educación IT
CODE YOUR FUTURE

Java Web API

Módulo 2

Tipos de Web Services

Tipos de Web Services

S — Simple
O — Object
A — Access
P — Protocol

SOAP es un protocolo estándar que define cómo **dos objetos en diferentes procesos pueden comunicarse** por medio de intercambio de datos **XML**.

R — REpresentational
E
S — State
T — Transfer

REST es un **servicio Web**, al estilo arquitectónico, en el que la atención está centrada en la **presencia de recursos en el sistema**. Cada recurso debe ser identificado por el identificador global de un **URI**. Para acceder a estos recursos, los clientes se comunican con el servicio REST por **HTTP**, y el servidor responde con la representación de los recursos.

REST vs. SOAP

- **REST utiliza HTTP**, en consecuencia es mucho más sencillo desarrollar APIs, crear clientes y la documentación es más fácil de entender.
- **REST permite muchos formatos de datos**. Esto le da al desarrollador la posibilidad de utilizar JSON en lugar de XML que es más rápido, permite también un mejor soporte a los clientes del explorador. **SOAP solamente permite XML**.
- **REST** tiene mejor **escalabilidad y rendimiento**.
- Las **lecturas del REST se pueden cachear**, las lecturas basadas en **SOAP no**.
- **SOAP resuelve más aspectos de seguridad**. Si bien tanto REST como SOAP soportan SSL, SOAP también soporta WS-Security lo que añade características de seguridad Enterprise.
- **SOAP proporciona una implementación estándar de integridad de datos** y privacidad de datos.
- **REST no soporta comunicación de fallos**.

Conclusión:

- Si el objetivo es **desarrollar algo práctico y sencillo**, **REST** debido a su simplicidad y efectividad es la solución.
- Pero si **la seguridad es prioritaria**, **SOAP** es el camino que se debe elegir.

SoapUI

SoapUI es una **aplicación de *testing* para *web services open-source***, utilizada en arquitecturas del tipo *Service-Oriented* (**SOA**) y *Representational State Transfer* (**REST**). Sus funcionalidades cubren inspección de *web services*, invocaciones, desarrollo, simulación y *mocking*, testeo funcional, testeos de carga y conformidad.

Es un *software free* que cuenta con una versión comercial, que agrega funcionalidades para mejorar la productividad. Está enteramente desarrollado en Java y utiliza Swing para la interfaz gráfica.

Gracias a esto existen versiones para todas las plataformas de sistemas operativos. Con SoapUI se pueden testear servicios web SOAP y REST, JMS, AMF, así como también realizar llamadas HTTP y JDBC.

La página oficial de SoapUI es: SoapUI.org. Allí está **disponible para descargar** la versión *free* para todos los sistemas operativos.

JAX-WS

***Java API for XML Web Services* (JAX-WS)** es una **especificación para la creación de servicios web basados en XML-SOAP**. Es la evolución de la especificación JAX-RPC, forma parte de JEE 5 y Java SE 6.

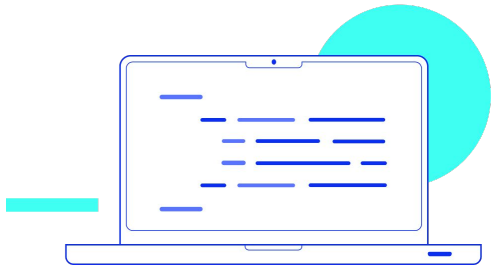
JAX-WS utiliza anotaciones, introducidas en Java SE 5 y permite la creación tanto de servicios como de clientes de servicios Web SOAP.

JAX-WS cuenta con una implementación de referencia llamada ***Metro***, aunque también existen otros productos que implementan la especificación como **CXF** o **AXIS**.

La **documentación oficial de Metro** se encuentra disponible en: jax-ws.java.net. En esa misma página, hay un link a la documentación oficial de la especificación: JCP.org.

WSDL

Web Services Description Language (WSDL) es un **lenguaje basado en XML utilizado para describir la funcionalidad de un servicio Web**. Permite comprender la interfaz del servicio Web al indicar cómo se debe llamar al servicio, qué parámetros espera, y qué estructuras de datos devuelve. Básicamente se trata del contrato entre quien produce el servicio web y quien lo consume.



