# Arquitectura SOAP / WSDL

Antonio Espín Herranz

# Contenidos

- Arquitectura SOAP
- WSDL: Web Services Description Language

# SOAP

- Arquitectura SOA:
  - Arquitectura Orientada a Servicios

- Define un estilo **basado en estándares** e integrado en distintas plataformas, distintos S.O. y lenguajes de programación. ***No se ha inventado nada nuevo***.

- Permite:
  - La **invocación de funcionalidades** desplegadas en la red (***como si estuvieran en local***), sin tener en cuenta el SO o el lenguaje de programación.

# ¿Qué son los Web Services?

- **Definiciones**:
  - Son colecciones de funciones que son presentadas como una sola entidad y anunciadas en la red para ser usadas por otras aplicaciones.
  - Componentes distribuidos accesibles a través de la Web (o intranet) definibles, soportados y descubiertos mediante protocolos estándares.
- Los servicios Web son identificados mediante una **URI** (**I**dentificador de **R**ecursos **U**niversal)

# Introducción

- **¿Qué necesitamos para consumir un WS?**
  - ¿Dónde está?
    - La **URI** por la que atiende → El ***endPoint*** del Servicio.
  - ¿Qué operación queremos ejecutar?
    - Utilizar la **funcionalidad** que nos ofrece → El método.
  - ¿Qué parámetros necesita la operación a ejecutar?
  - Recoger el resultado.

  - **Estas respuestas están en el contrato que publica el WS**.
  - Los WS son auto descriptivos → **WSDL (Es un fichero XML)**
  - Normalmente cuando a la URI del servicio le añadimos el parámetro **?WSDL** muestra el contenido del contrato (wsdl)

- **¿Cómo nos comunicamos con los WS?**
  - Mediante mensajes → **SOAP** (la base es el lenguaje XML)
  - Petición / Respuesta.
  - Sobre protocolos standard → http, smtp, etc.

- **¿Dónde están los WS?**
  - ¿Cómo los localizamos?
    - En los registros **UDDI**. Y aquí también se pueden publicar.
    - O el proveedor de Servicios proporciona los **endPoint**.

# Arquitectura SOAP

- Se compone de los siguientes estándares:
  - XML
  - Protocolo HTTP
  - Protocolo SOAP (basado en mensajes en formato XML)
  - WSLD: Lenguaje de descripción de servicios.
    - Documento auto descriptivo de las operaciones desplegadas por el Web Services.
    - Con lo que expone el WSDL es suficiente para conectar con el WS

# Arquitectura SOAP

- **¿Porqué surgen los WS?**
  - Necesidad de **establecer un intercambio de datos normalizado entre aplicaciones** distintas en un contexto globalizado (Internet).
    - Integrar aplicaciones que están programadas en distintos lenguajes (Cobol, C, Java, etc.)
    - Que se ejecutan en equipos diferentes.
    - Con Sistemas Operativos distintos.

