Ejercicios de creación de Servicios Web

Índice

1. Creación de un servicio web básico

Vamos a comenzar creando un servicio web básico. Este servicio web será un *Hola Mundo* en forma de servicio. Tendrá como nombre `HolaMundoSW` 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"`.

a) Implementar el servicio en un proyecto con nombre `servcweb-sesion02-hola`, y probarlo con *Web Services Explorer* para comprobar su correcto funcionamiento.

b) Implementar un cliente Java para este servicio en un proyecto con nombre `servcweb-sesion02-holacliente`, y probar que funciona correctamente.

2. Validación de NIFs (*)

Vamos a implementar un servicio con una serie de métodos que nos permitan validar un NIF. El servicio tendrá como nombre `ValidaDniSW`, y estará dentro de un proyecto `servcweb-sesion02-dni`. Se pide:

a) Implementar la siguiente operación:

```
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:

```
int obtenerLetra(String dni)
```

Esta función tomará como entrada un DNI, y como salida nos dirá el codigo ASCII de la letra que corresponde a dicho DNI. Para calcular la letra del DNI 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"
```

c) Implementar la siguiente operación:

```
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 Java que acceda a dicho servicio. El cliente deberá crearse en un proyecto `servcweb-sesion02-dnicliente`.

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 en Netbeans. El servicio se llamará `TiendaDvdSW`, y estará dentro de un proyecto con nombre `servcweb-sesion02-tienda`.

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.

b) Añadir al servicio anterior una segunda operación:

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

En este caso no obtendrá la lista de películas de nuestra base de datos local, sino que las buscará en Amazon mediante los servicios web que ofrece esta compañía.

Generar el cliente para acceder al servicio de Amazon desde nuestra aplicación. Se puede utilizar un código fuente como el visto en la sesión anterior para obtener el listado de películas, con la diferencia de que una vez obtenidos los datos, en lugar de imprimirlos en la consola los encapsularemos en una serie de objetos `PeliculaTO` que devolveremos como resultado.

c) Implementar un cliente Java que acceda a dicho servicio. El cliente deberá crearse en un proyecto `servcweb-sesion02-tiendacliente`.