Servicios Telemáticos Avanzados

5.- SERVICIOS WEB

OpenCourseWare 2014

Maider Huarte y Gorka Prieto
Escuela Técnica Superior de Ingeniería de Bilbao
Departamento de Ingeniería de Comunicaciones
Universidad del País Vasco (UPV/EHU)

Servicios Telemáticos Avanzados:
5.- Servisios Web.odp

Servicios Web

ÍNDICE

1.- Introducción

- ¿Qué es un servicio web?
  - Definición JavaEE
  - Diferencia con aplicaciones web orientadas a presentación
  - Relación con SOA
- ¿Qué es SOA?
  -
  -
  -
  -
  -
- ¿Qué es un “servicio” SOA?

1.- Introducción

- Tipos de servicios web
  - Servicios Web “grandes”
    -
    -
  - Servicios Web “ligeros”
    -
    -
    -

2.- Servicios Web REST

2.1.- Introducción

- Arquitectura sw de 2 niveles con protocolo de comunicación sin estado
- Recursos
- Principios para aplicaciones REST
  - Recursos y URIs
  - Operaciones de manipulación de recursos
    -
    -
    -
    -

2.- Servicios Web REST

2.2.- Codificación de recursos REST con JavaEE

- Recurso REST
  - POJO+anotaciones JAX-RS
- Anotaciones JAX-RS (clase/métodos)
  - @Path
  - @PathParam, @QueryParam, @FormParam, @CookieParam
  - @GET, @POST, @PUT, @DELETE
  - @Consumes, @Produces
- Métodos
  - Públicos
  - Retorno: void, tipo Java primitivo, javax.ws.rs.core.Response

2.- Servicios Web REST

2.2.- Codificación de recursos REST con JavaEE

- Ejemplo básico
  - URL: **http://*dominio*/*servicio*/rest/Ej1**

Ej1.java

```
//http://localhost:8080/T5-1AD_RESTWS_1/rest/Ej1

package sta;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/Ej1")
public class Ej1 {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String mensajeBienvenida() {
        return "KAIXO MUNDUA!";
    }
}
```

2.- Servicios Web REST

2.3.- Uso de Parámetros

- Uso de parámetros
  - **@Path(“***plantilla_URI***”)**
    - Anotación de clase o método
    - {variable}
    - {variable: expresiones regulares}
      - [a-zA-Z]*
      - [a-zA-Z][a-zA-Z_0-9]*
  - **@PathParam** *declaración_de_parámetro*
    - Anotación de parámetro de método
  - **@QueryParam** *declaración_de_parámetro*
    - Anotación de parámetro de método

2.- Servicios Web REST

2.3.- Uso de Parámetros

- Ejemplo @PathParam
  - URL: **http://***dominio***/***servicio***/rest/Ej2/patron/***lqs1***/***lqs2*

Ej2.java

```
//http://localhost:8080/T5-1AD_RESTWS_1/rest/Ej2/patron/1/2

package sta;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/Ej2")
public class Ej2 {

    @GET
    @Path("/patron/{p1}/{p2}")
    @Produces(MediaType.TEXT_PLAIN)
    public String mensaje(@PathParam("p1") String parametro1,@PathParam("p2") String parametro2) {
          return "Los valores recibidos en la URL han sido: p1="+parametro1+", p2="+parametro2;
    }
}
```

2.- Servicios Web REST

2.3.- Uso de Parámetros

- Uso de parámetros (*continuación*)
  - **@FormParam(“*nombre_del_dato*”)**
    - @POST
    - @Consumes(“application/x-www-form-urlencoded”)
  - **@CookieParam**
  - **@HeaderParam**

2.- Servicios Web REST

2.3.- Uso de Parámetros

- Ejemplo Formularios

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Página con formularios para prueba</title>
</head>
<body>
    <h1>Página con formularios para prueba</h1>
    <form action="rest/Ej3/formularioGET" method="get">
        <p><h2>FORMULARIO GET</h2>
            <input type="text" name="op1" />
            <select name="op">
                <option value="+">+</option>
                <option value="-">-</option>
                <option value="*">*</option>
                <option value="/">/</option>
            </select>
            <input type="text" name="op2" />
        </p>
        <input type="submit" value="CALCULAR" />
    </form>
    <form action="rest/Ej3/formularioPOST" method="post">
        <p><h2>FORMULARIO POST</h2>
<!-- Igual a en el formulario anterior -->
        </p>
        <input type="submit" value="CALCULAR" />
    </form>
</body>
</html>
```

Formularios.html

2.- Servicios Web REST

2.3.- Uso de Parámetros

- Ejemplo Formularios

Ej3.java

