Práctica 2. ETIQUETAS REST

DESARROLLO DE SERVICIOS REST JERSEY Y CLIENTES JQUERY

Guadalupe Ortiz Bellot
Departamento de Ingeniería Informática

PRÁCTICA 2 ETIQUETAS REST

- **Familiarízate con las etiquetas de Rest**
  - **@Path**
  - **@PathParam**
  - **@QueryParam**
  - **@XmlRootElement**
  - **@Post**
  - **@FormParam**
  - **@PUT, @DELETE**

Familiarízate con las etiquetas de Rest

@Path

- Añade la siguiente función dentro de Hello.java:

```
@GET
@Path("/hello2")
@Produces(MediaType.TEXT_PLAIN)
  public String sayPlainTextHello2(){return "Hello Plain 2";}
```

- Fíjate que le hemos añadido el Path a la función
- Fíjate que el servidor Tomcat se reinicia solo para añadir la nueva función (debes esperar a que termine para poder probarla)
- Ahora para poder probarla habría que añadir ese path a toda la ruta raíz: http://localhost:8080/HelloWorld/demo/hello/hello2
- Pruébalo
- NOTA: la URI contiene el path de la clase seguido del del método:
  - No es necesario que la clase tenga path, en tal caso se usa solo el del método.
  - Si la clase lleva path, pero el método no, busca un método que cumpla con la solicitud (en este caso GET que devuelve texto plano) que no tenga path; si hay más da una dará lugar a error.

Familiarízate con las etiquetas de Rest

@PathParam

- Añade la siguiente función dentro de Hello.java:

```
@GET
@Path("/helloId/{oid}")
@Produces(MediaType.TEXT_PLAIN)
  public String sayHelloWithId(@PathParam("oid")int id)
{return "Hello Plain " + id;}
```

- Nota: nuevas etiquetas requieren nuevos Import (import javax.ws.rs.PathParam;). **OJO, siempre con la raíz javax.ws**
- Le hemos añadido al Path un identificador de parámetro entre llaves
- En los parámetros de la función hemos identificado dicho parámetro poniendo `@PathParam("oid")` antes del tipo del parámetro correspondiente, siendo oid el identificador usado en el path
- Para poder probarla hay que reemplazar {oid} por el valor que queramos que tenga ese parámetro, i.e: http://localhost:8080/HelloWorld/demo/hello/helloId/5
- Pruébalo
- Si quieres añadir multiples parámetros separalos por /, cada uno con sus llaves :`@Path("/helloDate/{year}/{month}/{day}")`

Familiarízate con las etiquetas de Rest

@QueryParam

- Nos sirve para enviar directamente los parámetros de un formulario HTML por GET.
- Añade la siguiente función dentro de Hello.java:

```
@GET
@Path("/helloQuery")
@Produces(MediaType.TEXT_PLAIN)
  public String sayHelloWithQuery(@QueryParam("name")String
name,@QueryParam("surname")String surname )
{return "Hello " + name + " "+ surname;}
```

- Para poder probarla hay que añadir los parámetros en el path como sigue:
- http://localhost:8080/HelloWorld/demo/hello/helloQuery**?name=Guadalupe&surname=Ortiz**
- Pruébalo

Familiarízate con las etiquetas de Rest

@XmlRootElement (i)

- @XmlRootElement se añade a una clase que se quiere usar como parámetro de retorno. Nos va a permitir que se hagan transformaciones automáticas a tipo XML o tipo JSON
- Crea una nueva clase:

```
package nombrePaquete;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement
public class MyDate {
  private int day;
  private int month;
  private int year;

  public int getDay() {return day;}
  public void setDay(int day) {this.day = day;}
  public int getMonth() {return month;}
  public void setMonth(int month) {this.month = month;}
  public int getYear() {return year;}
  public void setYear(int year) {this.year = year;} }
```

Familiarízate con las etiquetas de Rest @XmlRootElement (ii)

- Añade un nuevo método a tu clase Hello:

```
@GET
@Path("/dateJSON")
@Produces({"application/json"})
public MyDate getDate_JSON() {
    MyDate oneDate = new MyDate();
    oneDate.setDay(25);
    oneDate.setMonth(12);
    oneDate.setYear (2014);
    return oneDate;}
```

