

Capítulo 3

Java API for XML-based Web Services

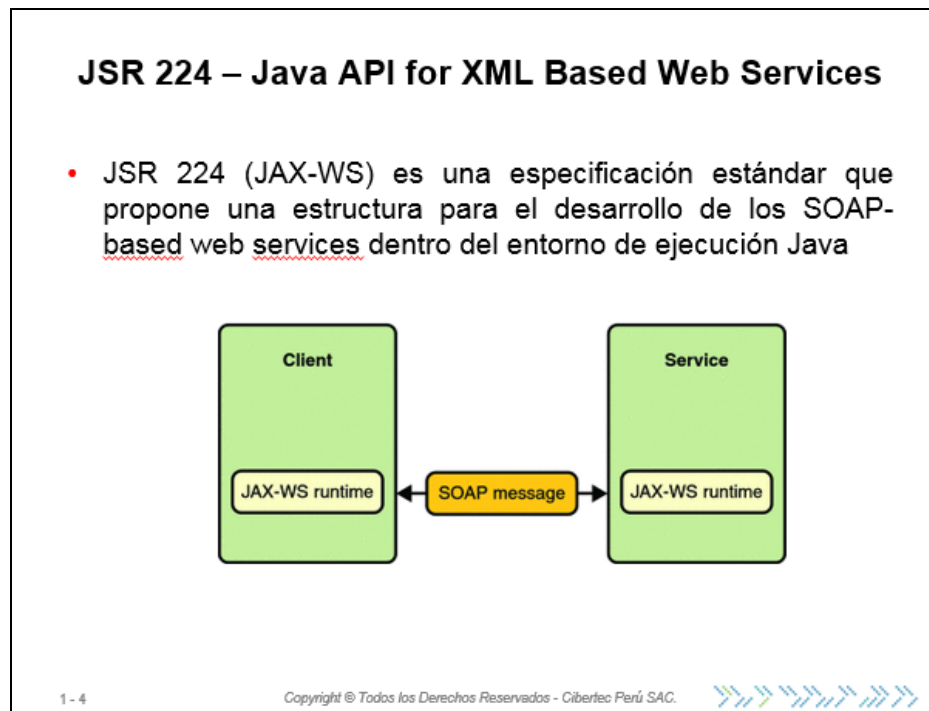
Al finalizar el capítulo, el alumno podrá

- Conocer las principales características de la especificación JAX-WS para Web Services.
- Implementar aplicaciones que utilicen JAX-WS

Temas

1. JSR 224—Java API for XML-Based Web Services
2. WSDL Document SOAP Structure and Binding
3. WS-Security JAX-WS Metro Reference Implementation

1. JSR 224 – Java API for XML-Based Web Services



1.1 Java API for XML-based web services


JAX-WS posee las siguientes características:

- Proporciona un API para desarrollo de SOAP-based web services tanto del lado servidor como del lado cliente.
- Es independiente de plataforma. Esta característica es heredada en sí de la plataforma Java que indica que los servicios desarrollados en una plataforma pueden ser fácilmente transportados a otra.
- Utiliza anotaciones para definir sus componentes y servicios, lo cual provee simplicidad en el desarrollo y escalabilidad en su implementación.
- Se basa en el concepto de POJO (Plain Old Java Object) para definir sus services e interfaces.
- Provee soporte a invocación asíncrona de servicios.

2. WSDL Document

WSDL Document

- Code First vs Contract First
- Es correcto primero definir un servicio como clases Java y luego autogenerar el WSDL ?.
- Se debe diseñar el WSDL manualmente y luego, generar las implementaciones del servicio en clases java ?.
- JAX-WS recomienda utilizar "Code First" para una simple generación del WSDL. Sin embargo, también puede soportar la personalización de algunos detalles del WSDL o incluso la generación de un servicio a partir de un documento WSDL, pero con algunas limitaciones.