# Características SOA

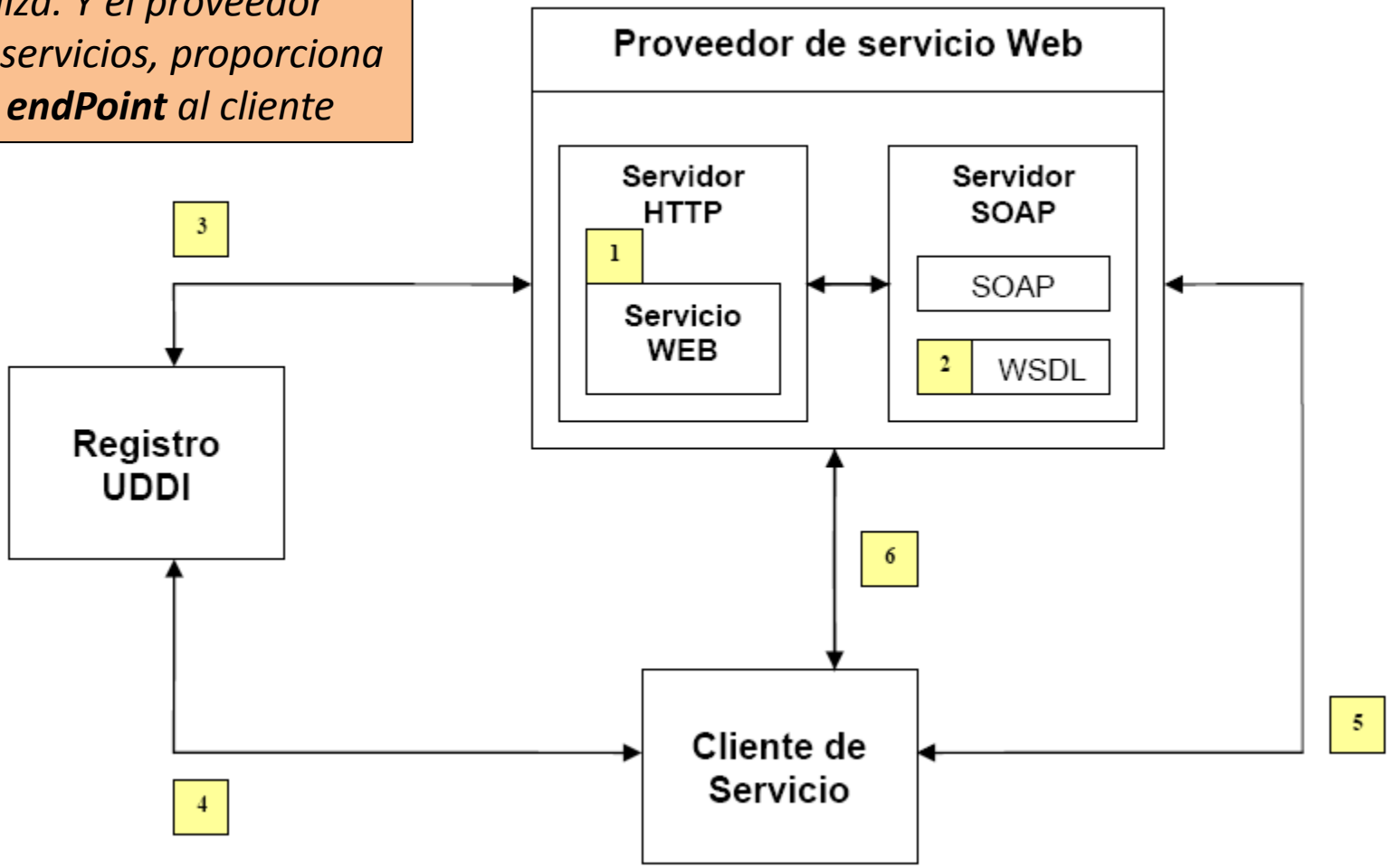
- **Bajo acoplamiento**:
  - Independencia en cuanto S.O., lenguaje de programación.
- **Granularidad gruesa**.
  - Operaciones de alto nivel.
- **Componentes reutilizables.**
  - Es muy fácil para una empresa exponer procesos para que sean consumidos por los clientes.
- **Mensajería.**
  - Tanto el emisor como el receptor intercambian mensajes SOAP.

# Entidades SOA

- **Servicio**: Componente / Programa con el que interactuamos intercambiando mensajes.
- **Consumidor del Servicio**: es una aplicación, servicio, o módulo de software que requiere el servicio.
- **Proveedor del Servicio**: La entidad direccionable mediante la red que acepta y ejecuta solicitudes desde los consumidores.
- **Registro de Servicios**: Directorio en la red que contiene los servicios disponibles.
- **Contrato del Servicio**: Es una especificación de la forma en que un consumidor del servicio debe interactuar con el proveedor del servicio.