Los elementos más importantes de un WSDL son los siguientes:

- **`types`:** define los tipos de datos que se intercambiarán en el mensaje. La definición de tipos puede verse como las definiciones Java de clase, con variables que pueden ser tipos primitivos o referencias a otras clases u objetos.
- **`message`:** define los distintos mensajes que se intercambiarán en la llamada al servicio web. Se definen los mensajes de entrada y salida para cada operación que ofrezca el servicio.
- **`portType`:** contiene una colección de una o más operaciones. Para cada una, indica cuáles son los mensajes de entrada y salida.
- **`binding`:** define el protocolo de red y el formato de los datos para las operaciones de un `portType`. Los *bindings* son definiciones concretas de los `portTypes`. Un `portType` puede tener múltiples *bindings* asociados.

JAXB

Java Architecture for XML Binding (JAXB) permite a los desarrolladores Java **asignar clases de Java a representaciones XML**.

JAXB proporciona dos características principales: la capacidad de serializar las referencias de objetos Java a XML y la inversa, es decir, deserializar XML en objetos Java. Estas operaciones son conocidas como ***marshalling* (Java a XML)** y ***unmarshalling* (XML a Java)**. Esta conversión es utilizada por los servicios Web cuando en los mensajes son enviados objetos en lugar de datos nativos.

Anotaciones

Es posible especificar la forma en la que se crea el servicio mediante diferentes anotaciones. Las principales son:

@WebService	Indica la clase que define un servicio web. Se pueden especificar como parámetros los nombres del servicio (`serviceName`), del componente `Port` (`portName`), del SEI del servicio (`name`), de su espacio de nombres (`targetNamespace`), y de la ubicación del WSDL (`wsdlLocation`), que figurarán en el documento WSDL del servicio:
	
	`@WebService(name="ConversionPortType",`
	`serviceName="ConversionService",`
	`portName="ConversionPort",`
	`targetNamespace="http://jtech.ua.es",`
	`wsdlLocation="resources/wsdl/")`
	
	Cuando se consume un WS, define de forma abstracta un servicio web, las operaciones que puede llevar a cabo y los mensajes involucrados en cada operación.

@WebMethod	Indica que un determinado método debe ser publicado como operación del servicio. Si no se indica para ningún método, se considerará que deben ser publicados todos los métodos públicos. Sino solo se publicarán los métodos indicados. Además, de forma opcional se puede indicar como parámetro el nombre con el que se desea que aparezca la operación en el documento WSDL:
	
	`@WebMethod(operationName="eurosAptas")`
	`public int euro2ptas(double euros) {`
	`  ...`
	`}`

@Oneway	Indica que la llamada a la operación no debe esperar ninguna respuesta. Esto sólo lo podremos hacer con métodos que devuelvan `void`. Por ejemplo:
	
	`@Oneway()`
	`@WebMethod()`
	`public void publicarMensaje(String mensaje) {`
	`  ...`
	`}`

@WebParam	Permite indicar el nombre que recibirán los parámetros en el fichero WSDL:
	
	`@WebMethod(operationName="eurosAptas")`
	`public int euro2ptas(`
	`  @WebParam(name="CantidadEuros",`
	`  targetNamespace="http://jtech.ua.es")`
	`  double euros) {`
	`  ...`
	`}`

@WebResult	Permite indicar el nombre que recibirá el mensaje de respuesta en el fichero WSDL:
	
	`@WebMethod(operationName="eurosAptas")`
	`@WebResult(name="ResultadoPtas",`
	`  targetNamespace="http://jtech.ua.es")`
	`public int euro2ptas(double euros) {`
	`  ...`
	`}`

@WebFault	Se utiliza para anotar excepciones Java. Cuando se usa esta anotación en una excepción se está indicando que cuando sea lanzada por una operación del servicio web debe generar un mensaje SOAP de respuesta con un *SOAP Fault* que indique el error producido. En el lado del cliente la clase con dicha excepción se habrá generado en el stub para el acceso al servicio, y al recibir el mensaje SOAP con el error el `stub` lanzará la excepción correspondiente. Es decir, para el desarrollador será como si la excepción saltase directamente desde el servicio hasta el cliente.
	
	`@WebFault`
	`public class ConversionFaultException extends Exception {`
	`  public ConversionFaultException(String msg) {`
	`  super(msg);`
	`  }`
	`}`

Fuente: Seguridad en aplicaciones Web Java - José Manuel Ortega Candel.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!