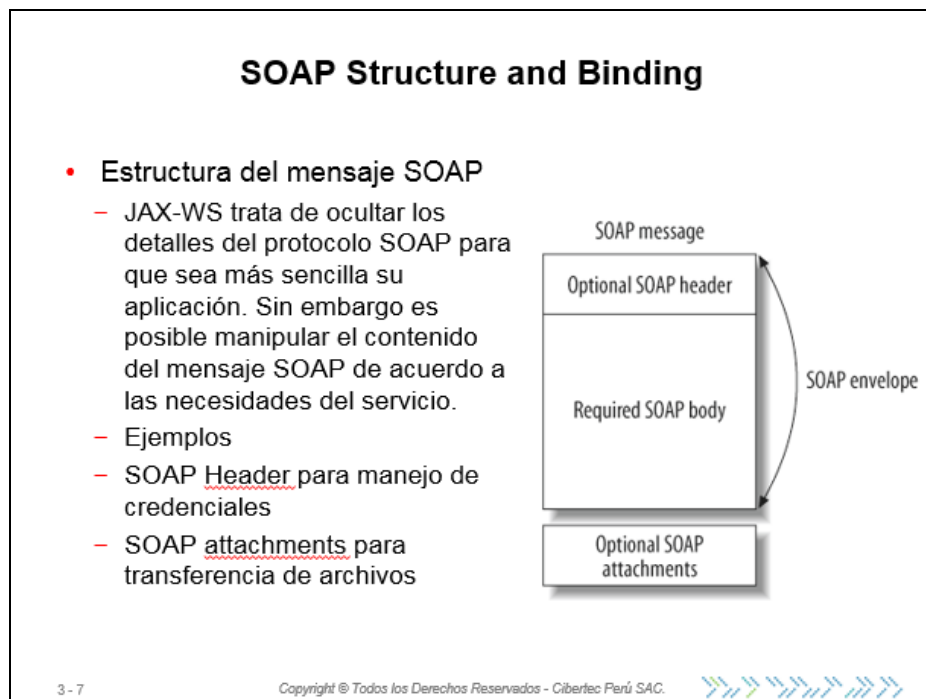
1 - 6Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.

2.1 WSDL (Web Service Description Language)

Es el contrato mediante el cual se especifican los detalles de transporte entre el cliente y el servidor para mensajes y su contenido.

La especificación JAX-WS recomienda utilizar sus herramientas para generar automáticamente el documento WSDL a partir de clases Java que definan un servicio. Sin embargo, también puede soportar la generación de un servicio a partir de un documento WSDL previamente definido.

3. WS-Security JAX-WS Metro Reference Implementation



3.1 Estructura del mensaje SOAP

La especificación JAX-WS trata de ocultar los detalles del protocolo SOAP (Simple Access Object Protocol) para una implementación más sencilla. Sin embargo, es posible manipular el contenido del mensaje SOAP de acuerdo a las necesidades del servicio.

3.2 SOAP Binding

La especificación JAX-WS se basa en la especificación JAXB (Java Architecture for XML Binding) para el “*binding*” o enlace entre las definiciones de entidades XML y objetos Java.

Las posibles combinaciones para los estilos de “*binding*” soportadas por JAX-WS son las siguientes:

- RPC/encoded
- RPC/literal
- Document/encoded
- Document/literal

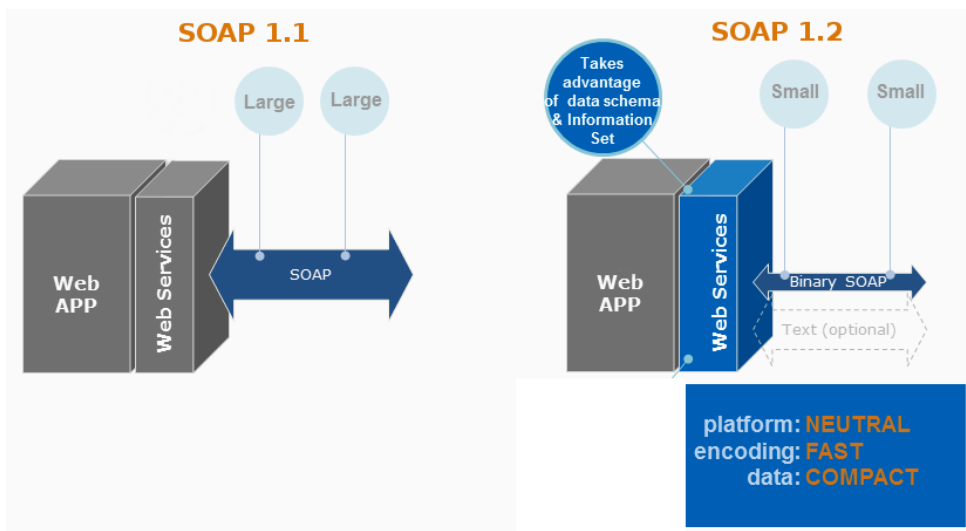
Los servicios desarrollados bajo los estilos “*encoded*” no han sido aprobados por el estándar WS-I (Web Services-Interoperability); por lo cual, no se recomienda su uso ya que la iniciativa de la WS-I es ayudar a que los Web Services sean interoperables sin importar las diferencias de plataforma ni lenguaje de programación.

