

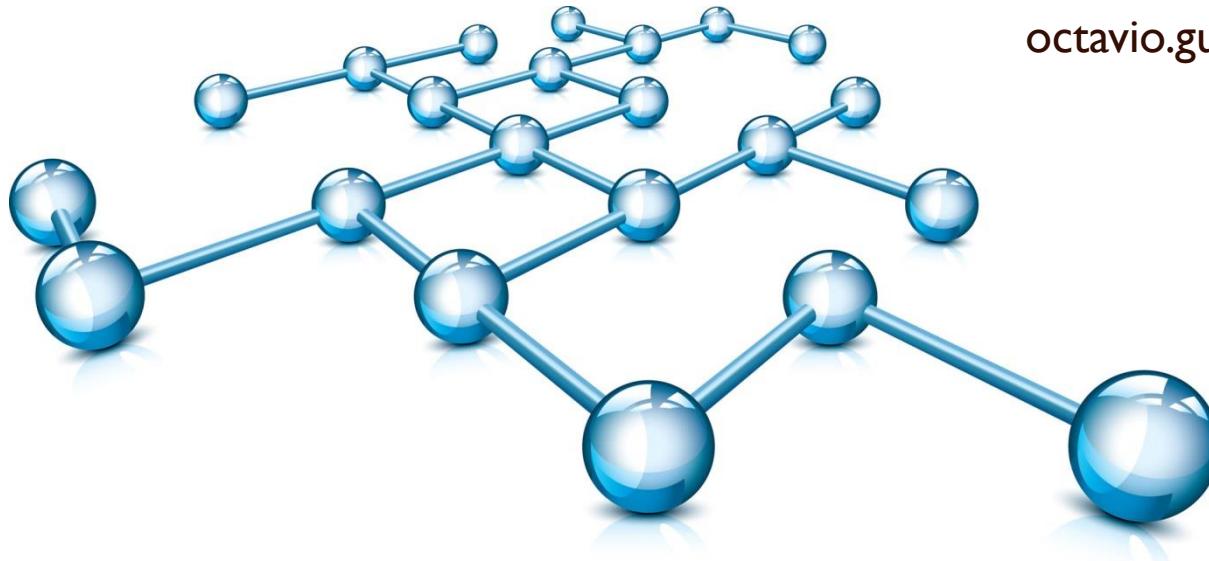


Servicios Web SOAP

Profesor:

Dr. J. Octavio Gutiérrez García

octavio.gutierrez@itam.mx



Problemas de aplicaciones Web

Diversas tecnologías:

Applets,

CGI (Common Gateway Interface),

Lenguajes de Scripts,

COM (Component object model), etc.

Desarrollos **muy ad-hoc**



Servicios Web



- **Idea:** Adaptar el modelo de programación web (**débilmente acoplado**) para su uso en aplicaciones **no** basadas en navegador



- **Objetivo:** ofrecer una plataforma para construir aplicaciones distribuidas utilizando un software que **enmascare la heterogeneidad**.

Servicios Web



- Estandarización controlada por un grupo del W3C



- Definición de W3C

“a software system designed to support interoperable machine-to-machine interaction over a network”

