SERVICIOS SOAP

Los servicios SOAP son una forma de implementar sistemas distribuidos empleando una arquitectura simple basada en la Web. Estos servicios se denominan *contract-first*, puesto que su desarrollo se basa en una especificación previa de las operaciones, los inputs y los outputs de estas.

Los servicios SOAP pueden ser implementados en cualquier lenguaje empleando las librerías adecuadas (PHP+Symfony2, C+gSOAP, Java+GlassFish…). En este caso, usaremos Apache CXF, un framework Java basado en la implementación de referencia (GlassFish) que facilita mucho el desarrollo, para crear 2 servicios distintos.

Servicio 1: Calculadora

Debéis desarrollar un servicio de calculadora que implemente las siguientes operaciones. Supondremos que todas las entradas serán número enteros.

1. Suma de dos números.
2. Resta de dos números
3. Multiplicación de dos números
4. División de dos números
5. Potencia de un número.
6. Raíz cuadrada de un número.
7. Logaritmo neperiano de un número.
8. Máximo de una lista de números.
9. Mínimo de una lista de números.
10. Media de una lista de números.
11. Mediana de una lista de números.
12. Moda de una lista de números.
13. Desviación típica de una lista de números

Servicio 2: Analizador de textos

Debéis desarrollar un servicio analizador de textos que implemente las siguientes operaciones:

1. Contar palabras.
2. Contar caracteres (incluyendo espacios, signos de puntuación, etc.)
3. Contar frases.
4. Número de veces que aparece una palabra.
5. Palabra más usada.
6. Palabra menos usada.
7. Reemplazar palabra.

Además, deberéis crear un cliente para esos servicios que permita, vía terminal, introducir los inputs y probar todos los servicios desarrollados.

Una de las mayores ventajas de Apache CXF es que elimina la necesidad de escribir la especificación de los servicios empleando XML, sustituyéndolo por anotaciones en el código Java.

Así, pasaríamos de tener que generar un fichero descriptor de los servicios (WSDL) a definir unas interfaces en Java a partir de las cuales este fichero será generado automáticamente:

```java
@WebService
public interface Calculator {
    int sum(int a, int b);
}
```

```java
@WebService(endpointInterface = "calculator.Calculator")
public class CalculatorImpl implements Calculator {
    @Override
    public int sum(int a, int b) {
        return a+b;
    }
}
```

Además, con Apache CXF podemos usar un servidor embebido en nuestra aplicación, de modo que no es necesario instalar ningún software adicional en nuestro equipo.

```java
public class Server {
    public static void main(String[] args) {
        Calculator c = new CalculatorImpl();
        String address = "http://localhost:8080/calculator";
        Endpoint.publish(address, c);
    }
}
```

Si además quisiésemos personalizar el nombre del servicio, de los métodos o de los parámetros, podemos usar el resto de anotaciones del paquete `javax.jws`

```java
@WebService
public interface Calculator {
    @WebMethod(operationName = "suma")
    @WebResult(name = "resultado")
    int sum(
       @WebParam(name = "sumando1") int a,
       @WebParam(name = "sumando2") int b);
}
```

Podéis consultar la especificación WSDL creada automáticamente por la librería para vuestros servicios añadiendo `?wsdl` al final de la URL donde despleguéis vuestros servicios.

Nota

Si usáis versiones de Java superiores a la 8, es probable que tengáis que añadir librerías extra como dependencia a vuestros proyectos. Esto es debido a que a partir de la versión 9 de Java se han eliminado algunas librerías que antes formaban parte de la distribución estándar. En este caso deberíais añadir `com.sun.xml.ws:jaxws-ri` y `com.sun.activation:jakarta.activation`.

Además de para crear servicios, Java también proporciona mecanismos para consumir estos. Para ello, podemos hacer uso de las clases presentes en el paquete `javax.xml.ws`

```
public class Client {
    public static void main(String[] args) throws
MalformedURLException {
        URL wsdlURL = new
URL("http://localhost:8080/calculator?wsdl");
        QName SERVICE_NAME = new
QName("http://calculator.soap.etse.usc.gal/", "Calculator");
        Service service = Service.create(wsdlURL, SERVICE_NAME);
        Calculator client = service.getPort(Calculator.class);
        int result = client.sum(1, 2);
        System.out.println(result);
    }
}
```

Además de este mecanismo, también podríais usar la herramienta wsdl2java para generar el código necesario para consumir los servicios en caso de que no tengáis acceso al código de las interfaces que definen el contrato que implementan los servicios.

Nota

Podéis encontrar el código completo del ejemplo usado en este enunciado en el siguiente repo de git: https://gitlab.citius.usc.es/docencia/soap

Evaluación

- Para obtener la máxima puntuación en esta práctica debéis implementar todos los servicios indicados.
- Todas las operaciones tendrán la misma puntuación.
- Una operación se considerará completamente implementada únicamente si se implementa el servicio y el cliente que la consuma y permita probarla.