El estilo “RPC/literal” es heredado de la especificación JAX-RPC que fue la predecesora de JAX-WS la cual contaba con su propio esquema de transformación de XML a objetos Java, pero poseía algunas limitaciones para el soporte a tipos complejos.

El estilo soportado por defecto por JAX-WS es “Document/literal” ya que es el más extensible en cuanto a soporte a tipos XML. Adicionalmente, se basa en JAXB (Java Architecture for XML Binding) para la transformación de XML a objetos Java, por lo cual puede manejar esquemas XSD complejos en el procesamiento de mensajes XML.

3.3 Diferencias entre SOAP 1.1 y SOAP 1.2

La organización World Wide Web Consortium (W3C) maneja dos versiones del protocolo SOAP 1.1 y 1.2. Si bien las diferencias entre ambas versiones son considerables, el impacto en el modelo de programación de JAX-WS es menor, ya que la especificación mantiene este nivel del protocolo oculto para su implementación.



A continuación se detallan las diferencias más resaltantes entre el protocolo SOAP 1.1 y 1.2:

SOAP 1.1	SOAP 1.2
<ul style="list-style-type: none"> La serialización de mensajes está basada únicamente en XML. 	<ul style="list-style-type: none"> La serialización de mensajes está basada en XML Information Set (XML InfoSet) lo cual le permite utilizar otras formas de serialización aparte de XML, como por ejemplo protocolos binarios más compactos.
<ul style="list-style-type: none"> El protocolo de transporte utilizado es HTTP. 	<ul style="list-style-type: none"> El protocolo de transporte es completamente configurable. Los mensajes pueden ser transportados utilizando HTTP, SMTP entre otros.
<ul style="list-style-type: none"> Permite múltiples elementos en el <body> del mensaje SOAP. 	<ul style="list-style-type: none"> Permite un único elemento en el <body> del mensaje SOAP.

<ul style="list-style-type: none"> • Maneja una estructura simple para mensajes de error <Fault>. 	<ul style="list-style-type: none"> • Maneja una estructura compleja y detallada para el manejo de errores <Fault>.
<ul style="list-style-type: none"> • Está compuesta por un documento único. 	<ul style="list-style-type: none"> • Está organizado en 3 partes. <ul style="list-style-type: none"> – Parte 0: Introducción no normativa a SOAP – Parte 1: Descripción del mensaje SOAP y binding framework. – Parte 2: Descripción opcional de add-ins para la codificación del mensaje.

WS-Security

WS Security es una familia de especificaciones orientadas a proveer seguridad en la implementación de Web Services.

Existen tres contextos orientados a la protección e intercambio de mensajes SOAP y la familia de especificaciones de WS-Security brinda soporte a todos ellos. Los contextos son los siguientes:

- El primer contexto define la forma de identificar y autenticar los clientes que invocan un servicio. WS-Security proporciona una forma estándar de transferencia de tokens de seguridad dentro del mensaje SOAP.
- El segundo contexto define la forma de garantizar la integridad de los mensajes SOAP. WS-Security utiliza un mecanismo de firmas XML para proteger los mensajes.
- El tercer contexto define la forma de garantizar que los mensajes no sean interceptados mientras están en tránsito. WS-Security se apoya en algoritmos de cifrado para este propósito.

A continuación se detalla la arquitectura modular de la familia de especificaciones de WS-Security:

WS-Policy

Esta especificación describe en general las características, restricciones y políticas de seguridad del servicio. Por ejemplo, una WS-Policy puede indicar que la transferencia de mensajes utilizará tokens de seguridad o un algoritmo particular de encriptación.

WS-Trust

Esta especificación está regulada por la organización OASIS (www.oasis-open.org), la cual proporciona normativas para garantizar las políticas de seguridad en los servicios. Por ejemplo, la forma en que los tokens de seguridad serán emitidos, renovados y validados.

WS-Privacy

Esta especificación define como se aplicarán las políticas de seguridad. Así como también, provee una forma de determinar si el invocador de servicios pretende seguir esas políticas de seguridad.

