

Arquitectura Orientada a Servicios

Servicios Web

Edwin Salvador

05 de junio de 2015

Sesión 9

1 Servicios Web

- Introducción
- Tipos
- Arquitectura de los Servicios Web
- Servicios web SOAP y JavaEE
- Tecnologías básicas

1 Servicios Web

- Introducción
- Tipos
- Arquitectura de los Servicios Web
- Servicios web SOAP y JavaEE
- Tecnologías básicas

¿Qué es?

- Los Servicios Web (WS) proporcionan una forma estándar de **interoperar** entre aplicaciones de software que se ejecutan en diferentes plataformas.

¿Qué es?

- Los Servicios Web (WS) proporcionan una forma estándar de **interoperar** entre aplicaciones de software que se ejecutan en diferentes plataformas.
- Escritos en **XML** lo que los hace **independientes de la plataforma** y permite la integración de aplicaciones escritas en **cualquier lenguaje de programación**.

¿Qué es?

- Los Servicios Web (WS) proporcionan una forma estándar de **interoperar** entre aplicaciones de software que se ejecutan en diferentes plataformas.
- Escritos en **XML** lo que los hace **independientes de la plataforma** y permite la integración de aplicaciones escritas en **cualquier lenguaje de programación**.
- Se puede ver como una **web para máquinas**. ¿Qué quiere decir esto? Podemos aclarar esto si entendemos como funciona la web para humanos.

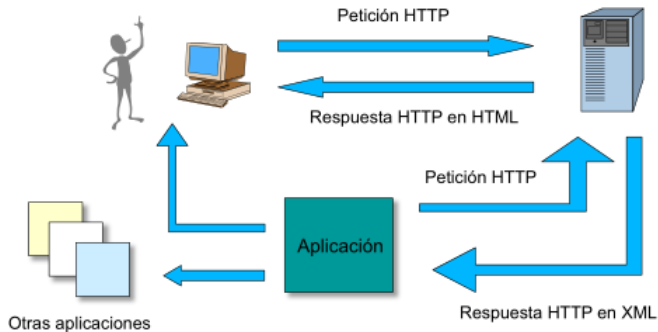
¿Qué es?

- Los Servicios Web (WS) proporcionan una forma estándar de **interoperar** entre aplicaciones de software que se ejecutan en diferentes plataformas.
- Escritos en **XML** lo que los hace **independientes de la plataforma** y permite la integración de aplicaciones escritas en **cualquier lenguaje de programación**.
- Se puede ver como una **web para máquinas**. ¿Qué quiere decir esto? Podemos aclarar esto si entendemos como funciona la web para humanos.
- Los WS son componentes de aplicaciones distribuidas.

¿Qué es?

- Los Servicios Web (WS) proporcionan una forma estándar de **interoperar** entre aplicaciones de software que se ejecutan en diferentes plataformas.
- Escritos en **XML** lo que los hace **independientes de la plataforma** y permite la integración de aplicaciones escritas en **cualquier lenguaje de programación**.
- Se puede ver como una **web para máquinas**. ¿Qué quiere decir esto? Podemos aclarar esto si entendemos como funciona la web para humanos.
- Los WS son componentes de aplicaciones distribuidas.
- Varios servicios simples pueden interoperar para proporcionar servicios más complejos.

Web para humanos vs web para máquinas



- **Accesible a través de la Web.** Protocolos HTTP y lenguaje estándar.

- **Accesible a través de la Web.** Protocolos HTTP y lenguaje estándar.
- **Debe describirse a sí mismo.** Facilita el uso por parte de otras aplicaciones de manera automática.

- **Accesible a través de la Web.** Protocolos HTTP y lenguaje estándar.
- **Debe describirse a sí mismo.** Facilita el uso por parte de otras aplicaciones de manera automática.
- **Fácil de localizar.** Las aplicaciones deben ser capaces de encontrar un servicio web de manera automática.

- **Accesible a través de la Web.** Protocolos HTTP y lenguaje estándar.
- **Debe describirse a sí mismo.** Facilita el uso por parte de otras aplicaciones de manera automática.
- **Fácil de localizar.** Las aplicaciones deben ser capaces de encontrar un servicio web de manera automática.
- La interoperabilidad y la extensibilidad es la clave.

1 Servicios Web

- Introducción
- Tipos
 - Arquitectura de los Servicios Web
 - Servicios web SOAP y JavaEE
 - Tecnologías básicas

- A nivel **conceptual**, un servicio es un componente de software que se proporciona a través de un *endpoint*.

- A nivel **conceptual**, un servicio es un componente de software que se proporciona a través de un *endpoint*.
 - Un *endpoint* se especifica mediante una URI.

- A nivel **conceptual**, un servicio es un componente de software que se proporciona a través de un *endpoint*.
 - Un *endpoint* se especifica mediante una URI.
 - Servicio productores y consumidores intercambian mensajes de invocación y respuesta.

- A nivel **conceptual**, un servicio es un componente de software que se proporciona a través de un *endpoint*.
 - Un *endpoint* se especifica mediante una URI.
 - Servicio productores y consumidores intercambian mensajes de invocación y respuesta.
- A nivel **técnico** tenemos dos tipos de servicios web según la forma de implementación:

- A nivel **conceptual**, un servicio es un componente de software que se proporciona a través de un *endpoint*.
 - Un *endpoint* se especifica mediante una URI.
 - Servicio productores y consumidores intercambian mensajes de invocación y respuesta.
- A nivel **técnico** tenemos dos tipos de servicios web según la forma de implementación:
 - Servicios Web SOAP

- A nivel **conceptual**, un servicio es un componente de software que se proporciona a través de un *endpoint*.
 - Un *endpoint* se especifica mediante una URI.
 - Servicio productores y consumidores intercambian mensajes de invocación y respuesta.
- A nivel **técnico** tenemos dos tipos de servicios web según la forma de implementación:
 - Servicios Web SOAP
 - Servicios Web RESTful

Servicios Web SOAP

- Adecuados para servicios web grandes.

