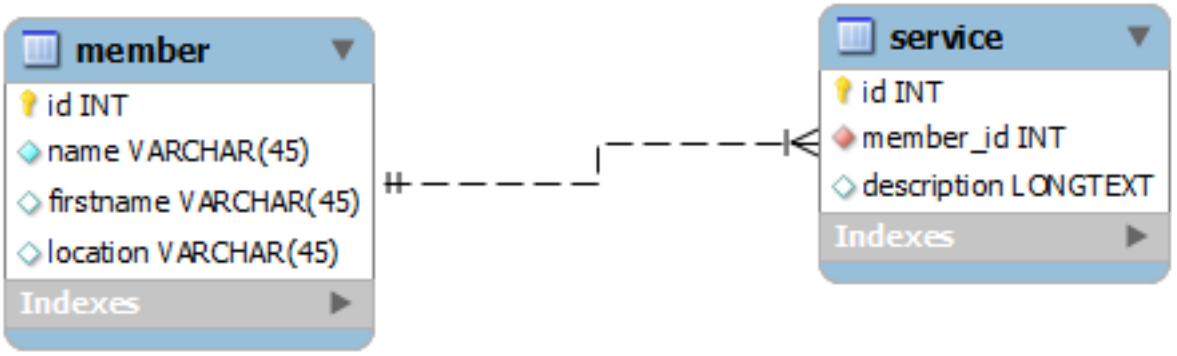
Vous pouvez télécharger le code source des projets Maven de web service REST (i.e. serveur et

client).

3. Le Web service

3.1 La base de données

La base de données bartering correspond au schéma suivant :



J'ai réalisé ce schéma avec Workbench. Créez la base de données correspondante ainsi que son propriétaire grâce au script Bartering-grant.sql :

```
CREATE DATABASE `bartering` CHARACTER SET utf8 COLLATE
    utf8_general_ci;
CREATE USER bartuser@localhost IDENTIFIED BY 'bartpwd';
GRANT ALL PRIVILEGES ON bartering.* TO bartuser@localhost;
FLUSH PRIVILEGES;
```

Créez les tables **person** et **service** à l'aide du script Bartering-schema.sql :

```
-- -----------------------------------------------------
-- Table `person`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `person` (
`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
`name` VARCHAR(45) NOT NULL ,
`firstname` VARCHAR(45) NULL ,
`location` VARCHAR(45) NULL ,
PRIMARY KEY (`id`) )
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `service`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `service` (
`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
`person_id` INT UNSIGNED NOT NULL ,
```

```
`description` LONGTEXT NULL ,
PRIMARY KEY (`id`) ,
INDEX `fk_service_person` (`person_id` ASC) ,
CONSTRAINT `fk_service_person`
FOREIGN KEY (`person_id` )
REFERENCES `person` (`id` )
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Insérez les personnes et les services en exécutant le script Bartering-data.sql :

```
INSERT INTO `person` VALUES (1,'Gates','Bill','Quartier Saint
    Bruno'),(2,'Jobs','Steve','Rue de la pomme');
INSERT INTO `service` VALUES (1,1,'Installation de Windows 7')
    ,(2,1,'Formation d\'1 heure en Economie');
```

3.2 Projet Maven

Créez un nouveau projet de type webapp avec Maven (commande à taper à la racine de votre workspace) :

