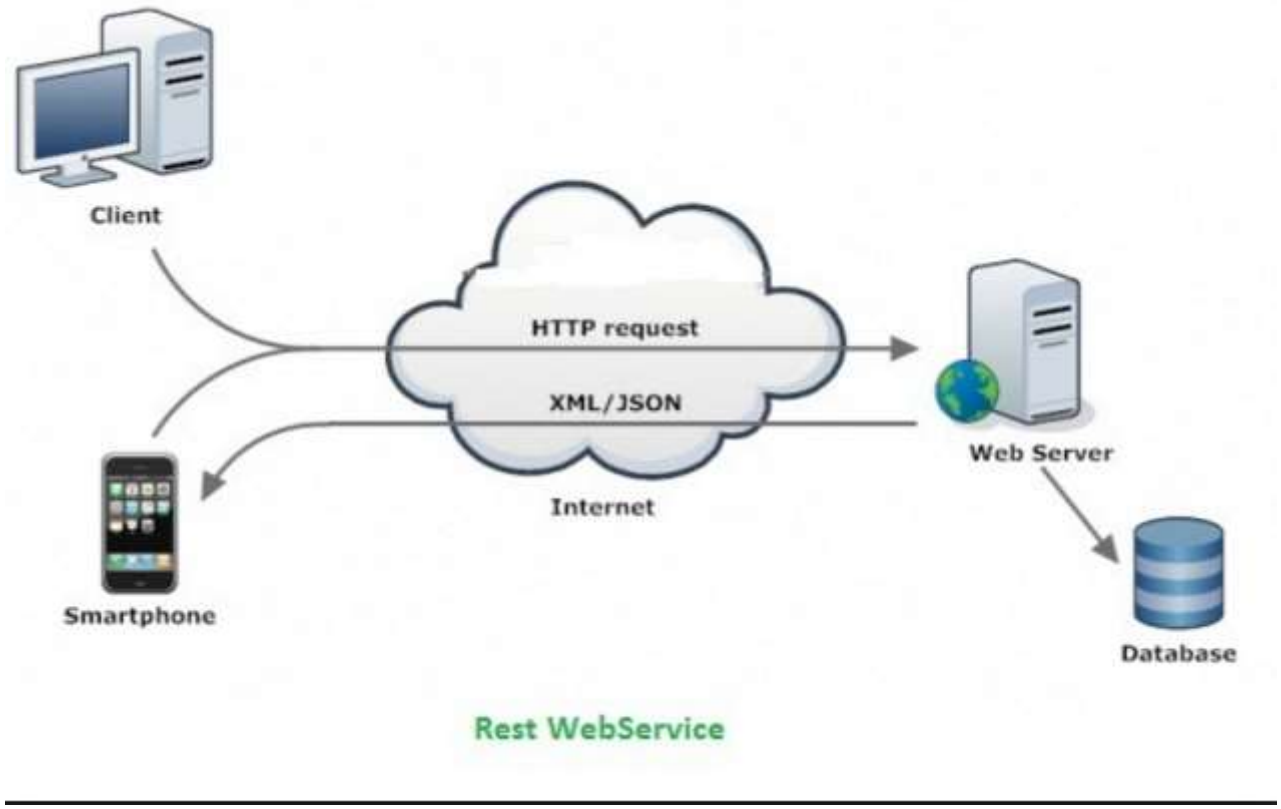
REST et les Services Web Restful

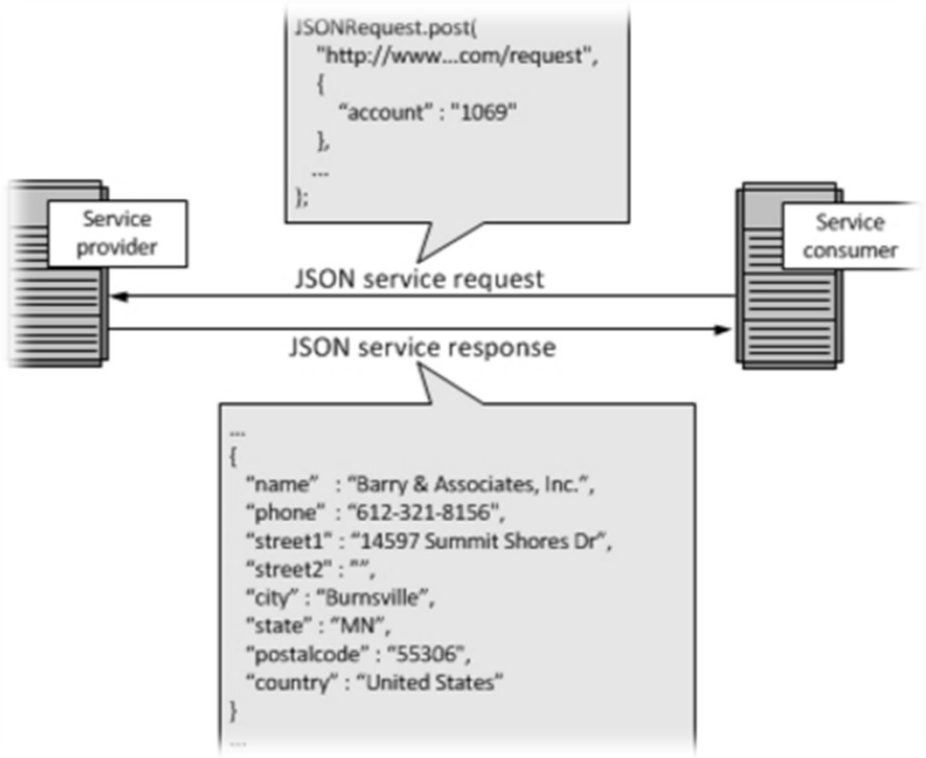
C'est quoi Rest?

- REST c'est un style d'architecture (introduit par Roy Fielding en 2000) ;
- REST utilise HTTP dans le protocole de transport ;
- REST se base sur des ressources où chaque composant est une ressource et une ressource est accessible par une interface commune en utilisant des méthodes HTTP standards (GET, POST, PUT et DELETE) ;
- Le serveur REST fournit l'accès aux ressources et le client REST accède et présente les ressources ;
- Une ressource est identifiée par une URI (Uniform Resource Identifier) ;
- REST utilise diverses représentations pour représenter une ressource comme du texte, JSON et XML (**JSON est le format le plus populaire utilisé dans les services Web**).

L'architecture REST



L'architecture REST



Méthodes HTTP

- **GET :** Fournit un accès en lecture seule à une ressource.
- **PUT :** Utilisé pour créer une nouvelle ressource.
- **DELETE :** Utilisé pour supprimer une ressource.
- **POST :** Utilisé pour mettre à jour une ressource existante ou créer une nouvelle ressource.
- **OPTIONS :** Utilisé pour obtenir les opérations supportées par une ressource.

Les services web Restful

- Les services Web basés sur l'architecture REST sont connus comme les services Web RESTful.

- Ces services Web utilisent des méthodes HTTP pour mettre en œuvre le concept de l'architecture REST.

- L'API JAX-RS 2.0 permet la création des services web Restful.

JAX-RS 2.0: Les implémentations

- ✓ Jersey (L'implémentation de référence de SUN) ;
  - https://jersey.java.net ;
  - Dernière version : 2.23.1 (Juin 2016).

- ✓ CXF (Framework développé par Apache) :
  - http://cxf.apache.org ;
  - Dernière version : 3.1.6 (Mars 2016).

- ✓ RESTEasy (Framework de Jboss) :
  - http://resteasy.jboss.org ;
  - Dernière version : 3.0.17 (Juin 2016).

- ✓ Restlet :
  - https://restlet.com/ ;
  - Dernière version : 2.3.7 (Mars 2016).