Servicios Web SOAP

- Adecuados para servicios web grandes.
- Utilizan el estándar SOAP (*Simple Object Access Protocol*) para definir la arquitectura y formato de sus mensajes.

Servicios Web SOAP

- Adecuados para servicios web grandes.
- Utilizan el estándar SOAP (*Simple Object Access Protocol*) para definir la arquitectura y formato de sus mensajes.
 - Un servicio web basado en SOAP debe establecer un contrato formal para describir la interfaz que ofrece el servicio.

Servicios Web SOAP

- Adecuados para servicios web grandes.
- Utilizan el estándar SOAP (*Simple Object Access Protocol*) para definir la arquitectura y formato de sus mensajes.
 - Un servicio web basado en SOAP debe establecer un contrato formal para describir la interfaz que ofrece el servicio.
- Utilizan WSDL (*Web Services Definition Language*) para describir su interfaz sintácticamente.

- Adecuados para servicios web grandes.
- Utilizan el estándar SOAP (*Simple Object Access Protocol*) para definir la arquitectura y formato de sus mensajes.
 - Un servicio web basado en SOAP debe establecer un contrato formal para describir la interfaz que ofrece el servicio.
- Utilizan WSDL (*Web Services Definition Language*) para describir su interfaz sintácticamente.
 - Se enfoca más en los detalles del contrato, incluyen mensajes, operaciones, bindings y la localización del servicio web.

- Adecuados para servicios web grandes.
- Utilizan el estándar SOAP (*Simple Object Access Protocol*) para definir la arquitectura y formato de sus mensajes.
 - Un servicio web basado en SOAP debe establecer un contrato formal para describir la interfaz que ofrece el servicio.
- Utilizan WSDL (*Web Services Definition Language*) para describir su interfaz sintácticamente.
 - Se enfoca más en los detalles del contrato, incluyen mensajes, operaciones, bindings y la localización del servicio web.
- Netbeans facilita el desarrollo de aplicaciones de servicios web.

- RESTful (*Representational State Transfer Web Services*)

Servicios Web RESTful

- RESTful (*Representational State Transfer Web Services*)
- Adecuados para escenarios de integración básicos *ad-hoc*.

Servicios Web RESTful

- RESTful (*Representational State Transfer Web Services*)
- Adecuados para escenarios de integración básicos *ad-hoc*.
- Se integran mejor con HTTP que los SOAP ya que no requieren XML o definiciones del servicio en forma WSDL.

Servicios Web RESTful

- RESTful (*Representational State Transfer Web Services*)
- Adecuados para escenarios de integración básicos *ad-hoc*.
- Se integran mejor con HTTP que los SOAP ya que no requieren XML o definiciones del servicio en forma WSDL.
- Utilizan estándares muy conocidos: HTTP, URI, MIME.

Servicios Web RESTful

- RESTful (*Representational State Transfer Web Services*)
- Adecuados para escenarios de integración básicos *ad-hoc*.
- Se integran mejor con HTTP que los SOAP ya que no requieren XML o definiciones del servicio en forma WSDL.
- Utilizan estándares muy conocidos: HTTP, URI, MIME.
- Tienen una estructura muy “ligera”.

Servicios Web RESTful

- RESTful (*Representational State Transfer Web Services*)
- Adecuados para escenarios de integración básicos *ad-hoc*.
- Se integran mejor con HTTP que los SOAP ya que no requieren XML o definiciones del servicio en forma WSDL.
- Utilizan estándares muy conocidos: HTTP, URI, MIME.
- Tienen una estructura muy “ligera”.
- Su desarrollo es barato y sin muchas barreras para su adopción.

1 Servicios Web

- Introducción
- Tipos
- **Arquitectura de los Servicios Web**
- Servicios web SOAP y JavaEE
- Tecnologías básicas

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.

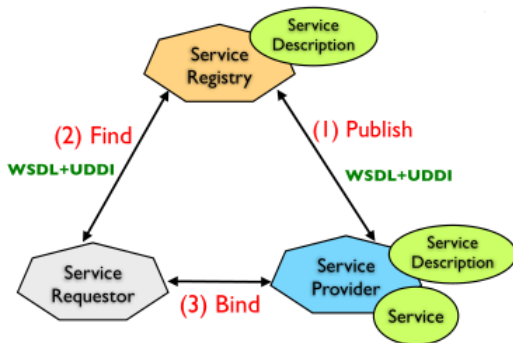
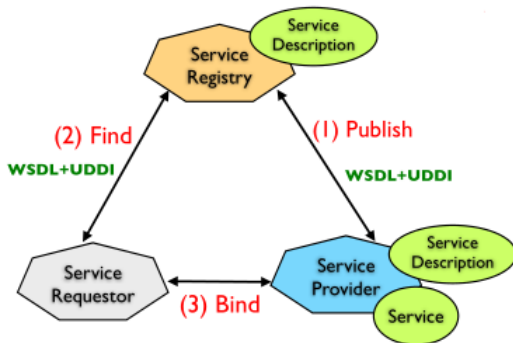


Figura: Arquitectura Orientada a Servicios

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.

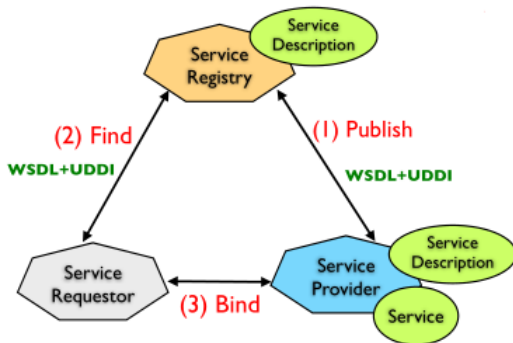


Permiten:

Figura: Arquitectura Orientada a Servicios

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.



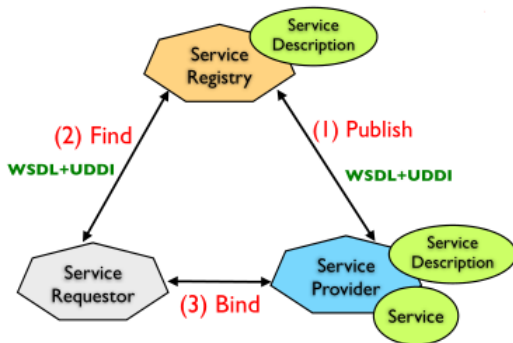
Permiten:

- implementar,

Figura: Arquitectura Orientada a Servicios

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.