```
$ mvn archetype:generate -DgroupId=net.dotmyself.bartering -
    DartifactId=bartering -Dversion=1.0 -Dpackage=net.dotmyself.
    bartering -DarchetypeArtifactId=maven-archetype-webapp -
    DinteractiveMode=false
```

Ecrasez le pom.xml généré, dans le dossier bartering, par ce pom.xml.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
    maven.apache.org/maven-v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>net.dotmyself.bartering</groupId>
<artifactId>bartering</artifactId>
<packaging>war</packaging>
<version>1.0</version>
<name>bartering Maven Webapp</name>
<url>http://maven.apache.org</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding
    >
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
</properties>
<dependencies>
<dependency>
```

```
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.26</version>
</dependency>
<dependency>
<groupId>org.glassfish.jersey.containers</groupId>
<artifactId>jersey-container-servlet</artifactId>
<version>2.26</version>
</dependency>
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-core</artifactId>
<version>5.2.12.Final</version>
</dependency>
<dependency>
<groupId>javax.ws.rs</groupId>
<artifactId>javax.ws.rs-api</artifactId>
<version>2.1</version>
</dependency>
<dependency>
<groupId>org.mariadb.jdbc</groupId>
<artifactId>mariadb-java-client</artifactId>
<version>2.2.1</version>
</dependency>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
<build>
<finalName>bartering</finalName>
</build>
</project>
```

Le pom.xml ci-dessous est paramétré pour récupérer les classes JDBC pour MariaDB. Si vous utilisez MySQL alors remplacez-y la dépendance à MariaDB par celle-ci :

```
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.45</version>
```

Depuis la racine de votre projet Maven, créez manuellement le dossier `src/main/java/net/dotmyself/fi`

3.3 Les flux XML

Les objets Java que vous sérialiserez en XML ressembleront aux flux XML suivant :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<person id="1">
<name>Gates</name>
<firstname>Bill</firstname>
<location>Quartier Saint Bruno</location>
<services>
<service id="1">
<description>Installation de Windows 7</description>
</service>
<service id="2">
<description>Formation d'1 heure en Economie</description>
</service>
</services>
</person>
```

Ce flux XML pouvant être validé par le schéma XSD suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema targetNamespace="http://file.dotmyself.net/source/13/
    model"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="
    qualified"
xmlns="http://file.dotmyself.net/source/13/model">
<xsd:element name="person">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="name" type="xsd:string" />
<xsd:element name="firstname" type="xsd:string" />
<xsd:element name="location" type="xsd:string" />
<xsd:element name="services">
<xsd:complexType>
<xsd:sequence>
<xsd:element maxOccurs="unbounded" ref="service" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:int" />
</xsd:complexType>
</xsd:element>
<xsd:element name="service">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="description" type="xsd:string" />
</xsd:sequence>
<xsd:attribute name="id" type="xsd:int" />
</xsd:complexType>
</xsd:element>
```

```
</xsd:schema>
```

3.4 La couche persistance

Les classes de la couche persistance appartiennent toutes au package **net.dotmyself.file.source._13.model**, et contiennent des annotations :

- **JAXB** (i.e. javax.xml.bind.annotation.*) : Pour la sérialisation java vers XML (et vice versa).
- **JPA** (i.e. javax.persistence.*) : Pour le mapping Objet-Relationnel

Créez la classe Service.java :

```
package net.dotmyself.file.source._13.model;
import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.Lob;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
/**
* The persistent class for the service database table.
*
* @author Hyacinthe MENIET
*/
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
"description"
})
@XmlRootElement(name = "service")
@Entity
@Table(name = "service")
@NamedQuery(name = Service.FIND_BY_PERSON,
query = "SELECT s FROM Service s WHERE s.id = :serviceId AND s.
    person.id = :personId")
public class Service implements Serializable {
```

```
private static final long serialVersionUID = 1L;
public static final String FIND_BY_PERSON = "Service.finByPerson
    ";
@XmlElement(required = true)
@Lob()
@Column(nullable = false)
protected String description;
@XmlAttribute(name = "id")
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
protected int id;
//bi-directional many-to-one association to Member
@XmlTransient
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "person_id")
protected Person person;
public Service() {
}
/**
* Gets the value of the description property.
*
* @return
*     possible object is
*     {@link String }
*
*/
public String getDescription() {
return description;
}
/**
* Sets the value of the description property.
*
* @param value
*     allowed object is
*     {@link String }
*
*/
public void setDescription(String value) {
this.description = value;
}
/**
* Gets the value of the id property.
*
* @return
*     possible value type is
*     {@link int }
*
*/
public int getId() {
```

```
return id;
}
/**
* Sets the value of the id property.
*
* @param value
*     allowed value type is
*     {@link int }
*
*/
public void setId(int value) {
this.id = value;
}
/**
* Gets the value of the Person property.
*
* @return
*     possible object is
*     {@link Person }
*
*/
public Person getPerson() {
return person;
}
/**
* Sets the value of the Person property.
*
* @param value
*     allowed object is
*     {@link Person }
*
*/
public void setPerson(Person value) {
this.person = value;
}
}
```

