RESTful **Web Services**

M.I. Capel

ETS Ingenierías Informática y
Telecomunicación
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada
Email: manuelcapel@ugr.es
http://lsi.ugr.es/mcapel/

October, 17th 2023
Máster Universitario en Ingeniería Informática

Representational State Transfer (REST)

Historic outline

Initially proposed by Roy Thomas Fielding in his PhD dissertation book: *Architectural Styles and the Design of Network-based Software Architectures*(2000)

Fundamental characteristics

- REST-based notation to be used is mainly based on the 1996 `Http 1.0` standard
- Client applications communicate with serversa by using *Http verbs*: `GET, POST, DELETE, PUT, PATCH`
- The server can access *resources* that are identified by `URI` (*Uniform Resource Identifier*)
- Resources can have several textual representations: `XML, JSON, HTML, ...`

Http methods

Recommended return values for primary HTTP methods which are combined with URI resources

No	**Verb**	**CRUD**	**Entire Collection**	**Specific Item**
1	POST	Create	201 (Created)	404 (Not Found), 409 (Conflict) if resource exists.
2	GET	Read	200 (OK)	200 (OK)
3	PUT	Update/Replace	404 (Not Found)	200 (OK) or 204 (No Content). 404 (Not Found *)
4	PATCH	Update/Modify	404 (Not Found)	200 (OK) or 204 (No Content). 404 (Not Found *)
5	DELETE	Delete	404 (Not Found)	200 (OK). 404 (Not Found *)

(*):404 (Not Found), if ID not found or invalid.

Explanation

1. 'Location' header with link to /customers/id containing new ID.
2. List of customers. Use pagination, sorting and filtering to navigate big lists.
3. Single customer. 404 (Not Found), if ID not found or invalid, unless you want to update/replace every resource in the entire collection.
4. if ID not found or invalid, unless you want to modify the collection itself.
5. if ID not found or invalid, unless you want to delete the whole collection—not often desirable.

Definition of the base URL of a resource

Idea fundamental

A SW implemented with RESTful technology must define the *base direction* of each one of the services that offers to its clients

Example:

```
import org.glassfish.jersey.client.ClientConfig;
    ClientConfig clientConfig = new ClientConfig();
import jakarta.ws.rs.client.Client;
    Client client = ClientBuilder.newClient(clientConfig);
import jakarta.ws.rs.client.WebTarget;
        WebTarget webTarget = client.target(getBaseURI());
...
        WebTarget todoWebTarget = webTarget.path("rest");
        WebTarget helloworldWebTarget = todoWebTarget.path("
            todos");
...
private static URI getBaseURI(){
                return UriBuilder.fromUri("http://localhost
                    :8080/p2-rest/").build();
```

Data exchange between the client and the *service*

```
import jakarta.ws.rs.client.Invocation;
        Invocation.Builder invocationBuilder =
            helloworldWebTargetWithQueryParam.request(MediaType
            .TEXT_XML);

import jakarta.ws.rs.core.Response;
        Response response = invocationBuilder.put(Entity.entity(
            todo, MediaType.TEXT_XML));
        ...
        Response response4 = invocationBuilder4.get();
        ...
        Response response5 = invocationBuilder4.delete();
```

We have to program with the prior pattern each one of the read(GET()), write(PUT()), update(PATCH(),POST()) operations..., which are going to be supported by the service

JAXB

Fundamental idea

- This is about a specific standard (*Java Architecture for XML Binding*) of use for obtaining a correspondence between 'regular' data objects (*POJO*) and their representation in XML
- The associated framework allow us to read/write from/in Java objects and in/from XML documents

JAXB annotations

`@XmlRootElement(namespace = "space_of_names")`	Root element of an "XML tree"
`@XmlType(propOrder = "field1",... )`	writting order for class fields into the XML
`@XmlElement(name = "newName")`	The XML element that is used instead[a]

[a]It only needs to be used if it is different from the name assigned by the JavaBeans framework

DAO

Definition

DAO or “data access object” is an object that provides an abstract interface to a DB or any other mechanism for persistence of entities of software applications

- DAO provide us with some operations on specific data without disclosing, however, the supporting DB low-level details to the user applications
- It also provide us a mapping between operation calls performed in an application to the *persistence layer* of a Web service