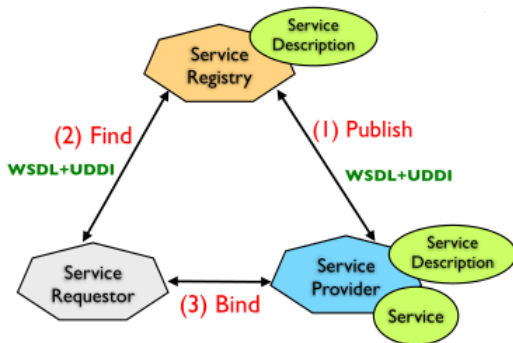
Permiten:

- implementar,
- publicar,

Figura: Arquitectura Orientada a Servicios

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.



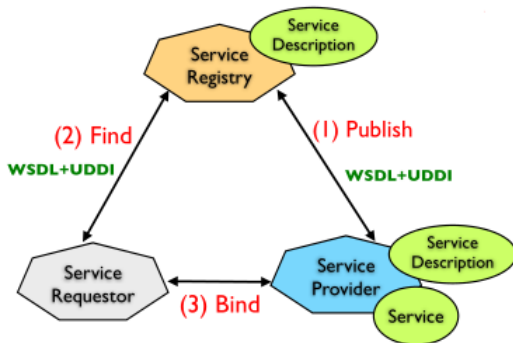
Permiten:

- implementar,
- publicar,
- localizar,

Figura: Arquitectura Orientada a Servicios

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.



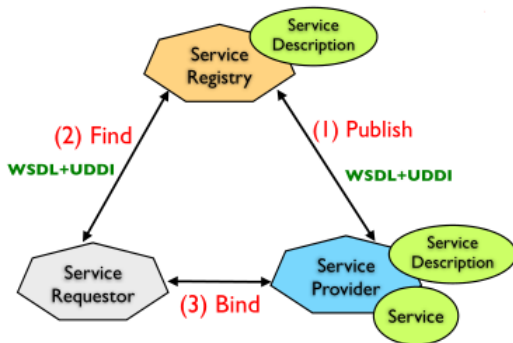
Permiten:

- implementar,
- publicar,
- localizar,
- seleccionar instancia y

Figura: Arquitectura Orientada a Servicios

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.



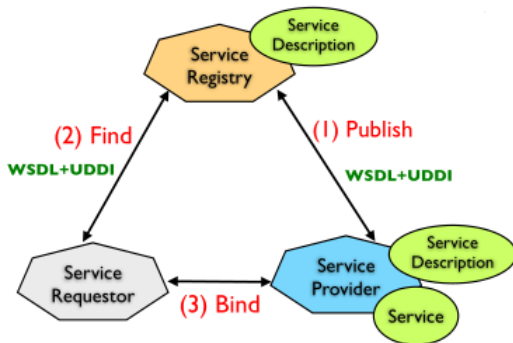
Permiten:

- implementar,
- publicar,
- localizar,
- seleccionar instancia y
- utilizar un servicio.

Figura: Arquitectura Orientada a Servicios

Arquitectura de los Servicios Web

- Los WS tienen una arquitectura orientada a servicios.



Permiten:

- implementar,
- publicar,
- localizar,
- seleccionar instancia y
- utilizar un servicio.

Figura: Arquitectura Orientada a Servicios

- UDDI (*Universal Description, Discovery and Integration*) Sirve para publicar y localizar servicios.

1 Servicios Web

- Introducción
- Tipos
- Arquitectura de los Servicios Web
- Servicios web SOAP y JavaEE
- Tecnologías básicas

- Los servicios web para JavaEE requieren que un componente Port (una instancia de un servicio) pueda ser referenciado desde un cliente, así como desde los contenedores web y EJB (*Enterprise JavaBeans*)

Servicios web SOAP y JavaEE

- Los servicios web para JavaEE requieren que un componente Port (una instancia de un servicio) pueda ser referenciado desde un cliente, así como desde los contenedores web y EJB (*Enterprise JavaBeans*)
- Estos servicios pueden implementarse de dos formas:

Servicios web SOAP y JavaEE

- Los servicios web para JavaEE requieren que un componente Port (una instancia de un servicio) pueda ser referenciado desde un cliente, así como desde los contenedores web y EJB (*Enterprise JavaBeans*)
- Estos servicios pueden implementarse de dos formas:
 - Como una clase Java que se ejecuta en un contenedor web.

Servicios web SOAP y JavaEE

- Los servicios web para JavaEE requieren que un componente Port (una instancia de un servicio) pueda ser referenciado desde un cliente, así como desde los contenedores web y EJB (*Enterprise JavaBeans*)
- Estos servicios pueden implementarse de dos formas:
 - Como una clase Java que se ejecuta en un contenedor web.
 - Como un EJB de sesión o singleton en un contenedor EJB.

- Los servicios web para JavaEE requieren que un componente Port (una instancia de un servicio) pueda ser referenciado desde un cliente, así como desde los contenedores web y EJB (*Enterprise JavaBeans*)
- Estos servicios pueden implementarse de dos formas:
 - Como una clase Java que se ejecuta en un contenedor web.
 - Como un EJB de sesión o singleton en un contenedor EJB.
- El contenedor actúa como mediador para acceder al servicio.

1 Servicios Web

- Introducción
- Tipos
- Arquitectura de los Servicios Web
- Servicios web SOAP y JavaEE
- Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.

Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.

Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:

Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:

Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:



Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:



Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:



- **Transporte de servicios:** mensajes entre aplicaciones. Utiliza **HTTP**, SMTP, FTP, BEEP.

Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:



- **Transporte de servicios:** mensajes entre aplicaciones. Utiliza **HTTP**, SMTP, FTP, BEEP.
- **Mensajería XML:** codifica mensajes en XML. Protocolos XML-RPC o SOAP.

Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:



- **Transporte de servicios:** mensajes entre aplicaciones. Utiliza **HTTP**, SMTP, FTP, BEEP.
- **Mensajería XML:** codifica mensajes en XML. Protocolos XML-RPC o SOAP.
- **Descripción de servicios:** define interfaz pública del servicio. Mediante WSDL.

Tecnologías básicas

- Todas las tecnologías fundamentales para el desarrollo de WS están basadas en XML.
- Son tecnologías independientes de SO y lenguaje de programación.
- Se organizan en capas:



- **Transporte de servicios:** mensajes entre aplicaciones. Utiliza **HTTP**, SMTP, FTP, BEEP.
- **Mensajería XML:** codifica mensajes en XML. Protocolos XML-RPC o SOAP.
- **Descripción de servicios:** define interfaz pública del servicio. Mediante WSDL.
- **Localización de servicios:** Registro centralizado de servicios. Utiliza el UDDI.

- Protocolo derivado de XML para intercambio de información entre aplicaciones.

SOAP

Tecnologías básicas

- Protocolo derivado de XML para intercambio de información entre aplicaciones.
- SOAP se utiliza para conectarse a un servicio e invocar métodos remotos.

- Protocolo derivado de XML para intercambio de información entre aplicaciones.
- SOAP se utiliza para conectarse a un servicio e invocar métodos remotos.
- Puede ser utilizado para enviar cualquier tipo de información. Dos tipos de mensajes según el contenido:

- Protocolo derivado de XML para intercambio de información entre aplicaciones.
- SOAP se utiliza para conectarse a un servicio e invocar métodos remotos.
- Puede ser utilizado para enviar cualquier tipo de información. Dos tipos de mensajes según el contenido:
 - **Orientados al documento** cualquier tipo de contenido que se desee enviar entre aplicaciones.

- Protocolo derivado de XML para intercambio de información entre aplicaciones.
- SOAP se utiliza para conectarse a un servicio e invocar métodos remotos.
- Puede ser utilizado para enviar cualquier tipo de información. Dos tipos de mensajes según el contenido:
 - **Orientados al documento** cualquier tipo de contenido que se desee enviar entre aplicaciones.
 - **Orientados a RPC** Tipo más concreto. Permiten invocar procedimientos de remotos (*Remote Procedure Calls*)

- Protocolo derivado de XML para intercambio de información entre aplicaciones.
- SOAP se utiliza para conectarse a un servicio e invocar métodos remotos.
- Puede ser utilizado para enviar cualquier tipo de información. Dos tipos de mensajes según el contenido:
 - **Orientados al documento** cualquier tipo de contenido que se desee enviar entre aplicaciones.
 - **Orientados a RPC** Tipo más concreto. Permiten invocar procedimientos de remotos (*Remote Procedure Calls*)
 - → petición (contenido del mensaje) contiene método a llamar y los parámetros.

- Protocolo derivado de XML para intercambio de información entre aplicaciones.
- SOAP se utiliza para conectarse a un servicio e invocar métodos remotos.
- Puede ser utilizado para enviar cualquier tipo de información. Dos tipos de mensajes según el contenido:
 - **Orientados al documento** cualquier tipo de contenido que se desee enviar entre aplicaciones.
 - **Orientados a RPC** Tipo más concreto. Permiten invocar procedimientos de remotos (*Remote Procedure Calls*)
 - → petición (contenido del mensaje) contiene método a llamar y los parámetros.
 - ← respuesta (mensaje SOAP) contiene los resultados devueltos.

- Protocolo derivado de XML para intercambio de información entre aplicaciones.
- SOAP se utiliza para conectarse a un servicio e invocar métodos remotos.
- Puede ser utilizado para enviar cualquier tipo de información. Dos tipos de mensajes según el contenido:
 - **Orientados al documento** cualquier tipo de contenido que se desee enviar entre aplicaciones.
 - **Orientados a RPC** Tipo más concreto. Permiten invocar procedimientos de remotos (*Remote Procedure Calls*)
 - → petición (contenido del mensaje) contiene método a llamar y los parámetros.
 - ← respuesta (mensaje SOAP) contiene los resultados devueltos.
- Diseñado para trabajar sobre HTTP, pero puede funcionar con otros protocolos.

Elementos de SOAP

Tecnologías básicas

- **Sobre SOAP** (Envelope). Contiene:



Elementos de SOAP

Tecnologías básicas

- **Sobre SOAP** (Envelope). Contiene:
 - Descripción del mensaje (destinatario, forma de procesarlo, definiciones de tipos)



Elementos de SOAP

Tecnologías básicas

- **Sobre SOAP** (Envelope). Contiene:
 - Descripción del mensaje (destinatario, forma de procesarlo, definiciones de tipos)
 - Cabecera (opcional) y cuerpo SOAP



Elementos de SOAP

Tecnologías básicas



- **Sobre SOAP** (Envelope). Contiene:
 - Descripción del mensaje (destinatario, forma de procesarlo, definiciones de tipos)
 - Cabecera (opcional) y cuerpo SOAP
- **Cabecera SOAP** (Header). Información sobre el mensaje (si es obligatorio, los actores, etc)

Elementos de SOAP

Tecnologías básicas



- **Sobre SOAP** (Envelope). Contiene:
 - Descripción del mensaje (destinatario, forma de procesarlo, definiciones de tipos)
 - Cabecera (opcional) y cuerpo SOAP
- **Cabecera SOAP** (Header). Información sobre el mensaje (si es obligatorio, los actores, etc)
- **Cuerpo SOAP** (Body). Contiene:

Elementos de SOAP

Tecnologías básicas



- **Sobre SOAP** (Envelope). Contiene:
 - Descripción del mensaje (destinatario, forma de procesarlo, definiciones de tipos)
 - Cabecera (opcional) y cuerpo SOAP
- **Cabecera SOAP** (Header). Información sobre el mensaje (si es obligatorio, los actores, etc)
- **Cuerpo SOAP** (Body). Contiene:
 - Mensaje (en caso de RPC la forma del mensaje se define por convención)

Elementos de SOAP

Tecnologías básicas



- **Sobre SOAP** (Envelope). Contiene:
 - Descripción del mensaje (destinatario, forma de procesarlo, definiciones de tipos)
 - Cabecera (opcional) y cuerpo SOAP
- **Cabecera SOAP** (Header). Información sobre el mensaje (si es obligatorio, los actores, etc)
- **Cuerpo SOAP** (Body). Contiene:
 - Mensaje (en caso de RPC la forma del mensaje se define por convención)
- **Error SOAP** (Fault) Indica en la respuesta que ha habido un error en el procesamiento de la petición.