Créez la classe Person.java :

```
package net.dotmyself.file.source._13.model;
import java.io.Serializable;
import java.util.List;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```
import javax.persistence.JoinColumn;
import javax.persistence.NamedAttributeNode;
import javax.persistence.NamedEntityGraph;
import javax.persistence.OneToMany;
import javax.persistence.OrderBy;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;
/**
* The persistent class for the person database table.
*
* @author Hyacinthe MENIET
*/
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
"name",
"firstname",
"location",
"services"
})
@XmlRootElement(name = "person")
@Entity
@NamedEntityGraph(name = "graph.Person.services",
attributeNodes = @NamedAttributeNode("services"))
@Table(name = "person")
public class Person implements Serializable {
private static final long serialVersionUID = 1L;
public static final String PERSISTENCE_UNIT = "BarteringPU";
@XmlElement(required = true)
@Column(nullable = false)
protected String name;
@XmlElement(required = false)
@Column(nullable = true)
protected String firstname;
@XmlElement(required = false)
@Column(nullable = true)
protected String location;
// bi-directional many-to-one association to Service
@XmlElement(name = "service", required = true)
@XmlElementWrapper(name = "services", required = true)
@OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
@JoinColumn(name = "person_id", insertable = true, updatable =
    true)
@OrderBy("id ASC")
```

```
protected List<Service> services;
@XmlAttribute(name = "id")
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
protected int id;
public Person() {
}
/**
* Gets the value of the name property.
*
* @return
*     possible object is
*     {@link String }
*
*/
public String getName() {
return name;
}
/**
* Sets the value of the name property.
*
* @param value
*     allowed object is
*     {@link String }
*
*/
public void setName(String value) {
this.name = value;
}
/**
* Gets the value of the firstname property.
*
* @return
*     possible object is
*     {@link String }
*
*/
public String getFirstname() {
return firstname;
}
/**
* Sets the value of the firstname property.
*
* @param value
*     allowed object is
*     {@link String }
*
*/
public void setFirstname(String value) {
```

```
this.firstname = value;
}
/**
* Gets the value of the location property.
*
* @return
*     possible object is
*     {@link String }
*
*/
public String getLocation() {
return location;
}
/**
* Sets the value of the location property.
*
* @param value
*     allowed object is
*     {@link String }
*
*/
public void setLocation(String value) {
this.location = value;
}
/**
* Gets a reference to the list of services property.
*
* @return
*     possible objects of the following type(s)
*     {@link Service }
*
*/
public List<Service> getServices() {
return services;
}
/**
* Sets the value of the list of services property.
*
* @param value
*     Objects of the following type(s) are allowed in the list
*     {@link Service }
*
*/
public void setServices(List<Service> value) {
this.services = value;
}
/**
* Gets the value of the id property.
*
```

```
* @return
*      possible value type is
*      {@link int }
*
*/
public int getId() {
return id;
}
/**
* Sets the value of the id property.
*
* @param value
*      allowed value type is
*      {@link int }
*
*/
public void setId(int value) {
this.id = value;
}
}
```