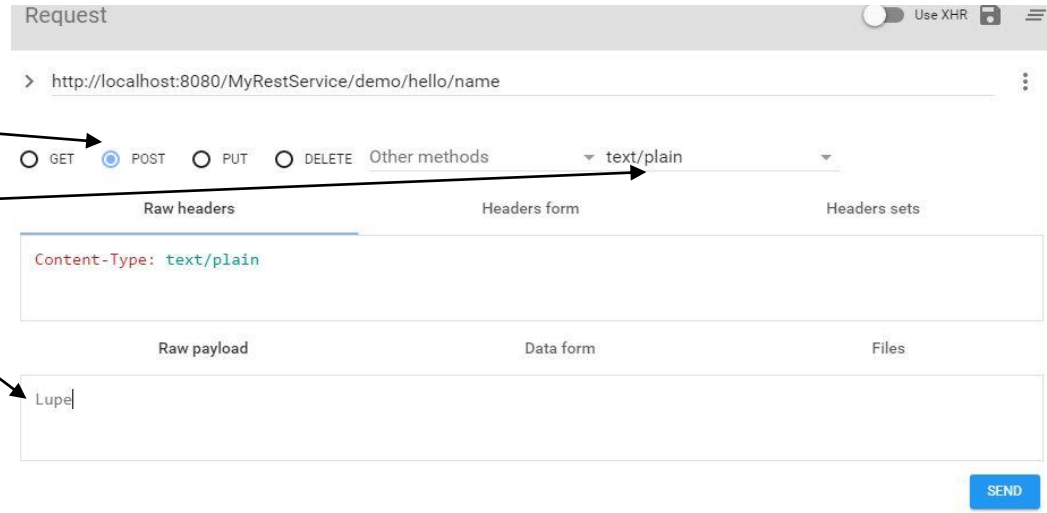
- Pruébala
- NOTA. Puedes usar tanto `@Produces({"application/json"})` como `@Produces({MediaType.APPLICATION_JSON})`
- Pruébalo
- Pruébalo ahora poniendo `@Produces({"application/xml"})` o `@Produces({MediaType.APPLICATION_XML})`

Familiarízate con las etiquetas de Rest @POST (i)

- Añade un nuevo método:

```
@POST
@Path("/name")
@Consumes(MediaType.TEXT_PLAIN)
@Produces(MediaType.TEXT_PLAIN)
public String HelloName(String myName){
return "Hello "+myName;}
```

- Hemos añadido la etiqueta que indica el tipo de dato que le vamos a enviar (@Consumes)
- Pruébalo:
  - Selecciona POST
  - En el desplegable elige text/plain
  - En el campo Payload escribe el texto a enviar



Familiarízate con las etiquetas de Rest @POST (ii)

- Vamos a ver ahora un ejemplo con formato JSON
  1. Recuerda que el JSON debe ir entre llaves y está compuesto por una serie de claves, **separadas por comas, entre comillas** seguidas de **dos puntos y** del **valor** correspondiente.

     Por ejemplo:

     {

     “Nombre”:”Mariano”,

     “Edad”:33

     }
  1. Además para que se pueda hacer la conversion automática a la clase correspondiente, **las claves deben coincidir con los atributos privados de la clase**.

Familiarízate con las etiquetas de Rest @POST (iii)

- Añade un nuevo método:

```
@POST
@Path("/myDate2015")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public MyDate dateToString(MyDate myDate){
myDate.setYear(2015);
return myDate;}
```

- Pruébalo
- Nota: En el **desplegable** elige **application/json**.
- En el **raw payload a**segúrate de enviar los datos correctamente en formato JSON. Recuerda que las **claves deben coincidir con los atributos privados de la clase** (en el ejemplo day, month, year). Visita la **tranparencia anterior** para ver el formato del JSON.
- Prueba ahora la misma función enviando un JSON y recibiendo un XML