Elementos de SOAP

Tecnologías básicas



- **Sobre SOAP** (Envelope). Contiene:
 - Descripción del mensaje (destinatario, forma de procesarlo, definiciones de tipos)
 - Cabecera (opcional) y cuerpo SOAP
- **Cabecera SOAP** (Header). Información sobre el mensaje (si es obligatorio, los actores, etc)
- **Cuerpo SOAP** (Body). Contiene:
 - Mensaje (en caso de RPC la forma del mensaje se define por convención)
- **Error SOAP** (Fault) Indica en la respuesta que ha habido un error en el procesamiento de la petición.
- **Anexo** (Attachment) **Opcional** permite enviar cualquier contenido (imágenes) junto al mensaje SOAP.

Ejemplos mensajes SOAP

- Mensaje de petición

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Ejemplos mensajes SOAP

- Mensaje de petición

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

- Mensaje de Respuesta

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

WSDL (*Web Services Description Language*)

Tecnologías básicas

- Basado en XML para describir la funcionalidad que proporciona un WS.

WSDL (*Web Services Description Language*)

Tecnologías básicas

- Basado en XML para describir la funcionalidad que proporciona un WS.
- Descripción de interfaz entendible por la máquina.

WSDL (*Web Services Description Language*)

Tecnologías básicas

- Basado en XML para describir la funcionalidad que proporciona un WS.
- Descripción de interfaz entendible por la máquina.
- Indica como se debe llamar al servicio, que parámetros espera y que estructuras de datos devuelve.

WSDL (*Web Services Description Language*)

Tecnologías básicas

- Basado en XML para describir la funcionalidad que proporciona un WS.
- Descripción de interfaz entendible por la máquina.
- Indica como se debe llamar al servicio, que parámetros espera y que estructuras de datos devuelve.
- Contiene la dirección *endpoint*. URL a la que hay que conectarse para acceder al servicio.

WSDL (*Web Services Description Language*)

Tecnologías básicas

- Basado en XML para describir la funcionalidad que proporciona un WS.
- Descripción de interfaz entendible por la máquina.
- Indica como se debe llamar al servicio, que parámetros espera y que estructuras de datos devuelve.
- Contiene la dirección *endpoint*. URL a la que hay que conectarse para acceder al servicio.
- Nos permite integrar un servicio automáticamente en nuestra aplicación o que otros usuarios utilicen los servicios que hayamos desarrollado nosotros.

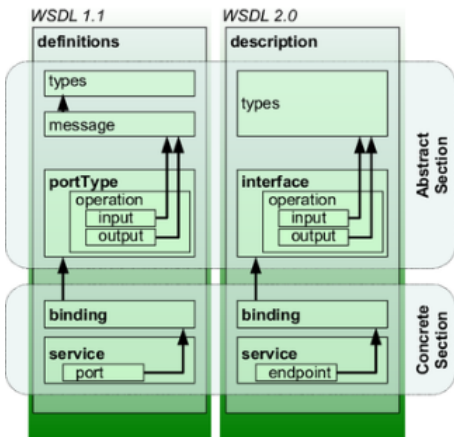
WSDL (*Web Services Description Language*)

Tecnologías básicas

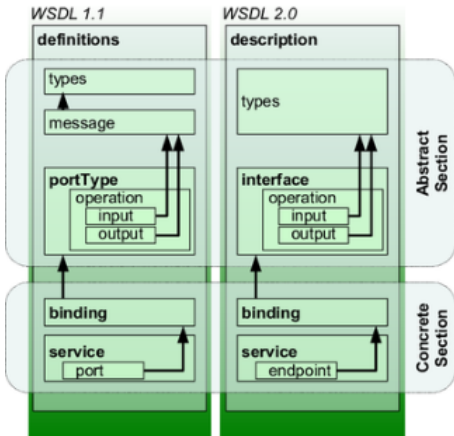
- Basado en XML para describir la funcionalidad que proporciona un WS.
- Descripción de interfaz entendible por la máquina.
- Indica como se debe llamar al servicio, que parámetros espera y que estructuras de datos devuelve.
- Contiene la dirección *endpoint*. URL a la que hay que conectarse para acceder al servicio.
- Nos permite integrar un servicio automáticamente en nuestra aplicación o que otros usuarios utilicen los servicios que hayamos desarrollado nosotros.
- La versión actual es la 2.0. Incluye ciertos cambios de estructura y nomenclatura a la 1.1. Cambia Description por Definition.

Estructura de un documento WSDL

- **Parte abstracta** QUÉ hace el servicio.

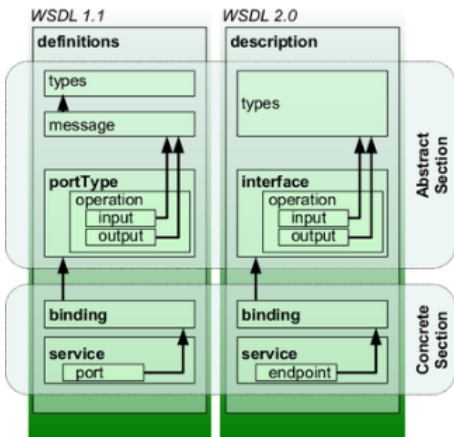


Estructura de un documento WSDL



- **Parte abstracta** QUÉ hace el servicio.
 - operaciones disp., E/S, mensajes de error, tipos de los mensajes.