# Registro Publicación Búsqueda Enlace del WS

*Normalmente la parte de publicación NO se utiliza. Y el proveedor de servicios, proporciona los **endPoint** al cliente*



# Descripción del Proceso

1. Un proveedor de servicio crea un servicio Web.

2. El proveedor de servicio utiliza WSDL para describir el servicio.

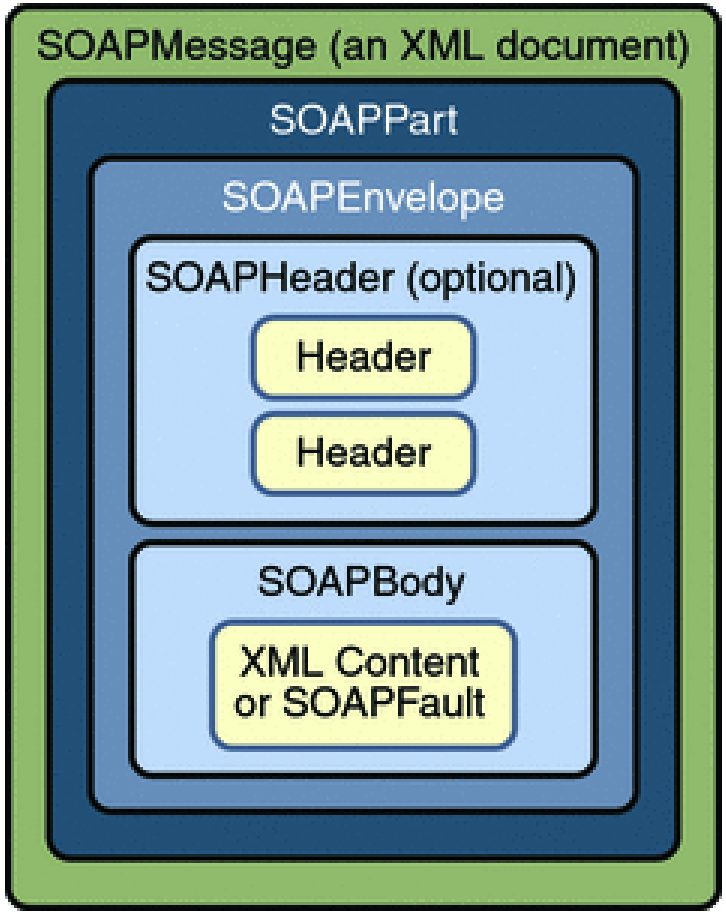
3. El proveedor de servicio registra el servicio en un registro UDDI.

4. Otro servicio o usuario localiza y solicita el servicio registrado al consultar los registros UDDI.

5. El cliente del servicio escribe una aplicación que liga el servicio registrado utilizando SOAP.

6. Se intercambian datos y mensajes XML sobre HTTP.

# Estructura de los mensajes SOAP




- Documentos **XML**.
- Sobre protocolos estándar:
  - http, smtp, ftp.

```
<s:Envelope xmlns=…>
    <s:Header>
        <!- - bloques de la cab.- - >
    </s:Header>
    <s:Body>
        <!- - Bloques del body - - >
    </s:Body>
</s:Envelope>
```

# Ejemplo de Petición

```
<m:Envelope
    xmlns:m=”http://www.w3.org/2001/6/soap-envelope”
    xmlns:e=“http://www.acme.org/stocks”>
     <m:Body>
          <e:obtenerStockLibro m:encodingStyle=“http://www.w3.org/2001/6/soap-
             encoding”>
               <e:codigoLibro xsi:type=“xsd:string”>FFJAJF</e:codigoLibro>
               </e:codigoLibro>
               <e:codigoEditorial xsi:type=“xsd:int”>FFJAJF
               </e:codigoEditorial>
          </e:obtenerStockLibro>
     </m:Body>
</m:Envelope>
```

# Ejemplo de Respuesta

```
<m:Envelope
    xmlns:m=”http://www.w3.org/2001/6/soap-envelope” xmlns:e=“http://www.acme.org/stocks”>
    <m:Body>
        <e:obtenerStockLibroResponse
            m:encodingStyle=“http://www.w3.org/2001/6/soap-encoding”>
            <e:response xsi:type=“xsd:int”>17</e:response>
        </ e:obtenerStockLibroResponse >
    </m:Body>
</m:Envelope>
```

# soap:Fault

- Los errores son tratados utilizando un elemento especializado llamado *Fault Envelope.*

- El elemento *SOAP Fault* aparece como un hijo inmediato del elemento *Body.*

- Elementos:
  - Faultcode: Código de error.
  - FaultString: Explicación del código de error.
  - FaulFactor: Quién es el causante del error.
  - Details: Da información específica de la aplicación que causo el error.