Jersey: Configuration de la Servlet

```
<servlet>
      <servlet-name>Jersey RESTful Application</servlet-name>
      <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
      <init-param>
          <param-name>jersey.config.server.provider.packages</param-name>
          <param-value>rest.service</param-value>
      </init-param>
  </servlet>
  <servlet-mapping>
      <servlet-name>Jersey RESTful Application</servlet-name>
      <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
```

Jersey: Méthode GET (1/5)

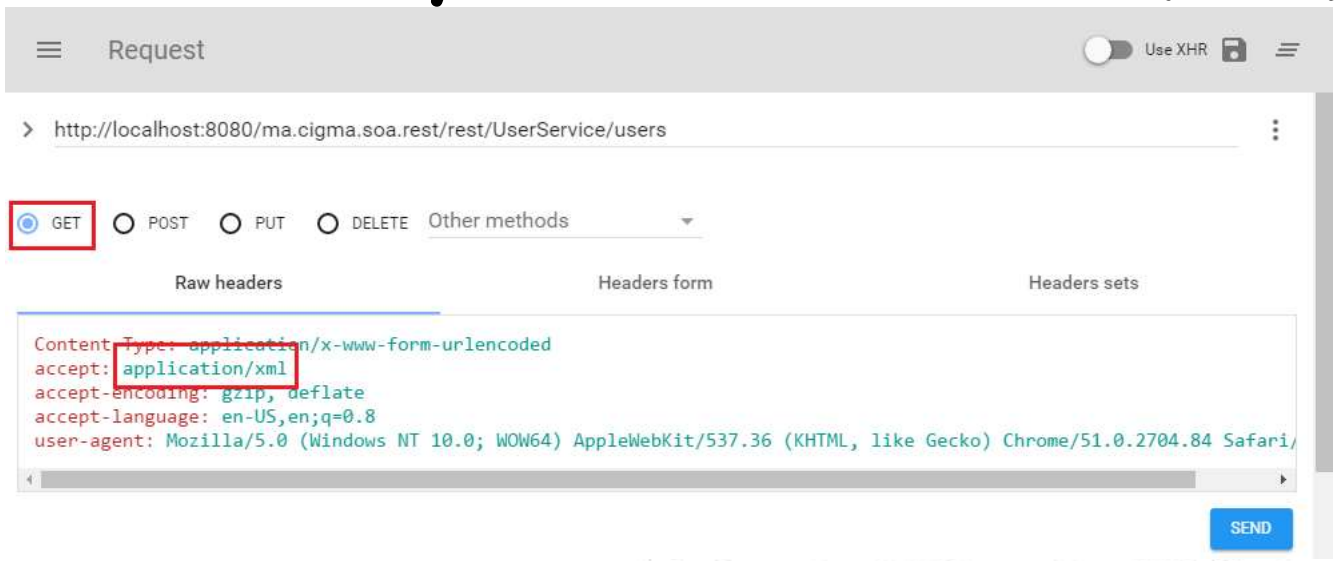
```java
@Path("/UserService")
public class UserService {

    IUserDao userDao = new UserDaoImpl();
    private static final String SUCCESS_RESULT = "<result>success</result>";
    private static final String FAILURE_RESULT = "<result>failure</result>";

    @GET
    @Path("/users")
    @Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
    public List<User> getUsers() {
        return userDao.getAllUsers();
    }
}
```

- L'exécution du lien :