DAO Todo

```java
import java.util.HashMap;
import java.util.Map;
//import the data domain model
public enum TodoDao {
  INSTANCE;//for singleton.
  private Map<String, Todo> contentsProvider = new HashMap<
      String, Todo>();
  private TodoDao() {
    Todo todo = new Todo("1", "Learn␣REST");
    todo.setDescription("Read␣http://lsi.ugr.es/dsbcs/Documentos
        /Practica/practica3.html");
    contentsProvider.put("1", todo);
    todo = new Todo("2", "Learn␣something␣about␣DSBCS");
    todo.setDescription("Read␣all␣the␣material␣placed␣at␣http://
        https://prado1718.ugr.es/moodle/course/view.php?id
        =63658");
    contentsProvider.put("2", todo);  }
  public Map<String, Todo> getModel(){
    return contentsProvider;  }
}
```

Data domain

```
@XmlRootElement
public class Todo{
  private String id;
  private String summary;
  private String description;

  public Todo(){
  }
  public Todo (String id, String summary){
    this.id = id;
    this.summary = summary;
  }
  public String getId() {
    return id;
  }
  public void setId(String id) {
    this.id = id;
  }
 ...
}
```

Resource

```java
import jakarta.ws.rs.GET;
import jakarta.ws.rs.POST;
import jakarta.ws.rs.PUT;
import jakarta.ws.rs.DELETE;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.PathParam;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.Consumes;
import jakarta.ws.rs.FormParam;
import jakarta.ws.rs.core.Context;
import jakarta.ws.rs.core.MediaType;
import jakarta.ws.rs.core.Request;
import jakarta.ws.rs.core.Response;
import jakarta.ws.rs.core.UriInfo;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

Resource II

```java
import jakarta.servlet.http.HttpServletResponse;
import jakarta.xml.bind.*;
@Path("/todos")//Mapping of resource into URL: todos
@Path("/todos")
public class TodosRecurso {
  //Devolvera la lista de todos lo elementos contenidos en el
      proveedor al
  //navegador del usuario.
  @Context
  UriInfo uriInfo;
  @Context
  Request request;
  String id;
//Devolvera la lista de todos lo elementos contenidos en el
    proveeedor
//a las aplicaciones cliente
  @GET
  @Produces(MediaType.APPLICATION_JSON)
  public List<Todo> getTodosBrowser() {
    List<Todo> todos = new ArrayList<Todo>();
    todos.addAll(TodoDAO.INSTANCE.getModel().values());
    return todos;
  }
```

Recurso III

```java
@GET
   @Path("cont")
   @Produces(MediaType.TEXT_PLAIN)
   public String getCount() {
               int cont = TodoDAO.INSTANCE.getModel().size();
               return String.valueOf(cont);
   }
   @PUT
   @Consumes(MediaType.TEXT_XML)
   public Response putTodo(JAXBElement<Todo> todo) {
        Todo c = todo.getValue();
        return putAndGetResponse(c);
   }
   @DELETE
   public void deleteTodo() {
        Todo c = TodoDAO.INSTANCE.getModel().remove(id);
        if(c==null)
             throw new RuntimeException("Delete:␣Todo␣con␣
                  identificador␣" + id + "␣no␣se␣encuentra");
   }
}
```

Service deployment description

```
<?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" id="
     WebApp_ID" version="3.1">
<display-name>mio.jersey.segundo</display-name>
<servlet>
<servlet-name>Servicio REST de Jersey</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</
     servlet-class>
<!-- Registra recursos que estan ubicados dentro de mio.jersey.
     primero-->
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>mio.jersey.segundo.modelo</param-value>
 </init-param>
<load-on-startup>1</load-on-startup>
 </servlet>
<servlet-mapping>
<servlet-name>Servicio REST de Jersey</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Test class for the implemented Web service

```java
package mio.jersey.segundo.cliente;
import java.net.URI;
import jakarta.ws.rs.core.MediaType;
import jakarta.ws.rs.core.Response;
import jakarta.ws.rs.core.UriBuilder;
import jakarta.ws.rs.client.Client;
import jakarta.ws.rs.client.ClientBuilder;
import jakarta.ws.rs.client.Invocation;
import jakarta.ws.rs.client.WebTarget;
import jakarta.ws.rs.client.Entity;
import org.glassfish.jersey.client.ClientConfig;
import mio.jersey.segundo.modelo.Todo;
import jakarta.ws.rs.core.Form;
```