3.5 La couche service

Les classes de la couche service appartiennent toutes au package **net.dotmyself.bartering.service**. Créez la classe ServiceResource.java :

```
package net.dotmyself.bartering.service;
import java.net.URI;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.persistence.EntityGraph;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.NotFoundException;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import net.dotmyself.file.source._13.model.Service;
```

```
import net.dotmyself.file.source._13.model.Person;
/**
*
* The Service resource that returns service's informations.
*
* @author Hyacinthe MENIET
*/
public class ServiceResource {
private final int personId;
public ServiceResource(int pid) {
this.personId = pid;
}
@GET
@Produces(MediaType.APPLICATION_XML)
public List<Service> getServices() {
EntityManagerFactory emf = Persistence
.createEntityManagerFactory(Person.PERSISTENCE_UNIT);
EntityManager em = emf.createEntityManager();
EntityGraph graph = em.getEntityGraph("graph.Person.services");
Map hints = new HashMap();
hints.put("javax.persistence.fetchgraph", graph);
Person p = em.find(Person.class, this.personId, hints);
if (null == p) {
emf.close();
throw new NotFoundException("person with personId=" + this.
    personId
+ " does not exist!");
}
emf.close();
return p.getServices();
}
@GET
@Path("{id : \\d+}")
@Produces(MediaType.APPLICATION_XML)
public Service getService(@PathParam("id") int id) {
Service s;
EntityManagerFactory emf;
emf = Persistence
.createEntityManagerFactory(Person.PERSISTENCE_UNIT);
s = (Service) emf.createEntityManager().createNamedQuery(Service.
    FIND_BY_PERSON).setParameter("serviceId", id)
.setParameter("personId", this.personId).getSingleResult();
emf.close();
return s;
}
@POST
@Consumes(MediaType.APPLICATION_XML)
public Response createService(Service s) {
EntityManagerFactory emf = Persistence
```

```
.createEntityManagerFactory(Person.PERSISTENCE_UNIT);
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
Person p = em.find(Person.class, this.personId);
s.setId(0);
s.setPerson(p);
em.persist(s);
tx.commit();
em.close();
emf.close();
return Response.created(URI.create("/bartering/api/persons/"+this
    .personId
+"/services/" + s.getId())).build();
}
@PUT
@Path("{id : \\d+}")
@Consumes(MediaType.APPLICATION_XML)
public void updateService(@PathParam("id") int id, Service
    extService) {
EntityManagerFactory emf = Persistence
.createEntityManagerFactory(Person.PERSISTENCE_UNIT);
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
Service intService = em.find(Service.class, id);
if (null == intService) {
throw new NotFoundException("service with personId="
+ this.personId + " and serviceId=" + id + " not found");
}
intService.setDescription(extService.getDescription());
em.merge(intService);
tx.commit();
em.close();
emf.close();
}
@DELETE
@Path("{id : \\d+}")
public void deleteService(@PathParam("id") int id) {
EntityManagerFactory emf = Persistence
.createEntityManagerFactory(Person.PERSISTENCE_UNIT);
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
Service s = em.find(Service.class, id);
if (null != s) {
em.remove(s);
}
tx.commit();
```

```
em.close();
emf.close();
}
}
```

Créez la classe PersonResource.java :

```
package net.dotmyself.bartering.service;
import java.util.HashMap;
import java.util.Map;
import javax.persistence.EntityGraph;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.ws.rs.GET;
import javax.ws.rs.NotFoundException;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import net.dotmyself.file.source._13.model.Person;
/**
*
* The Person resource that returns person's informations.
*
* @author Hyacinthe MENIET
*/
@Path("/persons")
public class PersonResource {
@GET
@Path("{id : \\d+}")
@Produces(MediaType.APPLICATION_XML)
public Person getPerson(@PathParam("id") int id) {
EntityManagerFactory emf = Persistence
.createEntityManagerFactory(Person.PERSISTENCE_UNIT);
EntityManager em = emf.createEntityManager();
EntityGraph graph = em.getEntityGraph("graph.Person.services");
Map hints = new HashMap();
hints.put("javax.persistence.fetchgraph", graph);
Person p = em.find(Person.class, id, hints);
if (null == p) {
emf.close();
throw new NotFoundException("person with personId=" + id
+ " does not exist!");
}
emf.close();
return p;
}
@Path("{id : \\d+}/services")
```

```
public ServiceResource getResources(@PathParam("id") int id) {
return new ServiceResource(id);
}
}
```