```
//http://localhost:8080/T5-2AD_RESTWS_2/Formularios.html
package sta;

import javax.ws.rs.FormParam;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Path("/Ej3")
public class Ej3 {
    @GET
    @Path("/formularioGET")
    @Produces(MediaType.TEXT_PLAIN)
    public String metodoGET(
                    @QueryParam("op1") float op1,
                    @QueryParam("op") String op,
                    @QueryParam("op2") float op2
                    ) {
        return "La operación indicada en el formulario GET ha sido: "+op1+op+op2;
    }

    @POST
    @Path("/formularioPOST")
    @Produces(MediaType.TEXT_PLAIN)
    public String metodoPOST(
                    @FormParam("op1") float op1,
                    @FormParam("op") String op,
                    @FormParam("op2") float op2
                    ) {
        return "La operación indicada en el formulario POST ha sido: "+op1+op+op2;
    }
}
```

2.- Servicios Web REST

2.4.- Tipos MIME

- Tipos MIME producidos o consumidos
  - **@Produces(“***tipo_MIME***”)**
    - Anotación de clase o método
    - Varios tipos
    - Error HTTP 406 Not Acceptable
  - **@Consumes(“***tipo_MIME***”)**
    - Anotación de clase o método
    - Varios tipos
    - Error HTTP 415 Unsupported Media Type

2.- Servicios Web REST

2.4.- Tipos MIME

- Ejemplo MIME

Ej4.java

```
import java.io.File;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.ResponseBuilder;


@Path("/Ej4")
public class Ej4 {

    private static final String FILE = "/home/sta1213/Documents/ficheroDePrueba.pdf";

    @GET
    @Produces("application/pdf")
    public Response getFile() {

        File file = new File(FILE);

        ResponseBuilder response = Response.ok((Object) file);
        response.header("Content-Disposition",
                        "attachment; filename="+FILE);
        return response.build();
    }
}
```

2.- Servicios Web REST

2.5.- Cliente JAX-RS

- Pasos

1. Obtener instancia del cliente:

```
Client client = ClientBuilder.newClient();
```

2. Establecer el target:

```
WebTarget myResource = client.target("http://example.com/webapi");
```

3. Crear el request:

```
Invocation.Builder builder =
myResource.request(MediaType.TEXT_PLAIN);
```

4. Invocar el request:

```
get(), post(), delete(), put(), head(), options()
```

- Todo en uno

```
ClientBuilder.newClient().target("http://example.com/webapi")
   .request(MediaType.TEXT_PLAIN).get()
```

2.- Servicios Web REST

2.5.- Cliente JAX-RS

- Ejemplo get()

```
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.MediaType;

Bean bean = ClientBuilder.newClient()
        .target("http://localhost:8080/Practica5/rest/shop/getItems")
        .request(MediaType.APPLICATION_XML)
        .get(Bean.class);
```

- Ejemplo post()

```
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.MediaType;

ClientBuilder.newClient()
        .target("http://localhost:8080/Practica5/rest/shop/saveItem")
        .request(MediaType.APPLICATION_XML)
        .post(Entity.entity(bean, MediaType.APPLICATION_XML));
```

3.- Configuración de la Aplicación

3.1.- Librería JAX-RS

- Implementaciones JAX-RS
  - Jersey (Oracle), RESTeasy (JBoss), etc.
  - Integrado desde Java EE 6
- Para entornos no Java EE
  - Descargar librería
  - Entrada en web.xml (deployment descriptor)

web.xml

```
<servlet>
  <servlet-name>Jersey REST Service</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
  <init-param>
   <param-name>jersey.config.server.provider.packages</param-name>
         <param-value>sta</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>Jersey REST Service</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

