Arquitectura Orientada a Servicios

Servicios Web

Edwin Salvador

05 de junio de 2015

Sesión 9

Contenido I

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

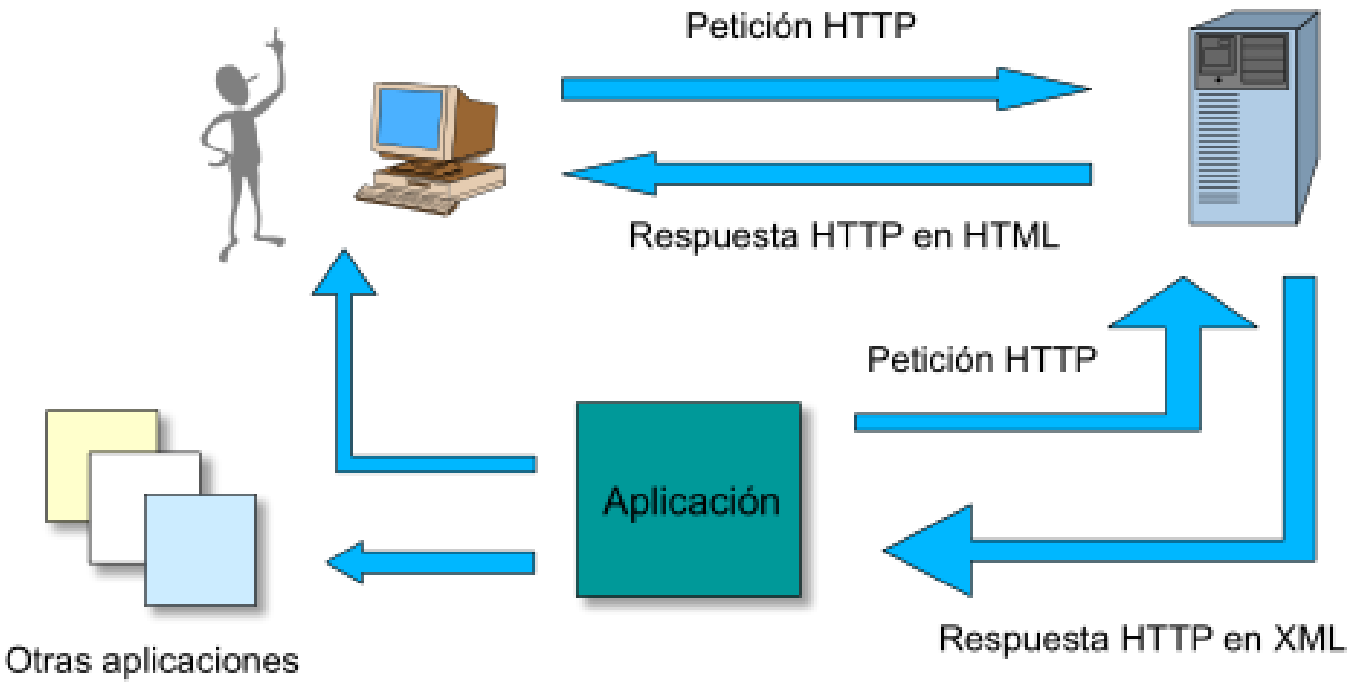
Contenido I

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.
- 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



Características

- **Accesible a través de la Web**. Protocolos HTTTP 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.

Contenido I

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

Tipos

- 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.
- 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.

