



FORMACIÓN EN NUEVAS TECNOLOGÍAS

CURSO

DESARROLLO DE SERVICIOS WEB SOAP CON JAX-WS

**VIEWNEXT**  
AN IBM SUBSIDIARY

# ICONO TRAINING



Formación en Nuevas Tecnologías  
● [www.iconotc.com](http://www.iconotc.com)



[linkedin.com/company/icono-training-consulting](https://linkedin.com/company/icono-training-consulting)



[training@iconotc.com](mailto:training@iconotc.com)

# FORMADOR



● **Jose Mª Díaz Charcán.**  
Consultor / formador en tecnologías  
del área de programación y  
desarrollo.

# DESARROLLO DE SERVICIOS WEB SOAP CON JAX-WS

- **DURACIÓN**

- ↳ 20 horas

- ↳ **LUGAR/FECHAS /HORARIO:**

- ↳ Málaga. Del 5 al 8 de marzo de 2018. De 15:00 hs a 20:00 hs

- ↳ **CONTENIDO:**

- ↳ Visión general de SOA y los servicios web
  - ↳ Conceptos básicos: SOAP / WSDL
  - ↳ XML y XML Schema
  - ↳ JAXB, Introducción a JAX-WS, Apache CXF
  - ↳ Diseño e implementación de servicios web SOAP: Contract First y Code First
  - ↳ Cliente SOAP
  - ↳ SOAP Faults
  - ↳ JAX-WS Handlers
  - ↳ MTOM, Cliente Asíncrono

# Arquitecturas orientadas a Servicios

## ¿Qué es SOA?

- Para el director de informática, una estrategia que reduce el coste del ciclo de vida de un sistema y maximiza el retorno de la inversión
- Para el equipo de negocio, un conjunto de servicios a los que sus clientes pueden acceder para interaccionar con dicho negocio
- Para el equipo de arquitectura, una solución que permite integrar y reutilizar sistemas diversos con un acoplamiento mínimo
- Para el desarrollador, un paradigma o modelo de programación donde servicios web y los contratos entre ellos se convierten en lo más importante para diseñar sistemas interoperables, flexibles y adaptables

# Nociones fundamentales SOA

El elemento esencial es el servicio, con las siguientes características:

- Sin estado
- Localizable
- Completamente descrito a través de una interfaz expuesta a clientes
- Debe poder estar compuesto de, y combinarse con, otros servicios
- Acoplamiento bajo
- Dirigido por políticas a nivel de servicio
- Deslocalizable y completamente independiente de la plataforma y de las herramientas empleadas para crearlo
- Suministrar información suficiente (*coarse-grained*, no *fine-grained*) : getAccount() y no getAccountNumber() + getAccountOwner()
- Asincronía: no obligatorio, pero aconsejable

Existen [diferencias conceptuales](#) entre un servicio SOA y un servicio Web

# Servicios Web

- ¿Qué es un servicio Web?
- Tipos de servicios Web
  - Orientados a XML
  - Orientados a HTTP
- ¿Cuál usamos?

# Tecnologías para servicios Web XML

- SOAP
- WSDL
- UDDI (Universal Description, Discovery and Integration)
- WS-SECURITY y este enlace

# Lenguaje XML (eXtensible Markup Language)

- XML es un mecanismo basado en etiquetas arbitrariamente elegidas para representar datos y sus relaciones, desde una perspectiva jerárquica
- Un documento XML debe estar bien formado:
  - Ha de tener un prólogo (instrucción de procesamiento)
  - Un único elemento raíz
  - Todas las etiquetas deben cerrarse
  - Mayúsculas y minúsculas son diferentes
  - Los elementos deben estar anidados correctamente
  - Los valores de los atributos deben inscribirse entre dobles comillas
- Un documento XML debe ser válido. Las reglas que definen la validez de un documento se suelen expresar mediante DTDs (Document Type Definition) o XSD (XML Schema Definition)

# Lenguaje XML Schema (XML Schema Definition)

- Un documento XML Schema (a partir de ahora XSD) describe la estructura, el vocabulario permitido y sus reglas sintácticas y semánticas de un documento XML
- El elemento esquema (`<schema>`) es el nodo raíz de un documento XSD
- La etiqueta elemento (`<element>`) define un contenido que sólamente tiene texto, sin otros elementos subordinados
- Los elementos pueden ser complejos. Para definirlos se usa la etiqueta `<complexType>`
- A menudo, para definir completamente un elemento complejo, debe añadirse la información adicional de que constituye una secuencia de otros elementos, simples o complejos. La etiqueta al efecto es `<sequence>`
- Los elementos pueden tener atributos y restricciones (por ejemplo, un dato numérico podría tener que estar obligatoriamente dentro de un rango)
- Los datos pueden tener tipo (por ejemplo `<string>`) y valores por defecto