- <**faultcode**> y <**faultstring**> son requeridos

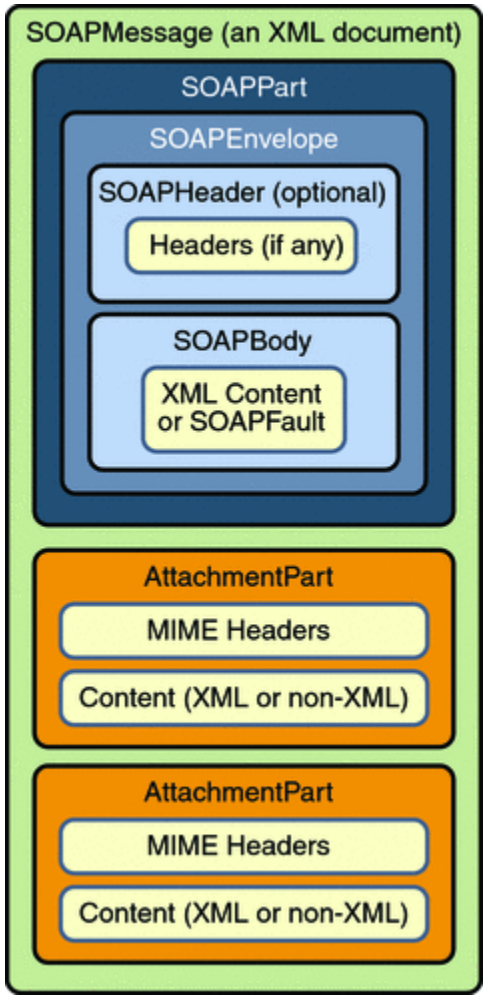
# Ejemplo

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
   <SOAP-ENV:Body>

        <SOAP-ENV:Fault>
                <faultcode>SOAP-ENV:Server</faultcode>
                <faultstring>Test fault</faultstring>
                <faultactor>/soap/servlet/rpcrouter</faultactor>
                <detail>
                        ..
                </detail>
        </SOAP-ENV:Fault>


   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# Estructura with Attachment




- Dentro de un mensaje SOAP se puede adjuntar datos binarios.
- Puede haber varios attachment dentro de un mismo mensaje.
- *Analogía : email attachments.*

# WSDL

- Lenguaje de Descripción de Servicios Web.
  - En la especificación 1.1
- Lenguaje de Definición de Servicios Web.
  - En la especificación 2.0

- Es un documento que describe la funcionalidad que proporciona un servicio Web.

- Está escrito en Lenguaje XML Schema.

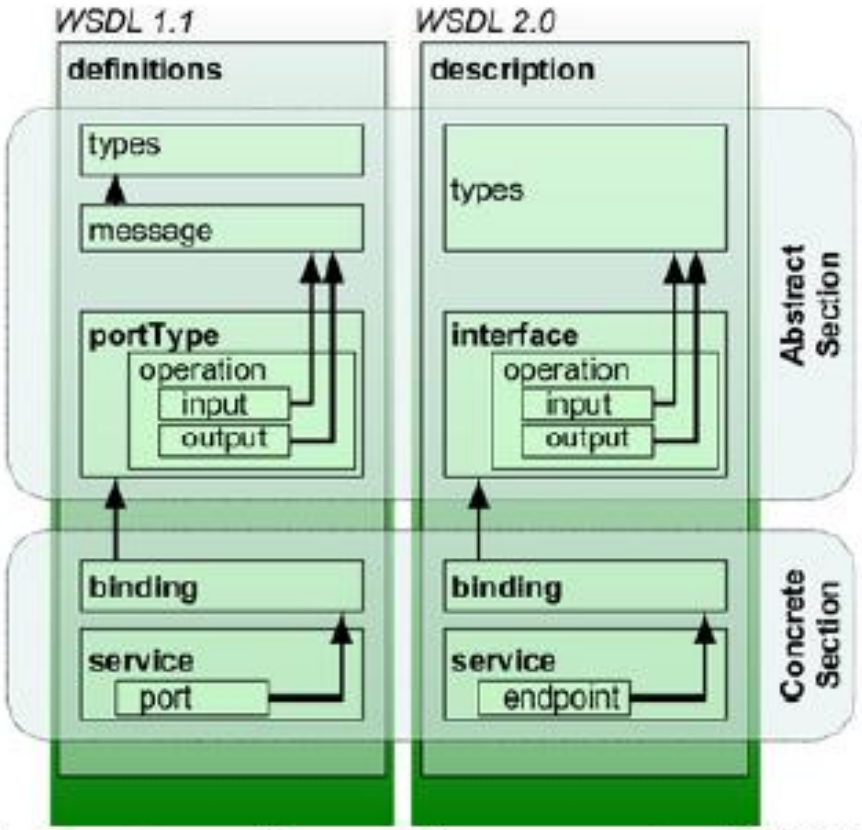
- Proporcional una descripción entendible por una máquina indicando de cómo se debe de llamar al Servicio y que parámetros espera y que devuelve.

