Desarrollo Basado en Componente y Servicios

Servicios Web de tipo REST

Parte II: Conceptos Avanzados

Introducción

Vista la creación de servicios web tipo REST básicos y cómo probarles para entender mejor sus bases conceptuales, vamos a avanzar un poco más añadiendo elementos que permiten crear servicios más complejos.

1. URI variables

Hasta ahora se han usado URIs fijas para crear el servicio web. Como ya hemos indicado, en los servicios web RESTful cada el servicio va indicado en la operación HTTP y en la URI. Por lo tanto, necesitaríamos definir tantos métodos en nuestra clase Java como URIs tenga nuestro servicio. Esto es poco práctico desde el punto de vista de la programación, por lo que Java permite al construir el servicio web utilizar parte/s de la ruta como variables, y, así, un mismo método puede realizar operaciones para distintos valores de una misma parte de la URI.

De igual manera necesitamos esa opción para invocar servicios sobre elementos que son definidos por el usuario al reclamar el servicio. Por ejemplo, el servicio es personalizado y cada usuario accede mediante una URL del tipo: http://URIbaseServicioWeb/idCliente, donde la parte "idCliente" es variable.

Esto se implementa en Java con la siguiente sintaxis en la anotación @Path: @Path("{usuario}"). Eso quiere decir que esa parte de la URI es una variable. Esa variable se pasa como argumento al método usando la anotación @PathParam antes de argumento. Esto se puede emplear en cualquier operación HTTP. Un ejemplo en GET sería:

```
@Path("{usuario}")
@GET
@Produces("text/html")
Public String getMensaje(@PathParam("usuario") String user) {
   Return "<html><body><h1>Hola " + user + " </h1></body></hmtl>";
   }
```

La anotación @PathParam permite distinguir los argumentos al método pasados como parte de la URL, de aquellos que van en el cuerpo de la petición (request), como hemos hecho hasta ahora. Un ejemplo que emplea ambos sería:

```
@POST
@Consumes("text/plain")
@Path("addMensaje/{usuario}")
public void addMensaje(@PathParam("usuario") String user, String saludo) {
   System.out.println(saludo + " " + user);
   }
```

La llamada en el cliente Java a este servicio quedaría:

```
public void addMensaje(String usuario, String saludo) {
    WebTarget resource = webTarget;
    resource = resource.path("addMensaje/"+ usuario);
    resource.request(javax.ws.rs.core.MediaType.TEXT_PLAIN)
            .post(javax.ws.rs.client.Entity.entity(saludo, javax.ws.rs.core.MediaType.TEXT_PLAIN));
}
```

Se puede incluir más de un elemento variable en la URL:

```
@GET
@Produces("text/html")
@Path("list/{usuario}/{categoria}")
Public String listVinos(@PathParam("usuario") String usuario, @PathParam("categoria") String tipo) {
   ....
   }
```

Ejercicio 3. Probar los ejemplos mostrados en este apartado.

2. Variables o parámetros en la URL (query parameters)

Una alternativa para enviar información del cliente al servidor, hasta ahora sólo hemos usado el cuerpo de la petición (ejemplos con PUT y POST), es mediante las variables o parámetros en la URL. Un ejemplo sería: http://www.peliculas.com/accion**?actor=Ann Smith&director=John Carter**, donde la parte de variables de la URL es la resaltada en negrita.

Para capturar esto al desarrollar servicios web RESTful con la tecnología Java se usan las anotaciones **@QueryParam** en los argumentos al método correspondientes. Ej. con la URL anterior, suponiendo que la base es "www.peliculas.com":

```
@GET
@Produces("text/html")
@Path("/accion")
public String pruebaQuery(@QueryParam("actor") String actor, @QueryParam("director") String dir) {
      ....
   }
```

De igual manera se puede usar con el resto de operaciones. Lo podemos usar en conjunción con el resto de opciones vistas hasta ahora: contenido del cuerpo de la petición y partes variables en la URL (anotaciones *@PathParam*).

Ejercicio 4. Probarlo mediante cualquier ejemplo. Lo más sencillo es probarlo con GET, ya que se puede poner directamente la URL en el navegador sin necesidad de usar un cliente.

3. Contenido de un formulario HTML

Una última opción de enviar información del cliente al servidor es mediante los formularios HTML. Lo primero a recordar es que HTML sólo permite usar dos operaciones al enviar el formulario: GET y POST, por lo tanto sólo podremos capturar el contenido de un formulario en esos casos.

Como en los casos anteriores, esos datos son pasados al método asociado a la URL como argumentos. Para identificarles, se usa ahora la anotación @FormParam.

Ejemplo:

- Página HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01">
<html>
<head>
  <title>Ejemplo de formulario</title>
</head>
<p>Introduce nombre de usuario y clave</p>
  <form action="http://localhost:8080/RESTful_pruebas/webresources/prueba1" method="post">
    Username: <input type="text" name="usuario" value="Invitado"><br>
    Password: <input type="password" name="clave"><br><br>
                <input type="hidden" name="control" value="formulario">
   <input type="submit" value="Enviar">
    <input type="reset" value="Vaciar">
 </form>
</body>
</html>
```

- Código Java del servicio:

```
@POST
public void verFormulario(@FormParam("usuario") String nombre, @FormParam("clave") String clave) {
        System.out.println("Nombre: " + nombre + " clave: " + clave);
    }
```

Ejercicio 5. Probar el código del ejemplo.