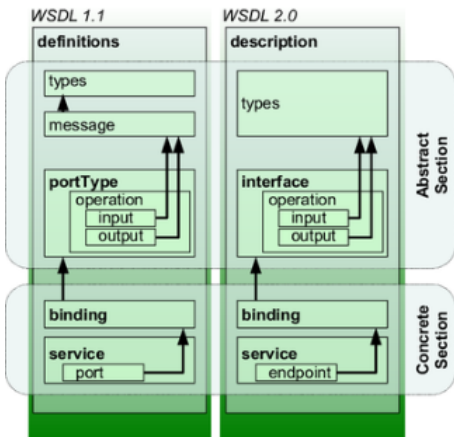
Estructura de un documento WSDL



- **Parte abstracta** QUÉ hace el servicio.

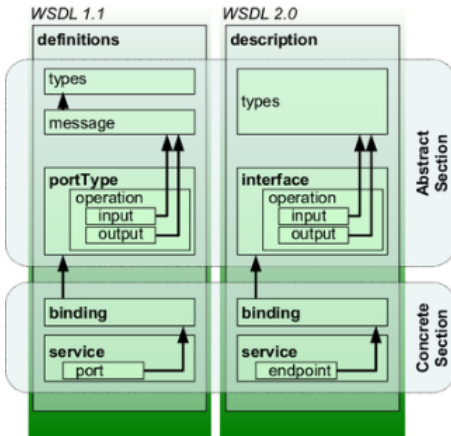
- operaciones disp., E/S, mensajes de error, tipos de los mensajes.
- En Java equivale a interfaz o clase abstracta.

Estructura de un documento WSDL



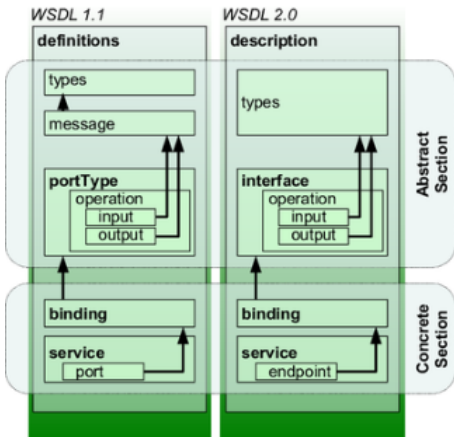
- **Parte abstracta** QUÉ hace el servicio.
 - operaciones disp., E/S, mensajes de error, tipos de los mensajes.
 - En Java equivale a interfaz o clase abstracta.
- **Parte concreta** CÓMO y DÓNDE del servicio.

Estructura de un documento WSDL



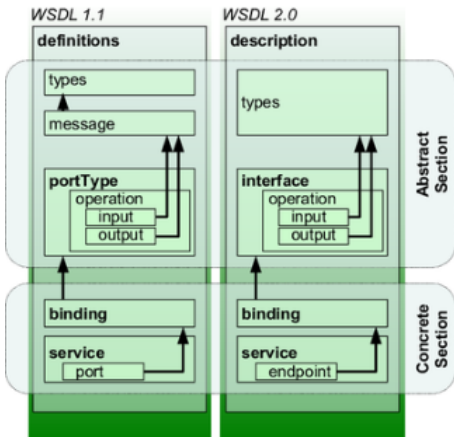
- **Parte abstracta** QUÉ hace el servicio.
 - operaciones disp., E/S, mensajes de error, tipos de los mensajes.
 - En Java equivale a interfaz o clase abstracta.
- **Parte concreta** CÓMO y DÓNDE del servicio.
 - Cómo se debe llamar (formato de los datos: SOAP)

Estructura de un documento WSDL



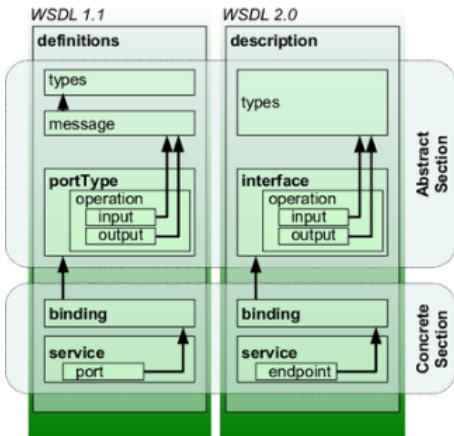
- **Parte abstracta** QUÉ hace el servicio.
 - operaciones disp., E/S, mensajes de error, tipos de los mensajes.
 - En Java equivale a interfaz o clase abstracta.
- **Parte concreta** CÓMO y DÓNDE del servicio.
 - Cómo se debe llamar (formato de los datos: SOAP)
 - protocolo de acceso (red)

Estructura de un documento WSDL



- **Parte abstracta** QUÉ hace el servicio.
 - operaciones disp., E/S, mensajes de error, tipos de los mensajes.
 - En Java equivale a interfaz o clase abstracta.
- **Parte concreta** CÓMO y DÓNDE del servicio.
 - Cómo se debe llamar (formato de los datos: SOAP)
 - protocolo de acceso (red)
 - dónde está el servicio (URL)

