Ejercicios. Creación de Servicios Web SOAP

Índice

1. Creación de un servicio web básico

Vamos a comenzar creando un servicio web básico JAX-WS con Maven (ver apartado Creación de un servicio Web JAX-WS con Maven). Este servicio web será un *Hola Mundo* en forma de servicio. La clase que implementará el servicio tendrá como nombre `HolaMundo` y una única operación `String saluda(nombre)`, que nos devolverá un mensaje de saludo incluyendo el nombre proporcionado. Por ejemplo, si al parámetro de entrada `nombre` le damos como valor `"Miguel"`, como salida producirá la cadena `"Hola Miguel, ¿Que tal?"`.

Cuando crees el arquetipo de Maven, utiliza los siguientes nombres para:

- groupId: expertoJava
- artifactId: Sesion2-HolaMundoWS

Configurar la raiz del contexto como "HolaMundoWS" en el fichero *glassfish-web.xml*. El nombre del servicio será *Hola* (valor del atributo *serviceName* de la anotación @WebService).

Crea un cliente web Maven con Netbeans para probar el funcionamiento de dicho servicio Web. El nombre del proyecto será "Sesion2-hola-WS-client (1 punto)

2. Validación de NIFs

Vamos a implementar un servicio web Maven (con Netbeans) con una serie de métodos que nos permitan validar un NIF. El servicio tendrá como nombre `ValidaDniWS`, y estará dentro de un proyecto Maven Netbeans con nombre `Sesion2-dni-WS`. Se pide:

a) Implementar la siguiente operación (0,25 puntos):

```
boolean validarDni(String dni)
```

Esta función tomará como entrada un DNI, y como salida nos dirá si es correcto o no. El resultado será `true` si el DNI es correcto (está formado por 8 dígitos), y `false` en caso contrario. Podemos utilizar un código similar al siguiente para validar el DNI:

```
Pattern pattern =  Pattern.compile("[0-9]{8,8}");
Matcher matcher = pattern.matcher(dni);
return matcher.matches();
```

b) Implementar la siguiente operación (0,25 puntos):

```
char obtenerLetra(String dni) throws DniFault
```

Esta función tomará como entrada un DNI, y como salida nos dirá la letra que corresponde a dicho DNI. Lo primero que deberá hacer es validar el DNI (puede utilizar para ello el método definido en el apartado anterior), y en caso de no ser correcto lanzará

una excepción de tipo `DniFault` (que deberemos crear previamente) indicando el mensaje de error "El DNI no es valido". Una vez validado, calcularemos la letra que corresponda al DNI. Para ello deberemos:

- Obtener el índice de la letra correspondiente con:

```
dni % 23
```

- La letra será el carácter de la siguiente cadena que esté en la posición obtenida anteriormente:

```
"TRWAGMYFPDXBNJZSQVHLCKE"
```

Nota

El cliente de prueba no trata correctamente la recepción de un *SOAP Fault* como respuesta, ya que la excepción que genera éste hace que falle la misma aplicación de pruebas. Para poder tratar la excepción de forma correcta deberemos hacerlo con un cliente propio, que se implementará en el apartado (d).

c) Implementar la siguiente operación (0,25 puntos):

```
boolean validarNif(String nif)
```

Esta función tomará como entrada un NIF, y como salida nos dirá si es correcto o no. El resultado será `true` si el NIF es correcto (está formado por 8 dígitos seguidos de la letra correcta), y `false` en caso contrario. Para hacer esta comprobación se pueden utilizar las dos funciones anteriores: se comprobarán los 8 primeros caracteres con la función `validarDni` y posteriormente se comprobará si la letra es la correcta utilizando la función `obtenerLetra`.

d) Implementar un cliente Maven Java con Netbeans que acceda a dicho servicio e invoque la operación *obtenerLetra*. El cliente deberá crearse en un proyecto con nombre `Sesion2-dni-WS-client`. Tened en cuenta que debéis capturar la excepción que hemos definido en el apartado (b). La clase correspondiente del *stub* la podéis ver en *Files*, dentro del directorio *target/generated-sources/wsimport*, con el sufijo *_Exception*. Concretamente podéis ver que la excepción es de tipo DniFault_Exception. Prueba con un DNI válido, y haz una segunda llamada con un DNI inválido para comprobar que se lanza la excepción (0,25 puntos).

3. Tienda de DVDs

Nuestro negocio consiste en una tienda que vende películas en DVD a través de Internet. Para dar una mayor difusión a nuestro catálogo de películas, decidimos implantar una serie de Servicios Web para acceder a información sobre las películas que vendemos.

De cada película ofreceremos información sobre su título, su director y su precio. Esta información podemos codificarla en una clase `PeliculaTO` como la siguiente:

```
public class PeliculaTO {
  private String titulo;
  private String director;
  private float precio;

  public PeliculaTO() {}

  public PeliculaTO(String titulo, String director, float precio) {
    this.titulo = titulo;
    this.director = director;
    this.precio = precio;
  }

  // Getters y setters
  ...
}
```

Vamos a permitir que se busquen películas proporcionando el nombre de su director. Por lo tanto, el servicio ofrecerá una operación como la siguiente:

```
List<PeliculaTO> buscaDirector(String director)
```

Proporcionaremos el nombre del director, y nos devolverá la lista de películas disponibles dirigidas por este director.

En un principio, podemos crear una lista estática de películas dentro del código de nuestro servicio, como por ejemplo:

```
final static PeliculaTO[] peliculas = {
    new PeliculaTO("Mulholland Drive", "David Lynch", 26.96f),
    new PeliculaTO("Carretera perdida", "David Lynch", 18.95f),
    new PeliculaTO("Twin Peaks", "David Lynch", 46.95f),
    new PeliculaTO("Telefono rojo", "Stanley Kubrick", 15.95f),
    new PeliculaTO("Barry Lyndon", "Stanley Kubrick", 24.95f),
    new PeliculaTO("La naranja mecánica", "Stanley Kubrick", 22.95f)
};
```

Se pide:

a) Implementar el servicio utilizando Netbeans. El servicio se llamará `TiendaDvdWS`, y estará dentro de un proyecto Maven con nombre `Sesion2-tienda-WS`.

Para construir una lista con las películas cuyo director coincida con el nombre del director que se ha solicitado, podemos utilizar un código similar al siguiente, donde se ha proporcionado un parámetro `director`:

```
director = director.toLowerCase();

ArrayList<PeliculaTO> list = new ArrayList<PeliculaTO>();

for (PeliculaTO pelicula : peliculas) {
  if (pelicula.getDirector().toLowerCase().indexOf(director) != -1) {
    list.add(pelicula);
  }
}

return list;
```

Una vez implementado el servicio, desplegarlo en Glassfish y probarlo mediante el

cliente de prueba generado. Observar los tipos de datos definidos en el documento WSDL generado y los mensajes SOAP para la invocación del servicio. (0,5 puntos)

b) Implementar un cliente java maven utilizando Netbeans. El cliente se llamará `Sesion2-tienda-WS-client`, y realizará una llamada al servicio mostrando, para cada película del director "kubrick", el título, el nombre del director y el precio. (0,5 puntos)