Familiarízate con las etiquetas de Rest

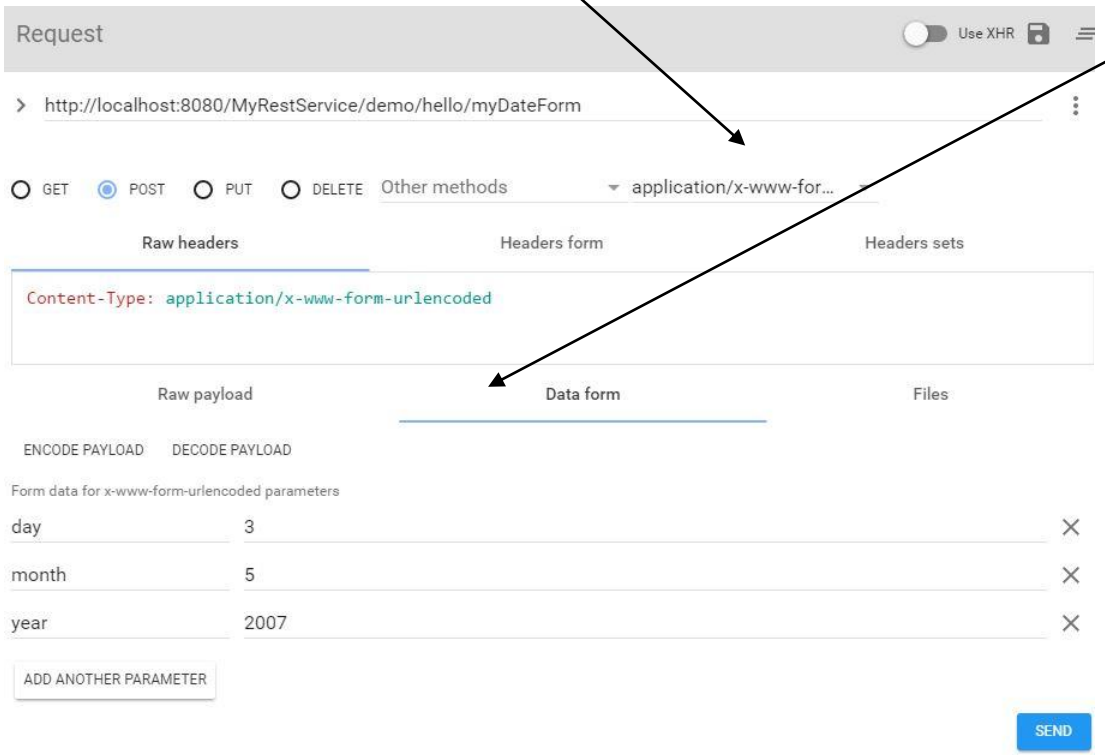
@FormParam (i)

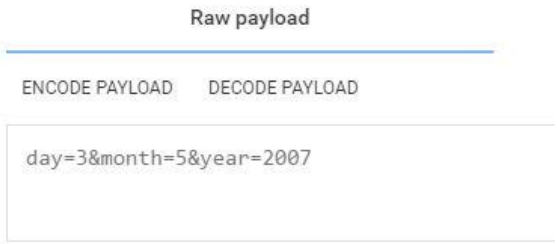
- Se utiliza para enviar directamente (mediante POST) parámetros de un formulario HTML
- Añade un nuevo método:

```
@POST
@Path("/myDateForm")
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public MyDate dateToText(@FormParam("day") int myDay,
@FormParam("month") int myMonth, @FormParam("year") int
myYear){
MyDate myDate = new MyDate();
myDate.setDay(myDay);
myDate.setMonth(myMonth);
myDate.setYear(myYear);
return myDate;
}
```

Familiarízate con las etiquetas de Rest @FormParam (ii)

- Pruébalo. Selecciona en el desplegable application/x-www-form-urlencoded
- Puedes añadir los parámetros uno a uno como Data Form o directamente en el raw payload separados por &.





Familiarízate con las etiquetas de Rest

@PUT, @DELETE (i)

- Para poder modificar y borrar, y después comprobar que se ha efectuado correctamente vamos a crearnos una estructura estática dentro de la clase Hello:

```
private static Map<String, MyDate> myMap = new HashMap<>();
    static{
    MyDate myDate = new MyDate();
    myDate.setDay(25);
    myDate.setMonth(12);
    myDate.setYear(2014);
    myMap.put("Navidad", myDate);
    myMap.put("Nochebuena", myDate);
    myMap.put("AnioNuevo", myDate);
    myMap.put("Reyes", myDate);
    }
```

Familiarízate con las etiquetas de Rest @PUT, @DELETE (ii)

- Añade un nuevo método:

```
@PUT
@Path("/modifyDate/{date}")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.TEXT_PLAIN)
  public String modifyImportantDate(@PathParam("date")
String key, MyDate myDate) {
myMap.put(key, myDate);
return "Date modified"; }
```

- Fíjate que para invocarlo tienes que añadir en el path el nombre de la fecha que quieres modificar (por ejemplo Navidad) y como raw payload el json con la nueva fecha.
- Después puedes comprobarlo hacienda un GET de la fecha.

- ¿Cómo sería el delete…?

Familiarízate con las etiquetas de Rest

@PUT, @DELETE (iii) Una posible solución

```
@GET
@Path("/allDates")
@Produces(MediaType.APPLICATION_JSON)
public Map<String, MyDate> allDates() {
return myDateMap;
}

@DELETE
@Path("/deleteDate/{date}")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.TEXT_PLAIN)
  public String deleteDate(@PathParam("date") String key) {
  myDateMap.remove(key);
return "Date deleted"; }
```