# Protocolo SOAP (Simple Object Access Protocol)

- SOAP es una definición precisa de cómo dos objetos en procesos potencialmente diferentes intercambian información
- SOAP usa XML para permitir y universalizar el intercambio
- Así pues, hemos de hablar de intercambio de mensajes
- La estructura de un mensaje SOAP es la siguiente:
  - Un sobre ([envelope](#)) obligatorio
  - Una o varias cabeceras ([header](#)) opcionales, que aportan metadatos sobre el intercambio de información, compuestos a su vez de bloques ([header block](#))
  - El cuerpo ([body](#)) del mensaje, obligatorio
  - Información sobre posibles errores ([Fault](#)) que se hayan ido produciendo en el procesamiento del mensaje, desde el origen al destino

# Lenguaje WSDL (Web Services Description Language)

- Un documento WSDL describe las características esenciales de un servicio
- La estructura fundamental es la siguiente:
  - Definiciones (`<definitions>`), nodo raíz del documento
  - Tipos (`<types>`) define la estructura de los datos y los tipos de los mismos empleados por el servicio, normalmente a través de un [archivo de esquema](#)
  - Mensaje (`<messages>`), los datos que se mueven en el servicio
  - Tipo de “puerto” (`<portType>`), conjunto de operaciones que soporta el servicio junto con los mensajes involucrados
  - Enlace (`<binding>`), protocolo y formatos soportados por cada tipo de puerto
  - Puerto (`<port>`), define un punto final (end point) especificando una única dirección para un enlace, para poder acceder al servicio
  - Servicio(`<service>`), define los puertos soportados

# JAXB (Java Architecture for XML Binding)

- Dicho en palabras simples, JAXB es un mecanismo estándar de Java para serializar objetos en XML y viceversa
- Para ello, creamos programas mediante el [API](#) de JAXB
- Definimos los objetos que deben ser serializados en XML y viceversa empleando [anotaciones](#)

# JAX-WS (Java Api for XML Web Services)

- La idea principal de esta herramienta es implementar todos los protocolos estándar de los servicios Web SOAP bajo la plataforma Java
- Aunque hay varias maneras de crear servicios web y sistemas cliente, Java ofrece un mecanismo que nos abstrae por completo de SOAP, WSDL, etc., de manera que los desarrolladores piensan y trabajan exclusivamente desde Java
- JAX-WS se apoya en JAXB de forma totalmente transparente para nosotros (si queremos)
- Información [adicional 1](#)
- Información [adicional 2](#)

# Apache CXF (Basado en los proyectos [Celtix](#) y [XFire](#))

- Herramienta clásica (junto con [Apache Axis2](#)) para el desarrollo de servicios
- [CXF soporta servicios](#) web SOAP y RESTful, apoyándose en las APIs estándar de Java al efecto: [JAX-WS](#) y [JAX-RS](#)
- La información que los servicios intercambian entre sí puede involucrar protocolos muy diferentes, SOAP, HTTP RESTful, etc.
- A su vez, la información estructurada según dichos protocolos puede moverse usando varios mecanismos de transporte, tales como [HTTP](#), [JMS](#) o [JBI](#)
- Esta herramienta se apoya en [Spring Framework](#)
- CXF [recomienda](#) el uso de [Maven](#) para los desarrolladores, a pesar de la existencia de plugins para herramientas visuales como [Eclipse](#)

# Diseño Contract-First

- La idea es: escribir primero el wsdl del servicio y a partir de ahí generar y/o escribir el código Java necesario
- Los pasos a seguir podrían ser:
  - Crear el archivo de esquema (XSD) para especificar los datos que va a intercambiar el servicio y los tipos request y response
  - Escribir el wsdl basándonos en el archivo de esquema
  - Generar el código Java de respaldo empleando JAX-WS a través de Maven
  - Escribir una implementación de la interfaz de servicio
  - Definir los “end points” con JAX-WS ([wsimport](#)) ó CXF ([wsdltojava](#))
- Para probar el correcto funcionamiento del sistema puede usarse [SoapUI](#), por ejemplo. O escribir/generar un cliente y usarlo