Estructura de un documento WSDL



- **Parte abstracta** QUÉ hace el servicio.
 - operaciones disp., E/S, mensajes de error, tipos de los mensajes.
 - En Java equivale a interfaz o clase abstracta.
- **Parte concreta** CÓMO y DÓNDE del servicio.
 - Cómo se debe llamar (formato de los datos: SOAP)
 - protocolo de acceso (red)
 - dónde está el servicio (URL)
 - En Java equivale a la implementación de la clase abstracta.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: *xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"*
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.
- **message** define mensajes de E/S intercambiados en cada operación del servicio.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.
- **message** define mensajes de E/S intercambiados en cada operación del servicio.
- **portType** Colección de operaciones.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.
- **message** define mensajes de E/S intercambiados en cada operación del servicio.
- **portType** Colección de operaciones.
 - Cada operación indica mensajes E/S basado en el elemento `messages`. Operaciones abstractas del servicio.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.
- **message** define mensajes de E/S intercambiados en cada operación del servicio.
- **portType** Colección de operaciones.
 - Cada operación indica mensajes E/S basado en el elemento `messages`. Operaciones abstractas del servicio.
- **binding** indica protocolo de red y formato de datos para las operaciones del `portType`.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.
- **message** define mensajes de E/S intercambiados en cada operación del servicio.
- **portType** Colección de operaciones.
 - Cada operación indica mensajes E/S basado en el elemento `messages`. Operaciones abstractas del servicio.
- **binding** indica protocolo de red y formato de datos para las operaciones del `portType`.
 - Definiciones concretas de los `portTypes`. Un `portType` puede tener varios `bindings` asociados.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.
- **message** define mensajes de E/S intercambiados en cada operación del servicio.
- **portType** Colección de operaciones.
 - Cada operación indica mensajes E/S basado en el elemento `messages`. Operaciones abstractas del servicio.
- **binding** indica protocolo de red y formato de datos para las operaciones del `portType`.
 - Definiciones concretas de los `portTypes`. Un `portType` puede tener varios `bindings` asociados.
- **service** define el servicio como una colección de elementos `port` a los que se puede acceder.

Elementos WSDL (versión 1.1)

- **definitions** Raíz. Especifica namespace del documento, nombre y prefijos. Ej: `xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"`
- **types** define tipos de datos intercambiados en el mensaje.
 - Podemos definir directamente o importar un fichero XSD si no son tipos primitivos.
- **message** define mensajes de E/S intercambiados en cada operación del servicio.
- **portType** Colección de operaciones.
 - Cada operación indica mensajes E/S basado en el elemento `messages`. Operaciones abstractas del servicio.
- **binding** indica protocolo de red y formato de datos para las operaciones del `portType`.
 - Definiciones concretas de los `portTypes`. Un `portType` puede tener varios `bindings` asociados.
- **service** define el servicio como una colección de elementos `port` a los que se puede acceder.
 - Un `port` se define asociando una dirección de red con un `binding`. Esta URL es donde el servicio actúa (dirección de acceso al servicio).

<definitions>

<types> *tipos de datos, si no son primitivos*

<message> *llamadas y respuestas SOAP*

<portType> *(INTERFAZ) operaciones: llamada + respuesta*

<binding> *protocolo de red y formato de datos SOAP*

<service> *URL del servicio para acceder a una colección de ports*

Ejemplo de documento WSDL (I)

```
<?xml version="1.0" encoding="utf-8"?>
<definitions targetNamespace="http://jaxwsHelloServer/"
name="HelloService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://jaxwsHelloServer/"
        schemaLocation="http://localhost:8080/JAXWSHelloAppServer/
          jaxwsHello?xsd=1" />
    </xsd:schema>
  </types>
  <message name="sayHello">
    <part name="parameters" element="tns:sayHello" />
  </message>
  <message name="sayHelloResponse">
    <part name="parameters" element="tns:sayHelloResponse" />
  </message>
</definitions>
```

los tipos se definen en el fichero xsd

Ejemplo de documento WSDL (II)

```
<portType name="Hello">
  <operation name="sayHello">
    <input wsam:Action="http://jaxwsHelloServer/Hello/sayHelloRequest"
      message="tns:sayHello"/>
    <output wsam:Action="http://jaxwsHelloServer/Hello/sayHelloResponse"
      message="tns:sayHelloResponse"/>
  </operation>
</portType>

<binding name="HelloPortBinding" type="tns:Hello">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
  <operation name="sayHello">
    <soap:operation soapAction="">
      <input> <soap:body use="literal"/> </input>
      <output> <soap:body use="literal"/> </output>
    </soap:operation>
  </operation>
</binding>

<service name="HelloService">
  <port name="HelloPort" binding="tns:HelloPortBinding">
    <soap:address location="http://localhost:8080/JAXWSHelloAppServer/
      jaxwsHello"/>
  </port>
</service>
```

operaciones soportadas por el servicio

protocolo de red y formato de los datos

dirección donde localizar el servicio

- Instalar plugin XML de Netbeans.

Edición de documentos WSDL con Netbeans

- Instalar plugin XML de Netbeans.
- Tools->Plugins->Settings->Add. Name: deadlock.netbeans.org. URL:
`http://deadlock.netbeans.org/job/xml/
lastSuccessfulBuild/artifact/`

Edición de documentos WSDL con Netbeans

- Instalar plugin XML de Netbeans.
- Tools->Plugins->Settings->Add. Name: deadlock.netbeans.org. URL:
`http://deadlock.netbeans.org/job/xml/
lastSuccessfulBuild/artifact/`
- Pestaña Available Plugins-> XML Tools-> Install.

Edición de documentos WSDL con Netbeans

- Instalar plugin XML de Netbeans.
- Tools->Plugins->Settings->Add. Name: deadlock.netbeans.org. URL:
`http://deadlock.netbeans.org/job/xml/
lastSuccessfulBuild/artifact/`
- Pestaña Available Plugins-> XML Tools-> Install.
- Este plugin nos facilita el trabajo con ficheros WSDL y XSD.

Ejemplo de uso de XSD en WSDL

- ¿Qué es un XML Schema?

Ejemplo de uso de XSD en WSDL

- ¿Qué es un XML Schema?
- un fichero WSDL utiliza ficheros XSD para definir los tipos de mensajes que se pueden utilizar como interfaz para comunicarnos con un WS.