Test class for the implemented Web service-II

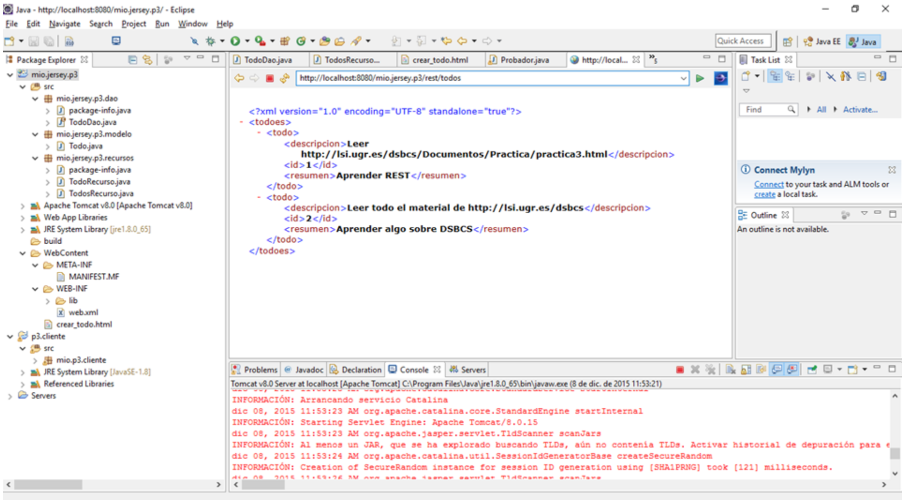
```java
public class Test {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ClientConfig clientConfig = new ClientConfig();
                Client client = ClientBuilder.newClient(
                    clientConfig);
                WebTarget webTarget = client.target(getBaseURI()
                    );
                //crearse un todo
                Todo todo = new Todo("99", "Este␣es␣el␣resumen␣
                    de␣otro␣registro");
                WebTarget todoWebTarget = webTarget.path("rest")
                    ;
                WebTarget helloworldWebTarget = todoWebTarget.
                    path("todos");
                WebTarget helloworldWebTargetWithQueryParam =
                    helloworldWebTarget.queryParam("greeting",
                    "Hi␣World!");
                 //////////////////
                Invocation.Builder invocationBuilder =
                    helloworldWebTargetWithQueryParam.request(
                    MediaType.TEXT_XML);
```

Test output

```
200
Mostrar contenido del recurso como HTML
{"id":"99","resumen":"Este es el resumen de otro registro"}
Mostrar el codigo de respuesta:5
204
Mostrar el codigo de respuesta:6
200
Mostrar contenido del recurso como HTML
[{"descripcion":"Leer http://lsi.ugr.es/dsbcs/Documentos/Practica/practica3.html","id":"1","resumen":"Aprender REST"},{"descripcion":"Leer todo el material de http://lsi.ugr.es/dsbcs","id":"2","resumen":"Aprender algo sobre DSBCS"}]
Mostrar el codigo de respuesta:6
200
Mostrar el codigo de respuesta:7
200
Mostrar contenido del recurso como HTML
[{"descripcion":"Leer http://lsi.ugr.es/dsbcs/Documentos/Practica/practica3.html","id":"1","resumen":"Aprender REST"},{"descripcion":"Leer todo el material de http://lsi.ugr.es/dsbcs","id":"2","resumen":"Aprender algo sobre DSBCS"}]
Formulario respuesta 200
<!doctype html><html lang="es"><head><title>Segundo ejemplo con
```

CRUD service deployed in a Tomcat server



```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <todoes>
  - <todo>
      <descripcion>Leer
        http://lsi.ugr.es/dsbcs/Documentos/Practica/practica3.html</descripcion>
      <id>1</id>
      <resumen>Aprender REST</resumen>
    </todo>
  - <todo>
      <descripcion>Leer todo el material de http://lsi.ugr.es/dsbcs</descripcion>
      <id>2</id>
      <resumen>Aprender algo sobre DSBCS</resumen>
    </todo>
</todoes>
```

```
Tomcat v8.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (8 de dic. de 2015 11:53:21)
INFORMACIÓN: Arrancando servicio Catalina
dic 08, 2015 11:53:23 AM org.apache.catalina.core.StandardEngine startInternal
INFORMACIÓN: Starting Servlet Engine: Apache Tomcat/8.0.15
dic 08, 2015 11:53:23 AM org.apache.jasper.servlet.TldScanner scanJars
INFORMACIÓN: Al menos un JAR, que se ha explorado buscando TLDs, aún no contenía TLDs. Activar historial de depuración para e
dic 08, 2015 11:53:24 AM org.apache.catalina.util.SessionIdGeneratorBase createSecureRandom
INFORMACIÓN: Creation of SecureRandom instance for session ID generation using [SHA1PRNG] took [121] milliseconds.
dic 08, 2015 11:53:26 AM org.apache.jasper.servlet.TldScanner scanJars
```