  http://localhost:8080/ma.cigma.soa.rest/rest/UserService/users

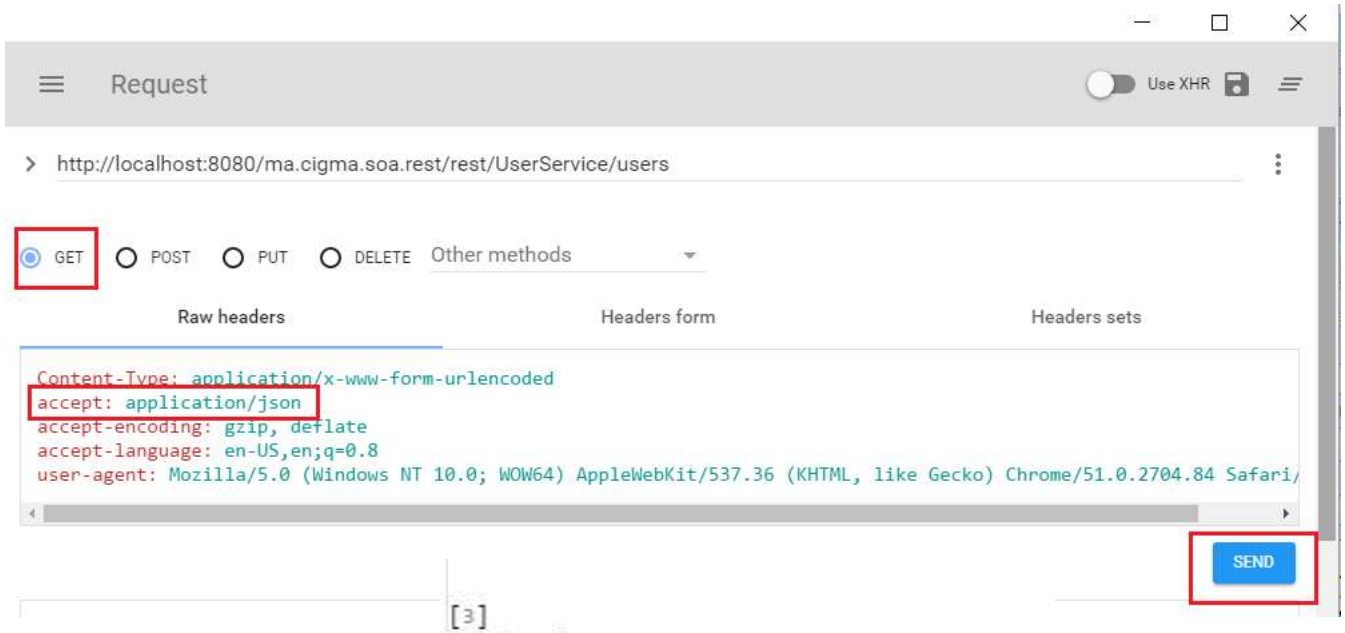
donnera comme résultat la liste des utilisateurs en format XML ou bien en format Json selon la requête.

Jersey: Méthode GET (2/5)




```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<users>
    <user>
        <id>1</id>
        <name>Jhon</name>
        <profession>Ingénieur</profession>
    </user>
    <user>
        <id>2</id>
        <name>Richard</name>
        <profession>Docteur</profession>
    </user>
    <user>
        <id>3</id>
        <name>Smith</name>
        <profession>Professeur</profession>
    </user>
</users>
```

Jersey: Méthode GET (3/5)




```
[3]
 -0: {
     "id": 1
     "name": "Jhon"
     "profession": "Ingénieur"
 }
 -1: {
     "id": 2
     "name": "Richard"
     "profession": "Docteur"
 }
 -2: {
     "id": 3
     "name": "Smith"
     "profession": "Professeur"
 }
```

Jersey: Méthode GET (4/5)