- Recibe el nombre de **CONTRATO**.

# WSDL

- Se compone de etiquetas XML.
- Se puede dividir en dos partes:
  - Una **abstracta**:
    - Que hace el servicio
      - Operaciones disponibles
      - Entradas, Salidas y mensajes de error
      - Definiciones de los tipos para los mensajes de entrada, salida y error.
  - Y otra **concreta**:
    - Cómo y donde se encuentra el servicio
      - Cómo tiene que llamar un cliente al servicio.
      - Que protocolo debería de usar.
      - Donde está disponible el servicio

# WSDL: Estructura

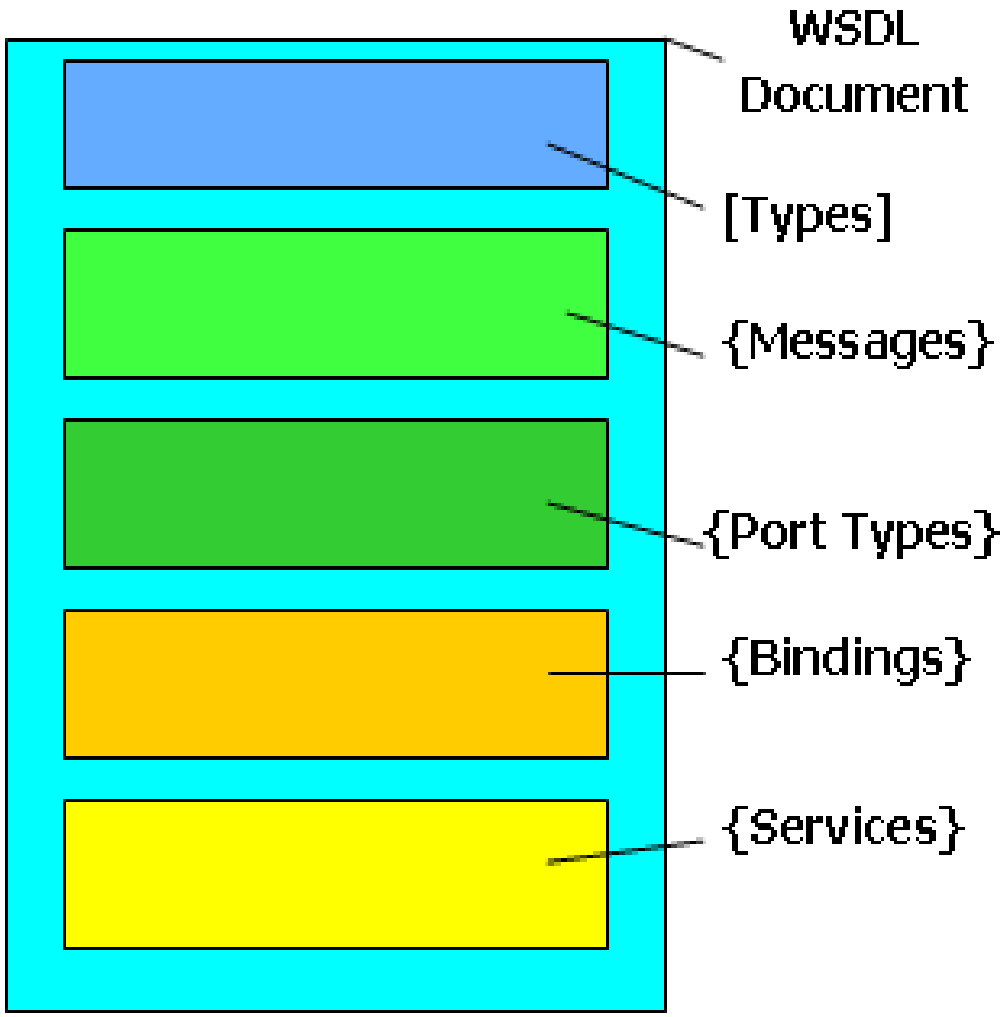



Estructura de un documento WSDL

# WSDL

- Para publicar un Servicio Web deberemos publicar su contrato, WSDL.
  - El WSDL se puede publicar en un servidor UDDI
  - Otros desarrolladores pueden utilizar el contrato para desarrollar clientes Web.
  - **Normalmente procesan el fichero WSDL a través de una herramienta que genera clases wrapper**.

# Estructura del documento



# Partes del documento

- **definitions**: elemento raíz del documento. Se utiliza para definir los espacios de nombres.

- **types**: Describe los tipos utilizados en los mensajes.
  - Viene a equivaler a las clases java, lo que son tipos primitivos se definen en los espacios de nombres.
  - Cuando definimos un **bean** vendrá declarado en esta parte. Además se mostrarán sus atributos de forma desglosada en XML-Schema.
  - El elemento types es Opcional.

  - El resto de partes pueden llevar o no documentación (es opcional).

# Partes de documento

- **message**: Define los distintos mensajes que se intercambiaran durante el proceso de invocación del servicio.

- Definir los mensajes de entrada y salida para cada operación que ofrezca el servicio.

- Los mensajes muestran descripciones abstractas de los datos que se van a intercambiar.

# Partes del documento

- **portType**: Colección de “**operations**” o “signatures”que definen el intercambio ordenado de los mensajes.
  - Se define cada operación y el formato de la entrada y la salida.
  - **OPERACIONES SOPORTADAS POR EL SERVICIO**