Terminez par la classe BarteringApplication.java qui indique à la Servlet de Jersey quelles classes resources charger :

```
package net.dotmyself.bartering.service;
import java.util.HashSet;
import java.util.Set;
import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;
/**
* Used to tell server container which Restful web services (@Path
    annotated
* classes) we want deployed.
*
* @author Hyacinthe MENIET
*/
@ApplicationPath("api")
public class BarteringApplication extends Application {
@Override
public Set<Class<?>> getClasses() {
Set<Class<?>> classes = new HashSet<>();
classes.add(net.dotmyself.bartering.service.PersonResource.class)
    ;
return classes;
}
}
```

3.6 Fichiers de configuration

Ecrasez le fichier src/main/webapp/WEB-INF/web.xml avec celui-ci :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">
<display-name>Bartering web application</display-name>
<servlet>
<servlet-name>net.dotmyself.bartering.service.
    BarteringApplication</servlet-name>
</servlet>
</web-app>
```

Créez le fichier src/main/resources/META-INF/persistence.xml suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/
    persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
<persistence-unit name="BarteringPU">
<provider>org.hibernate.jpa.HibernatePersistenceProvider</
    provider>
<properties>
<property name="javax.persistence.jdbc.driver" value="org.mariadb
    .jdbc.Driver" />
<property name="javax.persistence.jdbc.user" value="bartuser" />
<property name="javax.persistence.jdbc.password" value="bartpwd"
    />
<property name="javax.persistence.jdbc.url" value="jdbc:mariadb
    ://localhost:3306/bartering" />
<property name="hibernate.hbm2ddl.auto" value="update"/>
<property name="hibernate.dialect" value="org.hibernate.dialect.
    MariaDB53Dialect"/>
<property name="hibernate.use_sql_comments" value="true"/>
<property name="hibernate.show_sql" value="true"/>
</properties>
</persistence-unit>
</persistence>
```

Le fichier persistence.xml ci-dessus est paramétré pour se connecter à MariaDB. Si vous utilisez MySQL alors remplacez les 3 lignes spécifiques à MariaDB par leurs équivalentes MySQL :

```
<property name=&quot;javax.persistence.jdbc.driver&quot; value=&
    quot;org.hibernate.dialect.MySQL57InnoDBDialect&#039;&quot; />
<property name=&quot;javax.persistence.jdbc.url&quot; value=&quot
    ;jdbc:mysql://localhost:3306/bartering&quot; />
<property name=&quot;hibernate.dialect&quot; value=&quot;com.
    mysql.jdbc.Driver&quot;/>
```

4. Déploiement et test du Web service

4.1 Configuration et déploiement

Générez le war de votre projet maven. Quand vous avez terminé poussez simplement votre war dans **$CATALINA_HOME/webapps/** et redémarrez Tomcat. Vous pouvez consulter l'application web en vous connectant à l'adresse http://localhost:8080/bartering/. En particulier, vous pouvez consulter le flux XML de la personne 1 à l'adresse http://localhost:8080/bartering/api/persons/1

5. Le client

5.1 Projet Maven

Créez un nouveau projet Maven (commande à taper à la racine de votre workspace) :

```
$ mvn archetype:generate -DgroupId=net.dotmyself.bartering -
    DartifactId=bartering-client -Dversion=1.0 -Dpackage=net.
    dotmyself.bartering -DarchetypeArtifactId=maven-archetype-
    quickstart -DinteractiveMode=false
```

Ecrasez le pom.xml généré, dans le dossier bartering-client, par ce pom.xml.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
    maven.apache.org/maven-v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>net.dotmyself.bartering</groupId>
<artifactId>bartering-client</artifactId>
<packaging>jar</packaging>
<version>1.0</version>
<name>bartering-client</name>
<url>http://maven.apache.org</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding
    >
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
</properties>
<dependencies>
<dependency>
<groupId>org.glassfish.jersey.core</groupId>
<artifactId>jersey-client</artifactId>
<version>2.26</version>
</dependency>
<dependency>
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.26</version>
</dependency>
<dependency>
<groupId>org.glassfish.jersey.media</groupId>
<artifactId>jersey-media-jaxb</artifactId>
<version>2.26</version>
</dependency>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
```