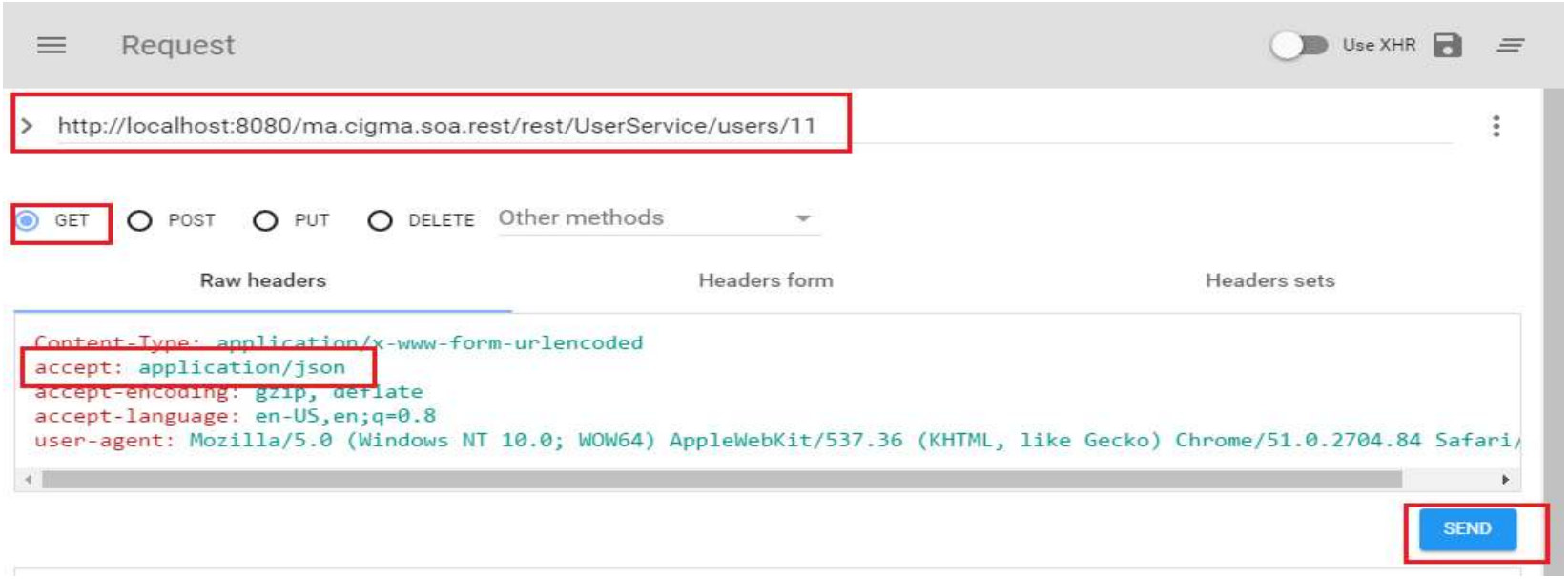
```
@GET
@Path("/users/{userid}")
@Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
public User getUser(@PathParam("userid") int userid) {
    return userDao.getUser(userid);
}
```

- L'exécution du lien :

http://localhost:8080/ma.cigma.soa.rest/rest/UserService/users/1

donnera comme résultat l'utilisateur ayant l'id 1 en format Json ou bien en format XML selon la demande du client.

Jersey: Méthode GET (5/5)

