Aplicacions Distribuïdes (AD)

Práctica 3: Desarrollo de servicios web con REST

Creación de servicios web basados en REST

En esta práctica vamos a crear servicios Web RESTful (RESTWS de aquí en adelante) con *Apache Netbeans*. Lo primero es crear una nueva aplicación web. Podéis llamar a dicha aplicación `RestAD`.

Por defecto Netbeans 12 crea un RESTWS en cada nueva aplicación web. Está en la carpeta `RESTful Web Services` folder y se llama `JavaEE8Resource`. La Figura 1 muestra la carpeta con el servicio y la operación `ping` que se crea automáticamente. Se tienen que implementar los métodos HTTP del RESTWS. Utilizaremos `GET` y `POST`, pero se pueden utilizar otros.

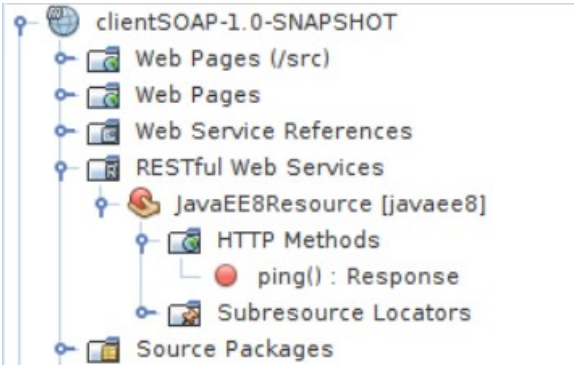

Figura 1. Localización del servicio RESTful.

La operación ping es accesible desde el navegador con la siguiente URL (`http://localhost:8080/<webapp>/resources/javaee8`) tal y como muestra la Figura 2.

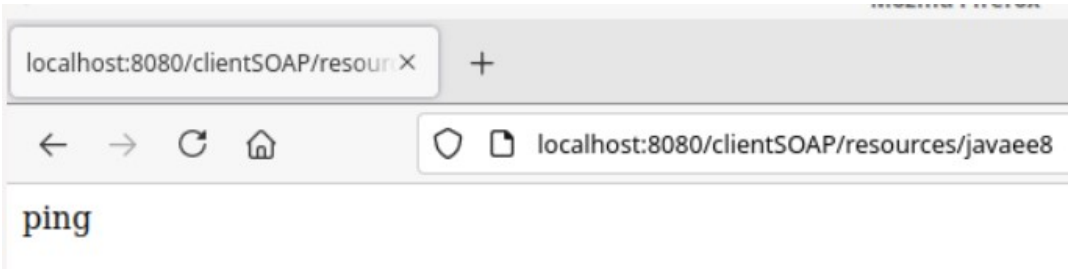

Figura 2. Resultado operación ping.

También es accesible desde la opción `Test Resource Uri` cuando se pulse el botón derecho sobre el método HTTP (por ejemplo, `ping`) que se muestra en Figura 3.

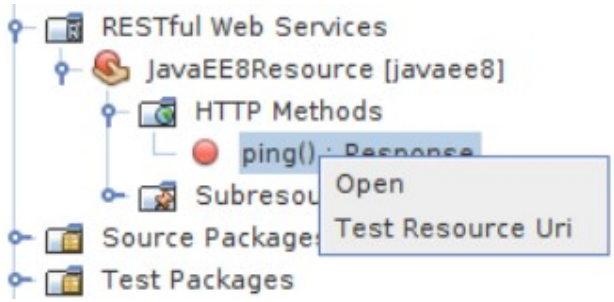

Figure 3. Opción Test Resource Uri.

Los ficheros que contienen el código se muestran en la Figura 4. `JAXRSConfiguration.java` contiene información de la configuración general y `JavaEE8Resource.java` contiene el código de las operaciones.

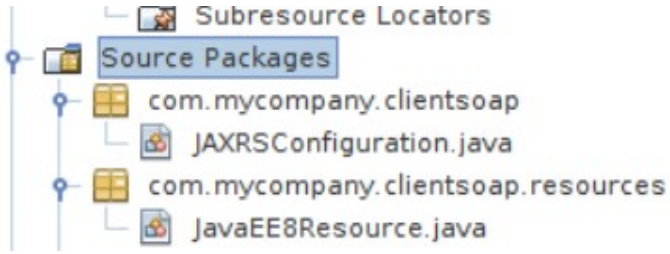

Figura 4. Código fuente servicio RESTful.

Responde a las siguientes cuestiones:

a) ¿Por qué podemos acceder la servicio desde el navegador?
b) ¿Qué método HTTP se está utilizando?
c) ¿Con qué tipo MIME? Comprueba la características de red en las opciones de desarrolladores de los navegadores (F12 en sistemas Windows y Linux).