3.- Configuración de la Aplicación

3.1.- Librería JAX-RS

- Para entornos Java EE
  - No necesarios librería ni entrada en web.xml

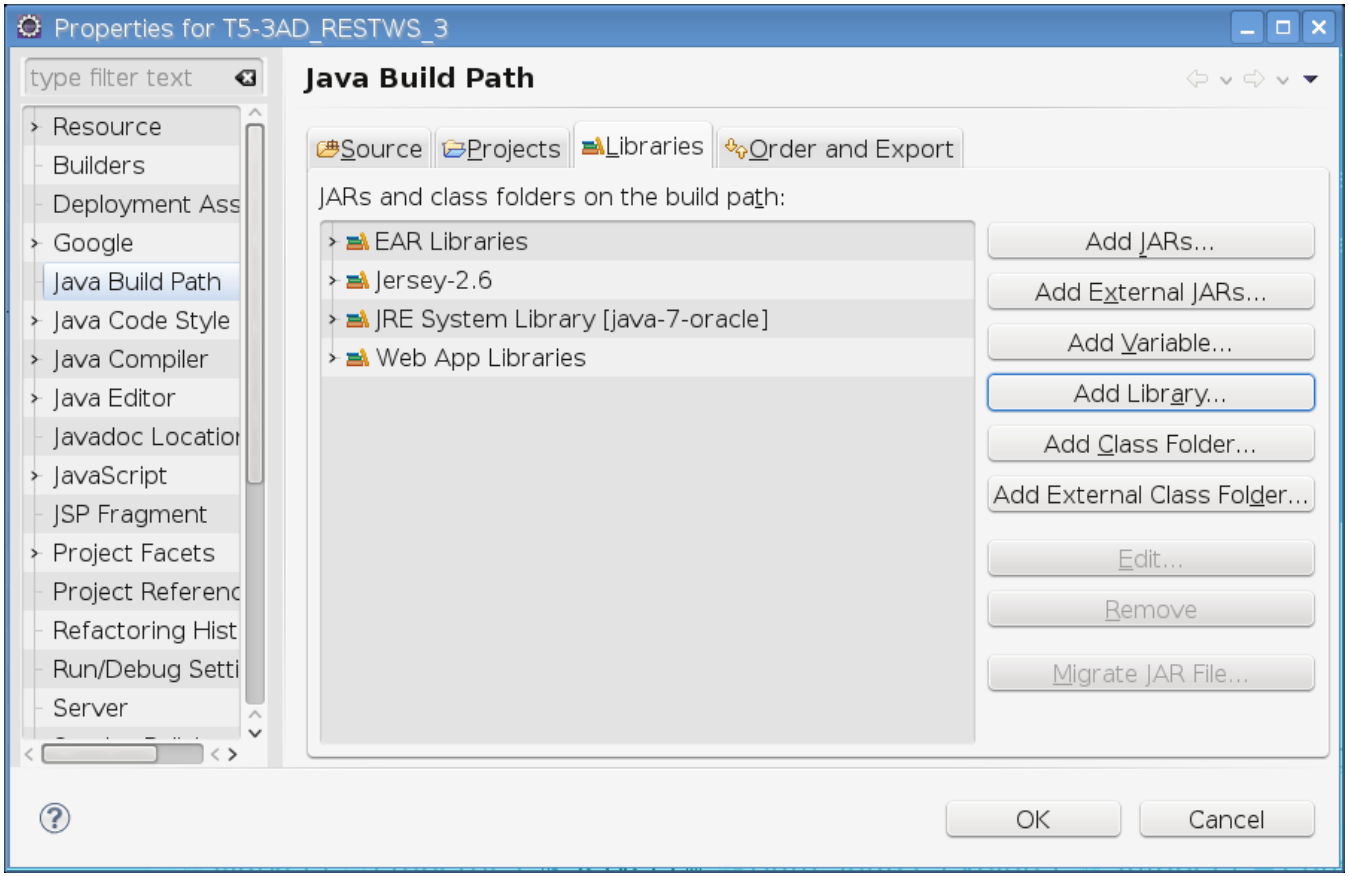
ApplicationConfig.java

```
package sta.service;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("/rest")
public class ApplicationConfig extends Application {
}
```

3.- Configuración de la Aplicación

3.2.- Eclipse

- Necesario referenciar librería
- Build Path



3.- Configuración de la Aplicación

3.2.- Eclipse

- Web Deployment Assembly