Ejemplo de uso de XSD en WSDL

- ¿Qué es un XML Schema?
- un fichero WSDL utiliza ficheros XSD para definir los tipos de mensajes que se pueden utilizar como interfaz para comunicarnos con un WS.
- Ejemplo Schema:

```
<xs:element name="CustomerAddress" type="xs:string"/>
```


Ejemplo de uso de XSD en WSDL

- ¿Qué es un XML Schema?
- un fichero WSDL utiliza ficheros XSD para definir los tipos de mensajes que se pueden utilizar como interfaz para comunicarnos con un WS.
- Ejemplo Schema:

```
<xs:element name="CustomerAddress" type="xs:string"/>
```

- Utilizamos este elemento en un WSDL:

```
<message name="msgResponse">  
  <part name="parameters" element="tns:CustomerAddress"/>  
</message>
```

Ejemplo de uso de XSD en WSDL

- ¿Qué es un XML Schema?
- un fichero WSDL utiliza ficheros XSD para definir los tipos de mensajes que se pueden utilizar como interfaz para comunicarnos con un WS.
- Ejemplo Schema:

```
<xs:element name="CustomerAddress" type="xs:string"/>
```

- Utilizamos este elemento en un WSDL:

```
<message name="msgResponse">  
  <part name="parameters" element="tns:CustomerAddress"/>  
</message>
```

- Mensaje de respuesta:

```
<Customer_address>Calle de los Pinos, 37</Customer_address>
```

Tipos en XSD

```
<xsd:element name="Customer" type="tns:CustomerType"/>
<xsd:complexType name="CustomerType">
  <xsd:sequence>
    <xsd:element name="Phone" type="xsd:integer"/>
    <xsd:element name="Addresses" type="tns:AddressType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AddressType">
  <xsd:sequence>
    <xsd:element name="Address1" type="xsd:string"/>
    <xsd:element name="Address2" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Ejemplo de uso de XSD en WSDL

Tipos en XSD

```
<xsd:element name="Customer" type="tns:CustomerType"/>
<xsd:complexType name="CustomerType">
  <xsd:sequence>
    <xsd:element name="Phone" type="xsd:integer"/>
    <xsd:element name="Addresses" type="tns:AddressType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AddressType">
  <xsd:sequence>
    <xsd:element name="Address1" type="xsd:string"/>
    <xsd:element name="Address2" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Definición de mensaje en WSDL

```
<message name="msgResponse2">
  <part name="parameters" element="tns:Customer"/>
</message>
```

Ejemplo de uso de XSD en WSDL

Tipos en XSD

```
<xsd:element name="Customer" type="tns:CustomerType"/>
<xsd:complexType name="CustomerType">
  <xsd:sequence>
    <xsd:element name="Phone" type="xsd:integer"/>
    <xsd:element name="Addresses" type="tns:AddressType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AddressType">
  <xsd:sequence>
    <xsd:element name="Address1" type="xsd:string"/>
    <xsd:element name="Address2" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Definición de mensaje en WSDL

```
<message name="msgResponse2">
  <part name="parameters" element="tns:Customer"/>
</message>
```

Ejemplo de mensaje

```
<Customer>
  <Phone>12345678</Phone>
  <Address1>Calle de los Pinos, 37</Address1>
  <Address2>Calle de los Manzanos, 25</Address2>
</Customer>
```

- Nuevo archivo->XML->XML Schema

- Nuevo archivo->XML->XML Schema
- Desarrollar el XML Schema Customer:

```
<xsd:element name="Customer" type="tns:CustomerType"/>
<xsd:complexType name="CustomerType">
  <xsd:sequence>
    <xsd:element name="Phone" type="xsd:integer"/>
    <xsd:element name="Addresses" type="tns:AddressType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AddressType">
  <xsd:sequence>
    <xsd:element name="Address1" type="xsd:string"/>
    <xsd:element name="Address2" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Ejercicio: Edición de WSDL con Netbeans

- Vamos a desarrollar el fichero hello.wsdl del ejemplo mostrado anteriormente (ubicado en el repositorio).

Ejercicio: Edición de WSDL con Netbeans

- Vamos a desarrollar el fichero hello.wsdl del ejemplo mostrado anteriormente (ubicado en el repositorio).
- Nuevo Proyecto->Java Web->Web Application. Nombre: Hello.

Ejercicio: Edición de WSDL con Netbeans

- Vamos a desarrollar el fichero hello.wsdl del ejemplo mostrado anteriormente (ubicado en el repositorio).
- Nuevo Proyecto->Java Web->Web Application. Nombre: Hello.
- Server: GlassFish Server. JavaEE 7Web. Finalizar.

Ejercicio: Edición de WSDL con Netbeans

- Vamos a desarrollar el fichero hello.wsdl del ejemplo mostrado anteriormente (ubicado en el repositorio).
- Nuevo Proyecto->Java Web->Web Application. Nombre: Hello.
- Server: GlassFish Server. JavaEE 7Web. Finalizar.
- Nuevo archivo->XML->XML Schema. Un Schema simple para los mensajes de entrada y respuesta del WS.

Ejercicio: Edición de WSDL con Netbeans

- Vamos a desarrollar el fichero hello.wsdl del ejemplo mostrado anteriormente (ubicado en el repositorio).
- Nuevo Proyecto->Java Web->Web Application. Nombre: Hello.
- Server: GlassFish Server. JavaEE 7Web. Finalizar.
- Nuevo archivo->XML->XML Schema. Un Schema simple para los mensajes de entrada y respuesta del WS.
- Nuevo archivo->XML-> Documento WSDL.

Ejercicio: Edición de WSDL con Netbeans

- Vamos a desarrollar el fichero hello.wsdl del ejemplo mostrado anteriormente (ubicado en el repositorio).
- Nuevo Proyecto->Java Web->Web Application. Nombre: Hello.
- Server: GlassFish Server. JavaEE 7Web. Finalizar.
- Nuevo archivo->XML->XML Schema. Un Schema simple para los mensajes de entrada y respuesta del WS.
- Nuevo archivo->XML-> Documento WSDL.
- Nombre: Hello. Documento WSDL Concreto. Binding: SOAP. Type: Document Literal.

Ejercicio: Edición de WSDL con Netbeans

- Vamos a desarrollar el fichero hello.wsdl del ejemplo mostrado anteriormente (ubicado en el repositorio).
- Nuevo Proyecto->Java Web->Web Application. Nombre: Hello.
- Server: GlassFish Server. JavaEE 7Web. Finalizar.
- Nuevo archivo->XML->XML Schema. Un Schema simple para los mensajes de entrada y respuesta del WS.
- Nuevo archivo->XML-> Documento WSDL.
- Nombre: Hello. Documento WSDL Concreto. Binding: SOAP. Type: Document Literal.
- PortTypeName. Operation Name. input. Output.