Implementando el servicio

En la Figura 5 se muestra la arquitectura de esta práctica. Se tienen que implementar dos nuevas aplicaciones web, una con el servicio web REST y otra que será el cliente del servicio REST. En ambas aplicaciones se puede reaprovechar código implementado en la práctica 2 (acceso a base de datos, formularios de petición de información, etc.), pero lo tendréis que organizar de forma diferente, ya que en esta práctica estamos implementando una arquitectura multinivel (explicada en clase de teoría el 20/09).

Así, en la aplicación cliente deberéis gestionar la sesión del usuario como en la práctica 2, pero, para saber si el login es correcto, deberéis llamar a la operación correspondiente del servicio web, ya que la aplicación web cliente no tiene acceso a la base de datos (ver Figura 5). Se recomienda el uso de las clases `java.net.URL` y `java.net.HttpURLConnection` para implementar la comunicación entre el servicio web y la aplicación cliente.

Figura 5. Arquitectura servicio web REST.

Las cabeceras de las operaciones REST deben ser tal y como se muestra a continuación (consultad cualquier modificación con la profesora). Se pueden añadir operaciones extra sin problema. La clase que se devuelve como respuesta de las operaciones es `javax.ws.rs.core.Response`.

```
/**
* OPERACIONES DEL SERVICIO REST
*/

/**
* POST method to login in the application
* @param username
* @param password
* @return
*/
@Path("login")
@POST
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.APPLICATION_JSON)
public Response Login(@FormParam("username") String username,
        @FormParam("password") String password)
```

```
 /**
 * POST method to register a new image
 * @param title
 * @param description
 * @param keywords
 * @param author
 * @param creator
 * @param capt_date
 * @return
 */
 @Path("register")
 @POST
 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
 @Produces(MediaType.APPLICATION_JSON)
 public Response registerImage (@FormParam("title") String title,
         @FormParam("description") String description,
         @FormParam("keywords") String keywords,
         @FormParam("author") String author,
         @FormParam("creator") String creator,
         @FormParam("capture") String capt_date)

/**
 * POST method to modify an existing image
 * @param id
 * @param title
 * @param description
 * @param keywords
 * @param author
 * @param creator, used for checking image ownership
 * @param capt_date
 * @return
 */
 @Path("modify")
 @POST
 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
 @Produces(MediaType.APPLICATION_JSON)
 public Response modifyImage (@FormParam("id") String id,
         @FormParam("title") String title,
         @FormParam("description") String description,
         @FormParam("keywords") String keywords,
         @FormParam("author") String author,
         @FormParam("creator") String creator,
         @FormParam("capture") String capt_date)

/**
 * POST method to delete an existing image
 * @param id
 * @return
 */

 @Path("delete")
 @POST
 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
 @Produces(MediaType.APPLICATION_JSON)
 public Response deleteImage (@FormParam("id") String id)
```

```
/**
 * GET method to list images
 * @return
 */
 @Path("list")
 @GET
 @Produces(MediaType.APPLICATION_JSON)
 public Response listImages ()

/**
 * GET method to search images by id
 * @param id
 * @return
 */
 @Path("searchID/{id}")
 @GET
 @Produces(MediaType.APPLICATION_JSON)
 public Response searchByID (@PathParam("id") int id)

/**
 * GET method to search images by title
 * @param title
 * @return
 */
 @Path("searchTitle/{title}")
 @GET
 @Produces(MediaType.APPLICATION_JSON)
 public Response searchByTitle (@PathParam("title") String title)

/**
 * GET method to search images by creation date. Date format should be
 * yyyy-mm-dd
 * @param date
 * @return
 */
 @Path("searchCreationDate/{date}")
 @GET
 @Produces(MediaType.APPLICATION_JSON)
 public Response searchByCreationDate (@PathParam("date") String date)

/**
 * GET method to search images by author
 * @param author
 * @return
 */
 @Path("searchAuthor/{author}")
 @GET
 @Produces(MediaType.APPLICATION_JSON)
 public Response searchByAuthor (@PathParam("author") String author)

/**
 * GET method to search images by keyword
 * @param keywords
 * @return
 */
 @Path("searchKeywords/{keywords}")
 @GET
 @Produces(MediaType.APPLICATION_JSON)
 public Response searchByKeywords (@PathParam("keywords") String keywords)
```

Trabajo adicional

Para mejorar la nota de esta práctica, podéis implementar una operación de búsqueda combinada, por ejemplo, por título y autor o por título y descripción (o cualquier otra combinación que se os ocurra, que incluya al menos dos criterios de búsqueda).

Entrega

Se debe entregar un documento que responda a las preguntas iniciales, el código de la práctica y un informe donde se describa brevemente el trabajo realizado y las ampliaciones realizadas.

Consultad la fecha de entrega en el Racó.