```
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

5.2 Programme client

Pour désérialiser votre flux XML en java, vous aurez besoin de classes munies d'annotations **JAXB**. Créez la classe Service.java :

```
package net.dotmyself.file.source._13.model;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;
/**
* The jaxb class for a service.
*
* @author Hyacinthe MENIET
*/
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
"description"
})
@XmlRootElement(name = "service")
public class Service {
@XmlElement(required = true)
protected String description;
@XmlAttribute(name = "id")
protected Integer id;
public Service() {
}
public Service(String description) {
this.description = description;
}
/**
* Gets the value of the description property.
*
* @return
*     possible object is
*     {@link String }
*
*/
public String getDescription() {
return description;
```

```
}
/**
* Sets the value of the description property.
*
* @param value
*     allowed object is
*     {@link String }
*
*/
public void setDescription(String value) {
this.description = value;
}
/**
* Gets the value of the id property.
*
* @return
*     possible object is
*     {@link Integer }
*
*/
public Integer getId() {
return id;
}
/**
* Sets the value of the id property.
*
* @param value
*     allowed object is
*     {@link Integer }
*
*/
public void setId(Integer value) {
this.id = value;
}
@Override
public String toString() {
return "Service{" + "description=" + description + ", id=" + id +
    '}';
}
}
```

Créez la classe Person.java :

```
package net.dotmyself.file.source._13.model;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
```

```
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;
/**
* The jaxb class for a person.
*
* @author Hyacinthe MENIET
*/
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
"name",
"firstname",
"location",
"services"
})
@XmlRootElement(name = "person")
public class Person {
@XmlElement(required = true)
protected String name;
@XmlElement(required = false)
protected String firstname;
@XmlElement(required = false)
protected String location;
@XmlElement(name = "service", required = true)
@XmlElementWrapper(name = "services", required = true)
protected List<Service> services;
@XmlAttribute(name = "id")
protected int id;
public Person() {
}
/**
* Gets the value of the name property.
*
* @return possible object is {@link String }
*
*/
public String getName() {
return name;
}
/**
* Sets the value of the name property.
*
* @param value allowed object is {@link String }
*
*/
public void setName(String value) {
this.name = value;
```

```
}
/**
* Gets the value of the firstname property.
*
* @return possible object is {@link String }
*
*/
public String getFirstname() {
return firstname;
}
/**
* Sets the value of the firstname property.
*
* @param value allowed object is {@link String }
*
*/
public void setFirstname(String value) {
this.firstname = value;
}
/**
* Gets the value of the location property.
*
* @return possible object is {@link String }
*
*/
public String getLocation() {
return location;
}
/**
* Sets the value of the location property.
*
* @param value allowed object is {@link String }
*
*/
public void setLocation(String value) {
this.location = value;
}
/**
* Gets a reference to the list of services property.
*
* @return possible objects of the following type(s) {@link
   Service }
*
*/
public List<Service> getServices() {
return services;
}
/**
* Sets the value of the list of services property.
```

```
*
* @param value Objects of the following type(s) are allowed in
    the list
*     {@link Service }
*
*/
public void setServices(List<Service> value) {
this.services = value;
}
/**
* Gets the value of the id property.
*
* @return possible value type is {@link int }
*
*/
public int getId() {
return id;
}
/**
* Sets the value of the id property.
*
* @param value allowed value type is {@link int }
*
*/
public void setId(int value) {
this.id = value;
}
@Override
public String toString() {
return "Person{" + "name=" + name + ", firstname=" + firstname +
    ", location=" + location +
", services=" + services + ", id=" + id + '}';
}
}
```