# Partes del documento

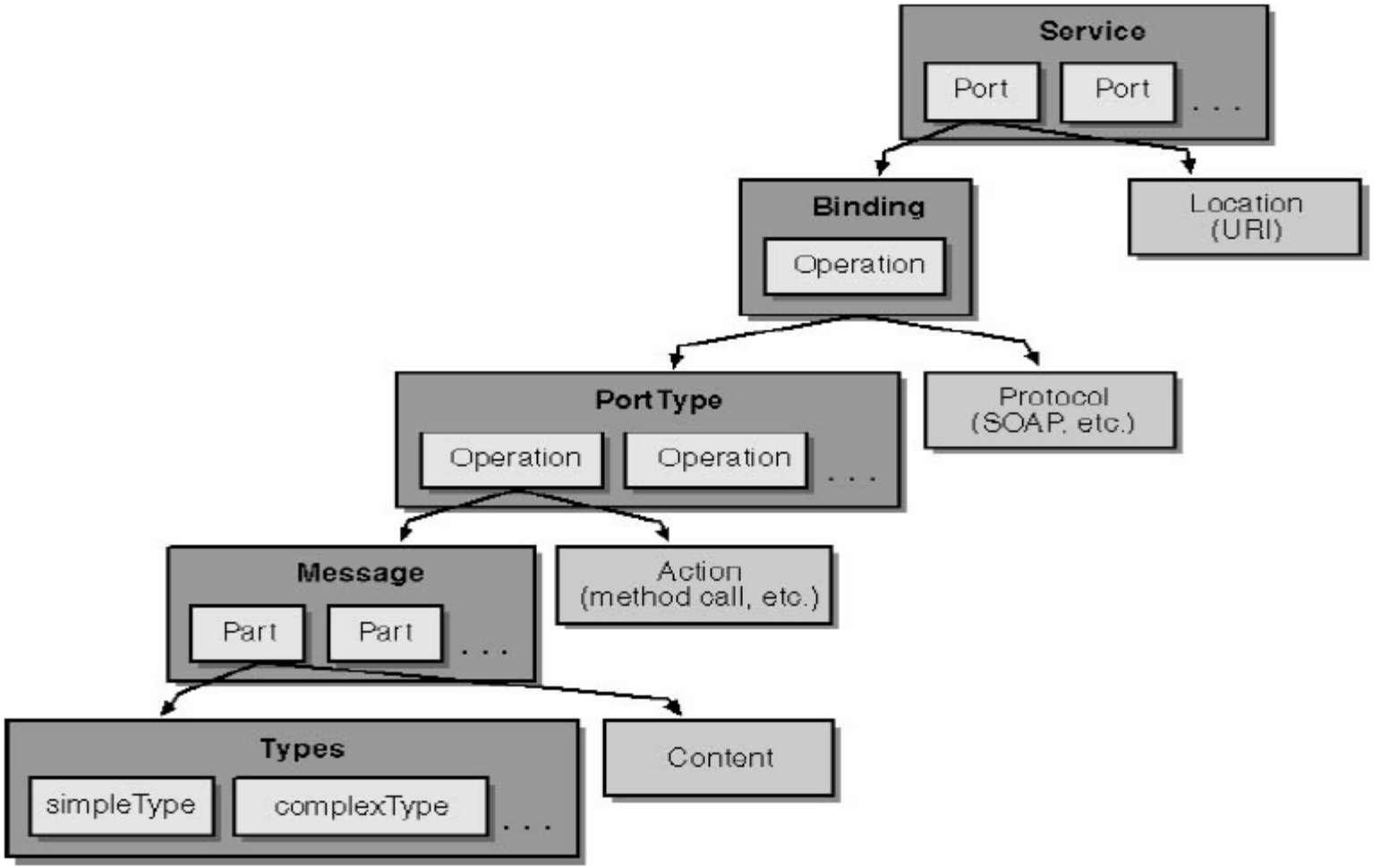
- **binding**: Especifica los protocolos de RED que utiliza cada port y especifica el formato de los datos para las operaciones definidas en portType.
  - Las operaciones pueden estar orientadas al documento o a RPC.
    - RPC: Remote Procedure Call
      - Llamada a un método con sus parámetros. Mas restrictivo.
    - Document
      - El contenido del mensaje puede ser XML, o datos binarios.
  - **SON LAS DEFINICIONES CONCRETAS DE LOS PORTTYPE**

# Partes del documento

- **service**: colección de **ports a los que se puede acceder** relacionados que especifican el servicio.
  - Cada puerto asocia una dirección de red con un binding.
  - La dirección de red es la URL donde el servicio actúan y es la dirección que tienen que utilizar las aplicaciones para conectarse al servicio.

  - **port**: Un elemento port define un extremo individual especificando una dirección única para una asociación.

# Componentes del Servicio



# Otras partes del documento

- **documentation**: Se utiliza para añadir comentarios a las definiciones.

- **import**: Para importar otros documentos dentro del actual incorporando otro espacio de nombres.

- **include**: Igual que import pero con el mismo espacio de nombres.

# Definitions

Definición de los espacios de nombres.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:ns1="http://org.apache.axis2/xsd" xmlns:ns="http://ws.ejemplo"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
```

# Types

```
- <wsdl:types>
  - <xs:schema attributeFormDefault="qualified" elementFormDefault="qualified"
      targetNamespace="http://ws.ejemplo">
    - <xs:element name="resuelve">
      - <xs:complexType>
        - <xs:sequence>
            <xs:element minOccurs="0" name="a" type="xs:double" />
            <xs:element minOccurs="0" name="b" type="xs:double" />
            <xs:element minOccurs="0" name="c" type="xs:double" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    - <xs:element name="resuelveResponse">
      - <xs:complexType>
        - <xs:sequence>
            <xs:element minOccurs="0" name="return" nillable="true" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
```

Definición de tipos

# Messages