WS-SecureConversation

Esta especificación se orienta a la comunicación segura entre los diferentes servicios incluyendo sus contextos de seguridad y dominios. Por ejemplo, se enfoca en como las llaves de seguridad serán emitidas e intercambiadas entre los servicios.

WS-Federation

Esta especificación se enfoca en el manejo de las identidades de los servicios entre las diferentes plataformas y organizaciones.

WS-Authorization

Esta especificación se enfoca en el manejo de información para la autorización como por ejemplo garantizar que el acceso sea restringido a los tokens de seguridad y a otros recursos protegidos.

JAX-WS Metro Reference Implementation

Metro es una implementación de referencia de la especificación JAX-WS desarrollada por Oracle - Sun.

Forma parte de la comunidad GlassFish y puede ser utilizada de manera integrada en su servidor de aplicaciones o en modo “*stand-alone*” integrada en otros servidores de aplicaciones.

Adicionalmente, posee características de alto rendimiento, extensibilidad y simplicidad de uso en su implementación y está preparada para ambientes de producción de alta exigencia.

Los componentes básicos para desarrollar un web service con la especificación JAX-WS se muestran a continuación:

- Service Endpoint Interface (SEI)
- Service Implementation Bean (SIB)
- WSDL Document
- Endpoint Publisher
- Java Requester for Web Service (opcional)

Service Endpoint Interface (SEI)

```
@WebService
@SOAPBinding(style = Style.DOCUMENT,
              use = Use.LITERAL)
public interface TimeService {

    @WebMethod
    public String getTimeAsString();

    @WebMethod
    public long getTimeAsElapsed();
}
```

El Service Endpoint Interface (SEI) es una clase Java en la cual se definen las operaciones y parámetros que serán implementadas en el servicio.

Service Implementation Bean (SIB)

```
@WebService(endpointInterface = "jaxws.TimeService")

public class TimeServiceImpl implements TimeService{

    public String getTimeAsString() {
        return new Date().toString();
    }

    public long getTimeAsElapsed() {
        return new Date().getTime();
    }
}
```

El Service Implementation Bean (SIB) es una clase Java en la cual se implementarán los métodos heredados de la interfaz del servicio (SEI).

Endpoint Publisher

```
public class TimeServicePublisher {

    public static void main(String[] args) {

        Endpoint.publish("http://localhost:9876/timeService",
            new TimeServiceImpl());

        System.out.println("Publish service");

    }
}
```


El Endpoint Publisher es la forma como se publicará el servicio una vez que esta implementado. Java cuenta con una forma ligera de publicación para ambientes de desarrollo a través de su clase Endpoint. Sin embargo, en ambientes de producción, se deberá utilizar un Servidor de Aplicaciones JEE como Glassfish, JBoss entre otros.

WSDL Document

```
<definitions
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  ...
  targetNamespace="http://jaxws/"
  name="TimeServiceImplService">
  ...
  <portType name="TimeService">
    <operation name="getTimeAsString">
      <input message="tns:getTimeAsString"></input>
      <output message="tns:getTimeAsStringResponse"></output>
    </operation>
  </portType>
  ....
  <service name="TimeServiceImplService">
    <port name="TimeServiceImplPort" binding="tns:TimeServiceImplPortBinding">
      <soap:address location="http://localhost:9876/timeService"></soap:address>
    </port>
  </service>
</definitions>
```

El documento WSDL es generado a partir de la interfaz del servicio (SEI) mediante de las herramientas que proporciona la plataforma Java. Este documento será utilizado por las aplicaciones cliente para invocación del servicio.

Java Requester for Web Service

```
public class TimeServiceClient {
    public static void main(String[] args) throws MalformedURLException {

        // Reference the URL
        URL url = new URL("http://localhost:9876/timeService?wsdl");

        //Qualified name of the service
        QName qName = new QName("http://jaxws/", "TimeServiceImplService");

        //Create a factory for the service
        Service service = Service.create(url, qName);

        //Get SEI or service port
        TimeService servicePort = service.getPort(TimeService.class);

        System.out.println("TimeAsString : " + servicePort.getTimeAsString());
        System.out.println("TimeAsElapsed : " + servicePort.getTimeAsElapsed());
    }
}
```

El Java Requester o cliente Web Service es la forma como se invocaría al servicio utilizando la plataforma Java. Sin embargo, debido a que los Web Services están basados en estándares de interoperabilidad, es posible crear aplicaciones cliente en otros lenguajes de programación tales como Pearl, Ruby, .Net entre otros.