Ecrasez la classe net.dotmyself.bartering.App par celle-ci :

```
package net.dotmyself.bartering;
import java.net.URI;
import java.util.List;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import net.dotmyself.file.source._13.model.Person;
import net.dotmyself.file.source._13.model.Service;
```

```
/**
* Java client for the Bartering's restful web service
*
* @author hyacinthe MENIET
*/
public class App {
private static final String REST_URL = "http://localhost:8080/
    bartering/api/persons";
private static final String BGID = "1";
private static final String SJID = "2";
private static final String SERVICE_URI = "services";
public static void main(String[] args) {
Client client = ClientBuilder.newClient();
WebTarget target = client.target(REST_URL);
Service createdService;
URI CreatedServiceUri;
WebTarget createdServiceTarget;
Person SteveJobs;
System.out.println("Reading the person Bill Gates ...");
try (Response response = target.path(BGID).request(MediaType.
    APPLICATION_XML).get()) {
Person BillGates = response.readEntity(Person.class);
System.out.println(response.getStatus());
System.out.println(BillGates);
System.out.println();
}
System.out.println("Reading the person Steve Jobs ...");
try (Response response = target.path(SJID).request(MediaType.
    APPLICATION_XML).get()) {
SteveJobs = response.readEntity(Person.class);
System.out.println(response.getStatus());
System.out.println(SteveJobs);
System.out.println();
}
System.out.println("Adding a Service to the person Steve Jobs
    ...");
createdService = new Service("Telechargement de 3 morceaux sur l'
    Apple Music Store");
try (Response response = target.path(SJID).path(SERVICE_URI).
    request(MediaType.APPLICATION_XML)
.post(Entity.entity(createdService, MediaType.APPLICATION_XML)))
    {
CreatedServiceUri = response.getLocation();
System.out.println(response.getStatus());
System.out.println();
}
System.out.println("Reading the list of services from the person
    Steve Jobs ...");
try (Response response = target.path(SJID).path(SERVICE_URI).
```

```
    request(MediaType.APPLICATION_XML).get()) {
List<Service> services = response.readEntity(new GenericType<List
    <Service>>() {
});
System.out.println(response.getStatus());
System.out.println(services);
System.out.println();
}
System.out.println("Reading one service from the person Steve
    Jobs ...");
createdServiceTarget = client.target(CreatedServiceUri);
try (Response response = createdServiceTarget.request(MediaType.
    APPLICATION_XML).get()) {
createdService = response.readEntity(Service.class);
System.out.println(response.getStatus());
System.out.println(createdService);
System.out.println();
}
System.out.println("Updating one service from the person Steve
    Jobs ...");
createdService = new Service("1 Entree pour le concert : Flash
    Haters");
try (Response response = createdServiceTarget.request(MediaType.
    APPLICATION_XML)
.put(Entity.entity(createdService, MediaType.APPLICATION_XML))) {
System.out.println(response.getStatus());
System.out.println();
}
System.out.println("Re-reading one service from the person Steve
    Jobs ...");
try (Response response = createdServiceTarget.request(MediaType.
    APPLICATION_XML).get()) {
createdService = response.readEntity(Service.class);
System.out.println(response.getStatus());
System.out.println(createdService);
System.out.println();
}
System.out.println("Deleting one service from the person Steve
    Jobs ...");
try (Response response = createdServiceTarget.request(MediaType.
    APPLICATION_XML).delete()) {
System.out.println(response.getStatus());
System.out.println();
}
System.out.println("Re-reading the person Steve Jobs ...");
try (Response response = target.path(SJID).request(MediaType.
    APPLICATION_XML).get()) {
SteveJobs = response.readEntity(Person.class);
System.out.println(response.getStatus());
```

```
System.out.println(SteveJobs);
System.out.println();
}
}
}
```

Compilez et exécutez cette classe via la commande ci-dessous (à taper à la racine de votre projet Maven client) :

```
$ mvn -q clean compile exec:java -Dexec.mainClass="net.dotmyself.
    bartering.App"
```