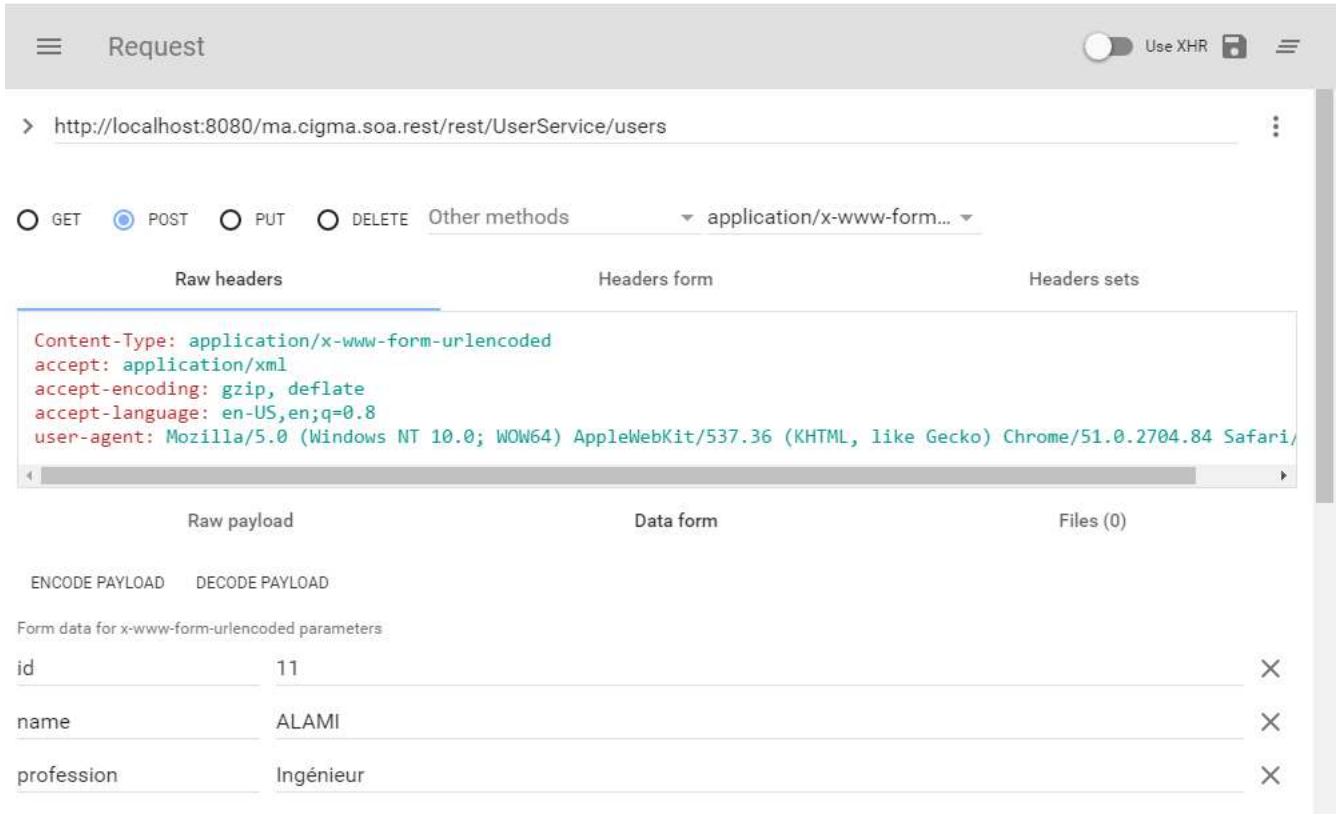



```
{
    "id": 11
    "name": "ALAMI"
    "profession": "Ingénieur"
}
```

Jersey : Méthode POST (1/2)

```
@POST
@Path("/users")
@Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON,MediaType.TEXT_PLAIN })
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
public String updateUser(@FormParam("id") int id, @FormParam("name") String name,
        @FormParam("profession") String profession, @Context HttpServletResponse servletResponse)
                throws IOException {
    User user = new User(id, name, profession);
    int result = userDao.updateUser(user);
    if (result == 1) {
        return SUCCESS_RESULT;
    }
    return FAILURE_RESULT;
}
```

Jersey : Méthode POST (2/2)




```
<result>success</result>
```

Jersey : L'api Client (Méthode GET)

```java
package test;

import java.util.List;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.MediaType;
import rest.modele.User;
public class Test2 {
    private static String REST_SERVICE_URL = "http://localhost:8080/ma.cigma.soa.rest/rest/UserService/users";
    private static final String PASS = "pass";
    private static final String FAIL = "fail";
    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();
        GenericType<List<User>> list = new GenericType<List<User>>() {};
        List<User> users = client.target(REST_SERVICE_URL).request(MediaType.APPLICATION_XML).get(list);
        String result = PASS;
        if (users.isEmpty()) {
            result = FAIL;
        }
        System.out.println("Test case name: testGetAllUsers, Result: " + result);
    }
}
```

Jersey : L'api Client (Méthode DELETE)

```java
package test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.core.MediaType;
public class Test2 {
    private static String REST_SERVICE_URL = "http://localhost:8080/ma.cigma.soa.rest/rest/UserService/users";
    private static final String PASS = "pass";
    private static final String FAIL = "fail";
    private static final String SUCCESS_RESULT = "<result>success</result>";
    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();
        String callResult = client.target(REST_SERVICE_URL).path("/{userid}").resolveTemplate("userid", 11)
                .request(MediaType.APPLICATION_XML).delete(String.class);
        String result = PASS;
        if (!SUCCESS_RESULT.equals(callResult)) {
            result = FAIL;
        }
        System.out.println("Test case name: testDeleteUser, Result: " + result);
    }
}
```