Servicios Web RESTful

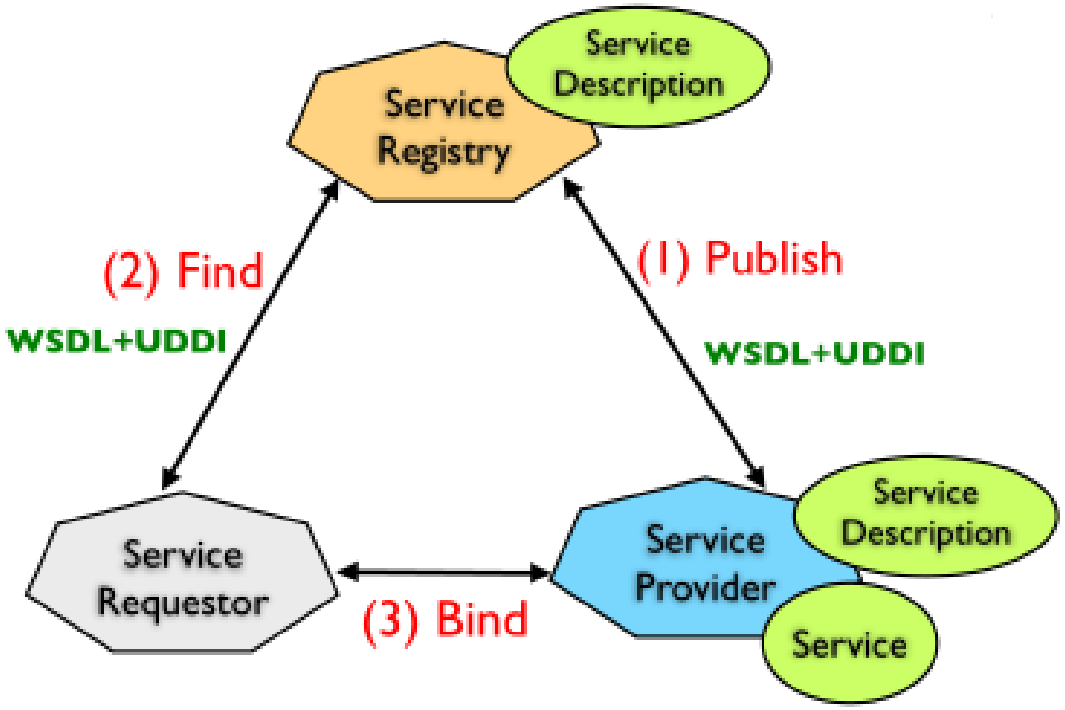
- RESTful (*Representational State Transfer Web Services*)
- Adecuados para escenarios de integración básicos *ad-hoc*.
- Se integran mejor con HTTP que los SOAP ya que no requieren XML o definiciones del servicio en forma WSDL.
- Utilizan estándares muy conocidos: HTTP, URI, MIME.
- Tienen una estructura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

Contenido I

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.



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.

Contenido I

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

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.
- El contenedor actúa como mediador para acceder al servicio.

Contenido I

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

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 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*)
    - $\rightarrow$ petición (contenido del mensaje) contiene método a llamar y los parámetros.
    - $\leftarrow$ 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:
  - 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>
```

- Mensaje de Respuesta

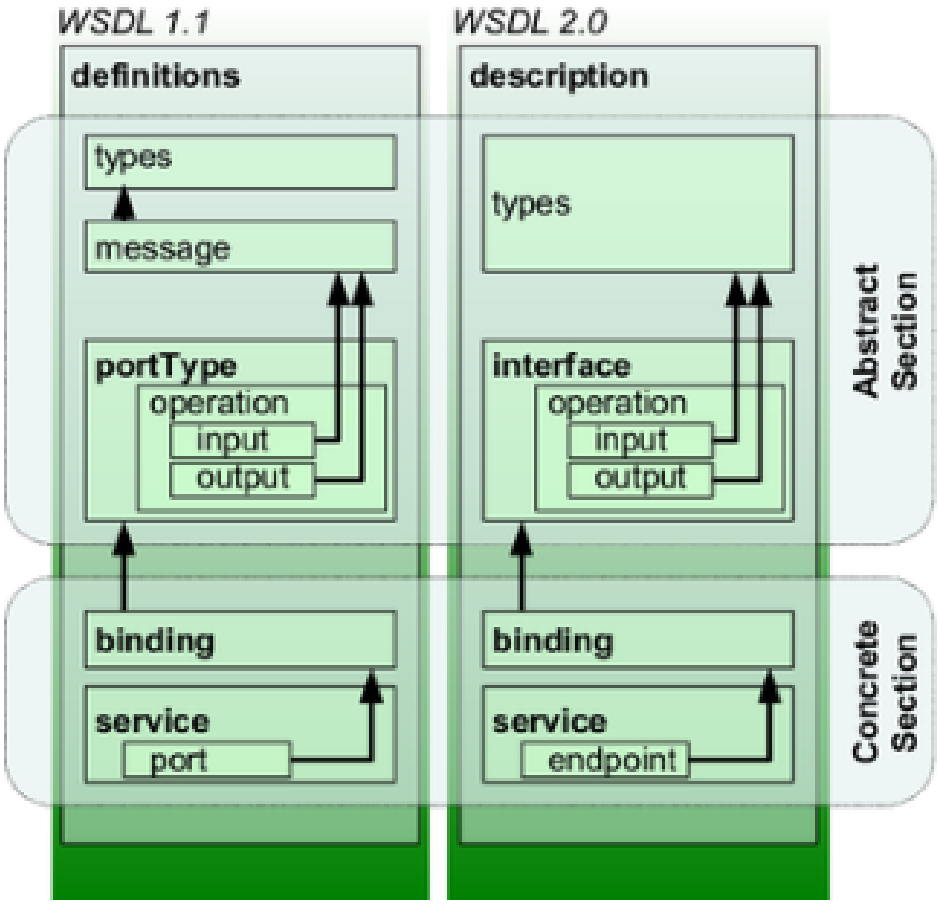
```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
      soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>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