```
- <wsdl:message name="resuelveRequest">
    <wsdl:part name="parameters" element="ns:resuelve" />
  </wsdl:message>
- <wsdl:message name="resuelveResponse">
    <wsdl:part name="parameters" element="ns:resuelveResponse" />
  </wsdl:message>
```

# Port Type

```
- <wsdl:portType name="EcuacionServicePortType">
  - <wsdl:operation name="resuelve">
      <wsdl:input message="ns:resuelveRequest" wsaw:Action="urn:resuelve" />
      <wsdl:output message="ns:resuelveResponse" wsaw:Action="urn:resuelveResponse" />
    </wsdl:operation>
  </wsdl:portType>
```

# Binding

```
- <wsdl:binding name="EcuacionServiceSoap11Binding" type="ns:EcuacionServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  - <wsdl:operation name="resuelve">
      <soap:operation soapAction="urn:resuelve" style="document" />
    - <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
    - <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

Definición de los protocolos:

Soap 1.1, también habrá otra para Soap 1.2 y http.

# Service

```
- <wsdl:service name="EcuacionService">
  - <wsdl:port name="EcuacionServiceHttpSoap11Endpoint" binding="ns:EcuacionServiceSoap11Binding">
      <soap:address
        location="http://localhost:8082/axis2/services/EcuacionService.EcuacionServiceHttpSoap11Endpoint/" />
    </wsdl:port>
  - <wsdl:port name="EcuacionServiceHttpSoap12Endpoint" binding="ns:EcuacionServiceSoap12Binding">
      <soap12:address
        location="http://localhost:8082/axis2/services/EcuacionService.EcuacionServiceHttpSoap12Endpoint/" />
    </wsdl:port>
  - <wsdl:port name="EcuacionServiceHttpEndpoint" binding="ns:EcuacionServiceHttpBinding">
      <http:address
        location="http://localhost:8082/axis2/services/EcuacionService.EcuacionServiceHttpEndpoint/" />
    </wsdl:port>
  </wsdl:service>
```

Define un port por cada tipo de enlace.

# Ejemplo llamada SOAP

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:soap='http://schemas.xmlsoap.org/soap/
   envelope/' xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
   soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
   <soap:Body>
      <n:sayHello xmlns:n='urn:examples:helloservice'>
         <firstName xsi:type='xsd:string'>World</firstName>
      </n:sayHello>
   </soap:Body>
</soap:Envelope>
```

# Lista de espacios de nombres

| Prefijo | Espacio de nombre URI | Definición |
|---|---|---|
| **wsdl** | http://schemas.xmlsoap.org/wsdl | Espacio de nombre WSDL para el marco WSDL. |
| **soap** | http://schemas.xmlsoap.org/wsdl/soap | Espacio de nombre WSDL para enlaces WSDL SOAP. |
| **http** | http://schemas.xmlsoap.org/wsdl/http | Espacio de nombre WSDL para WSDL HTTP GET y enlaces POST. |
| **mime** | http://schemas.xmlsoap.org/org/wsdl/mime | Espacio de nombre para enlaces WSDL MIME. |
| **soapenc** | http://schemas.xmlsoap.org/soap/encoding | Codificación de espacio de nombre dentro de la definición de SOAP 1.1. |
| **soapenv** | http://schemas.xmlsoap.org/soap/envelope | Sobre de espacio de nombre dentro de la definición SOAP 1.1. |
| **xsi** | http://www.w3.org/2001/XML-Schema-instance | Instancia de espacio de nombre dentro de la definición XSD. |
| **xsd** | http://www.w3.org/2001/XMLSchema | Esquema de espacio de nombre dentro de la definición XSD. |
| **tns** | El documento de espacio de nombres actual. | El prefijo (tns) “this namespace” se utiliza como convención para referirse al documento actual. |

# Enlaces

- **Web Services en la Red:**
  - https://geoservices.tamu.edu/Services/Geocode/WebService/v04_01/Simple/SOAP/
  - https://geoservices.tamu.edu/Services/Geocode/WebService/GeocoderService_V04_01.asmx

- Otros
  - https://stackoverflow.com/questions/115316/how-can-i-consume-a-wsdl-soap-web-service-in-python