# Diseño Code-First

- En este caso, la idea es justamente la contraria
- Escribimos el código Java del servicio Web y definimos sus características usando anotaciones estándar de la plataforma empresarial de Java ([JEE](#), Java Enterprise Edition)
- Configuramos Maven para que el WSDL se genere automáticamente al, por ejemplo, desplegar el proyecto en un servidor de aplicaciones
- Probamos el servicio
- Escribimos una aplicación cliente y generamos automáticamente el código Java necesario para invocar a los métodos del servicio

# Cliente SOAP

- El objetivo de un sistema cliente es intercambiar datos con un servicio Web remoto llamando a sus métodos
- Dado que una de las características fundamentales de los servicios Web es la interoperabilidad, podemos escribir el cliente en el lenguaje que queramos
- Por ejemplo, nuestro cliente podría usar Java, JavaScript, etc..

# SOAP Fault

- Estrategia para el manejo de [errores en servicios Web](#)
- Al invocar a un servicio Web, pueden ocurrir errores de varios tipos
- Los más significativos para los sistemas cliente son aquellos que se dan en el servidor
- Mediante las “SOAP Faults” la información asociada al error aparecerá en la response del servidor, así como su definición en el wsdl
- Posible secuencia de uso:
  - creamos un servicio Web con anotaciones JAX-WS estándar con o sin SEI ([Service Endpoint Interface](#))
  - escribimos una excepción Java normal anotada con [@WebFault](#)
  - Creamos un “Fault Bean”, un POJO estándar y lo usamos como atributo de la excepción
  - En los métodos del servicio Web, lanzamos nuestra(s) excepción cuando convenga

# JAX-WS Handlers

- A veces es necesario procesar los mensajes SOAP de entrada y salida tanto a nivel de cliente como de servidor
- El motivo normalmente es manipular y/o detectar metainformación que ayuda a clientes y servidores a mejorar el intercambio de información
- La respuesta de JAX-WS es la figura de un manipulador ([handler](#)) de mensajes SOAP
- Los handlers pueden constituir una cadena, es decir, puede haber varios que JAX-WS invoca automáticamente en secuencia
- El método para usar esta estrategia podría ser:
  - Crear un archivo de configuración, a menudo llamado [handler-chain.xml](#)
  - Escribir una clase que implemente la interfaz [SOAPHandler](#)
  - Manipular los mensajes de la manera deseada y devolver true para permitir que la cadena siga ejecutándose, false si queremos detenerla

# MTOM (Message Transmission Optimization Mechanism )

- El objetivo de esta tecnología es optimizar la velocidad de transferencia y acceso a información incluída en un mensaje SOAP como adjunto
- MTOM usa [XOP](#) (XML-binary Optimizing Packaging) para llevar esto a cabo
- La activación de este protocolo debe llevarse a cabo en servidor y en cliente
  - En servidor, pueden emplearse anotaciones estándar de la JEE
  - En cliente, es posible instalar una “característica MTOM” ([MTOMFeature](#))
  - Uno de los valores importantes aquí es el [umbral](#) ([threshold](#))
- También es posible seguir un camino diferente, especialmente en CXF, empleando [DataHandlers](#)
- Si los adjuntos pesan mucho, existe la opción en [JAX-WS](#) RI ([Reference Implementation](#)) de emplear [Streaming](#)

# Cliente Asíncrono

- En primer lugar, es conveniente comentar que el propio servicio Web puede ser explícitamente asíncrono
- En cuanto al sistema cliente, una vez ejecutada la llamada asíncrona, tenemos dos estrategias principales para acceder a la respuesta:
- Preguntar periódicamente si los resultados han llegado (polling). En java, esto suele implicar el uso de futuros
- Instalar una retrollamada (callback) que se ejecutará cuando la información solicitada al servidor esté disponible
- Es posible llevar todo esto a cabo empleando diferentes artefactos, pero tal vez lo más general sea aprovechar los `CompletableFuture<T>` disponibles a partir de JSE 8.0

# Patrones de diseño

- Referencia

- Asynchronous Interaction
- JMS Bridge
- Web Service Cache
- Web Service Broker

# APIs para servicios Web en Java

- Orientados a XML: JAX-WS
  - SAAJ (Soap with Attachments for Java)
- Orientados a HTTP
- JAX-RS especificación
- JAX-RS implementación de referencia

# Ejemplos básicos

- Creación de un servicio Web con JAX-WS
  - Creación de un cliente para este servicio
- Creación de un servicio Web con JAX-RS
  - Creación de un cliente para este servicio