WSDL (*Web Services Description Language*)

Tecnologías básicas

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

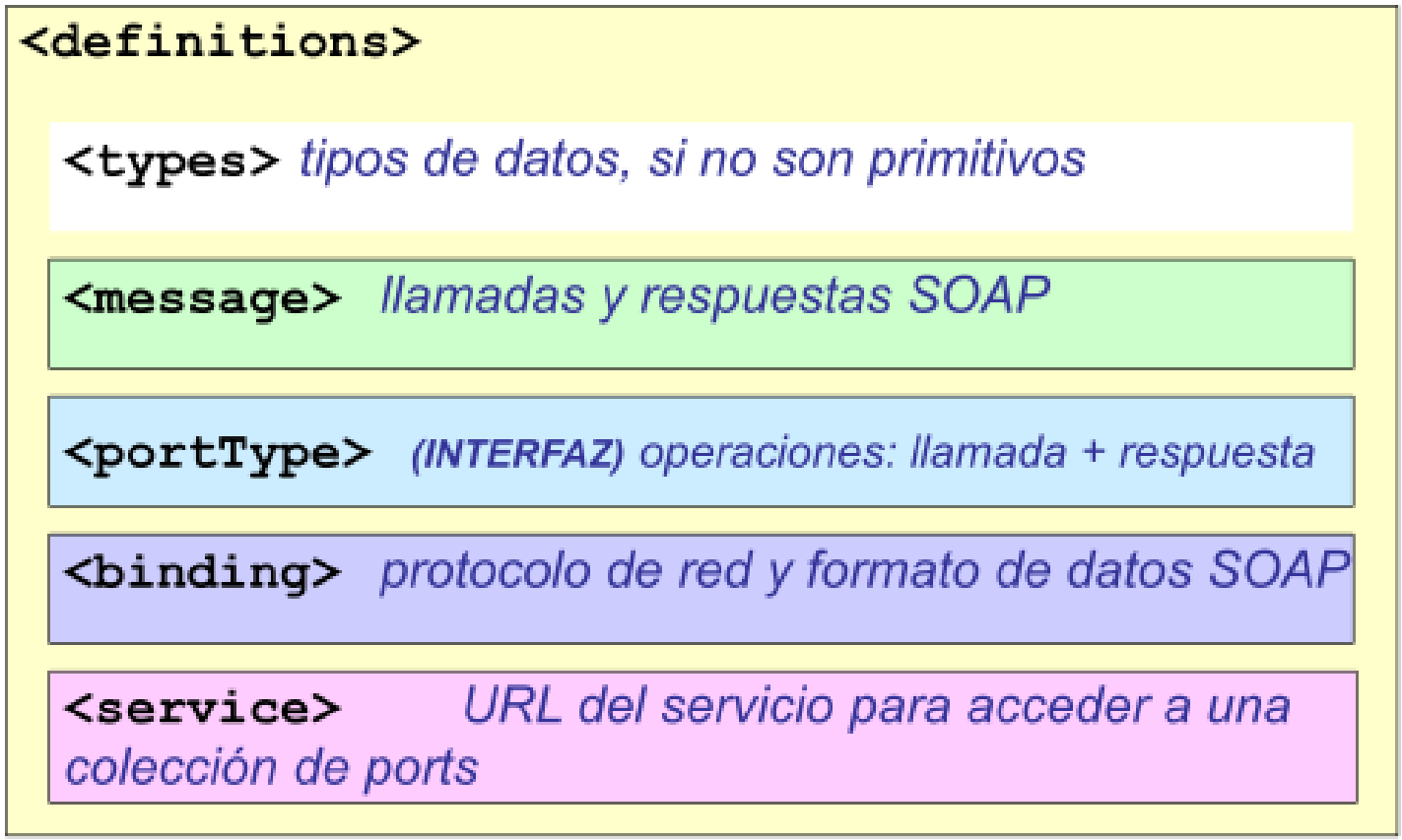


- **Parte abstracta** QUÉ hace el servicio.
  - operaciones disp., E/S, mensajes de error, tipos de los mensajes.
  - En Java equivale a interfaz o clase abstracta.
- **Parte concreta** CÓMO y DÓNDE del servicio.
  - Cómo se debe llamar (formato de los datos: SOAP)
  - protocolo de acceso (red)
  - dónde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

Elementos WSDL



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>
```

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>
    </operation>
</binding>

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

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?
- 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>
```

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>
```

Edición de XML Schemas con Netbeans

- 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).
- 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.