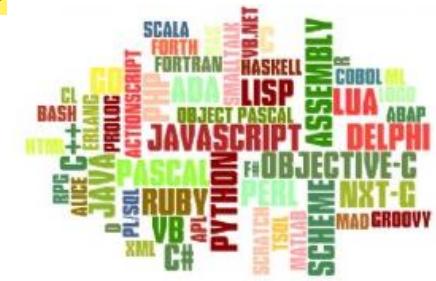


Servicios Web

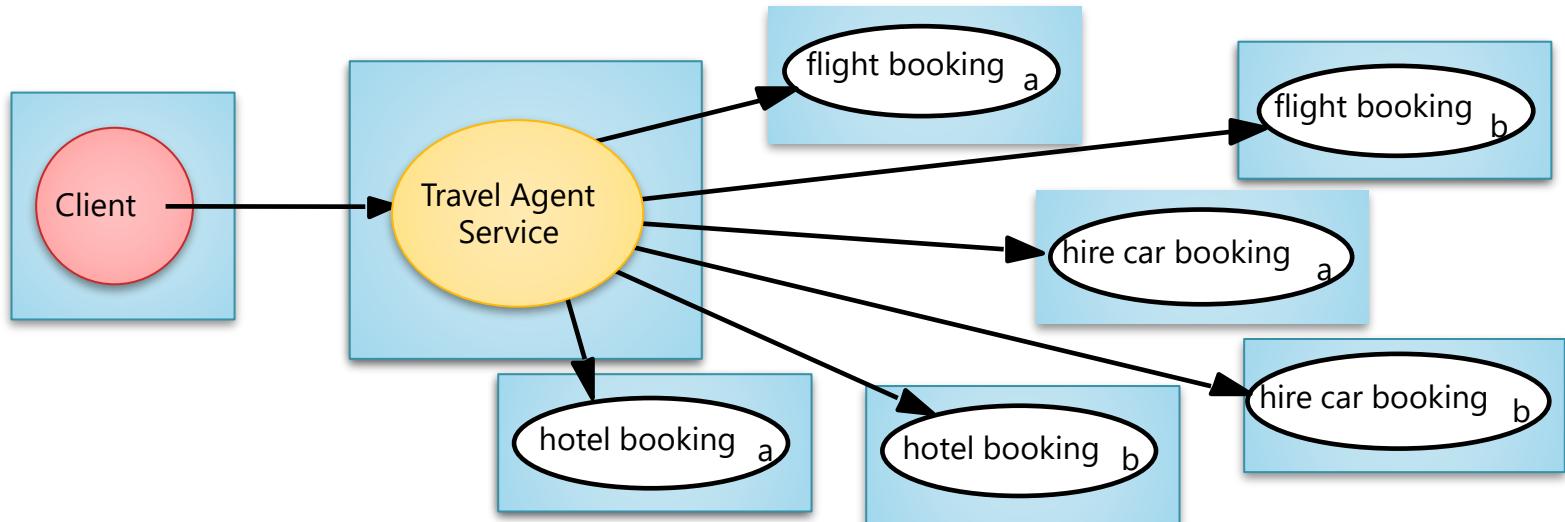


Un **servicio web** es una **colección** de **protocolos** y **estándares** abiertos que sirven para **intercambiar datos** entre aplicaciones

- Escritos en **distintos lenguajes de programación**
- Ejecutan en **distintos sistemas operativos y arquitecturas**
- Desarrollados de manera **independiente**
- **Independientes** de la **aplicación** que los usa



Servicios Web



Comunicación asíncrona y síncrona

Servicios Web



- **Ventajas:**

- **Interoperabilidad** entre aplicaciones.
- **Independencia** entre el **servicio web** y el **cliente**
- **Uso de estándares**
- Al ejecutar “comúnmente” HTTP, pueden **atravesar firewalls** sin necesidad de cambiar las reglas de filtrado.

- **Desventajas:**

- **Bajo rendimiento** comparado con otros modelos de computación distribuida: RMI o Corba.
- Pueden **esquivar firewalls**



STANDARDS

Aplicaciones

Servicios de directorio

Seguridad

Web Services

WSDL

SOAP

URLs

XML

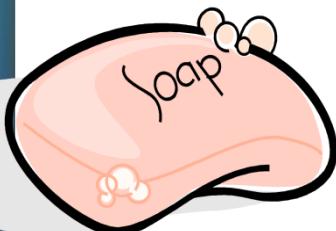
HTTP, SMTP u otros



Interoperabilidad en entornos heterogéneos

- Servicios basados en protocolos abiertos y mecanismos estándar

- **HTTP** 
- **Simple Object Access Protocol - SOAP:** Empaque la información y la transmite entre el cliente y el proveedor del servicio





Interoperabilidad en entornos heterogéneos

- Servicios basados en protocolos abiertos y mecanismos estándar



- Extensible Markup Language - XML: Representación externa de los datos y mensajes

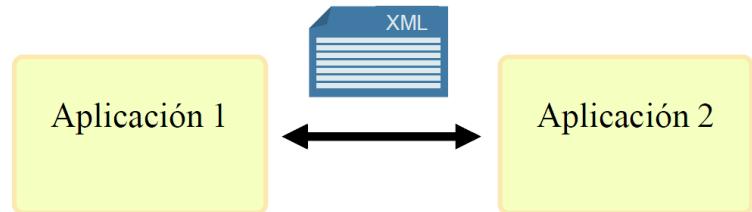


- Universal Description, Discovery and Integration - UDDI: Lista de servicios disponibles



- Web Service Definition Language - WSDL: Descripción del servicio

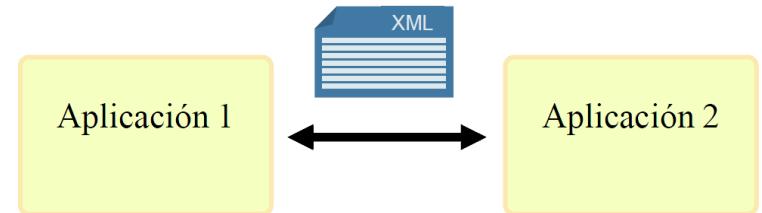
SOAP



- **Simple Object Access Protocol** permite intercambiar **mensajes** basados en **XML** sobre redes de computadoras
- Define un **esquema** para:
 - Usa **XML** para representar el contenido de mensajes
- SOAP (versión 1.2) **usa**:
 - **HTTP, SMTP, TCP o UDP**

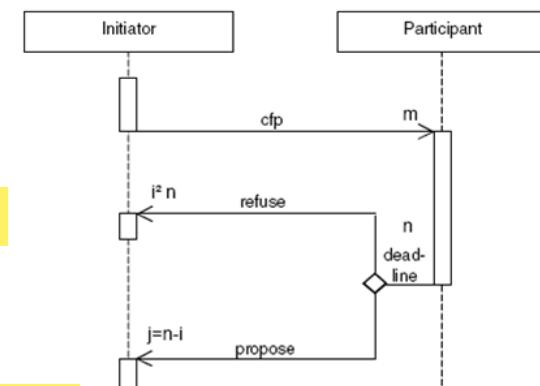


SOAP



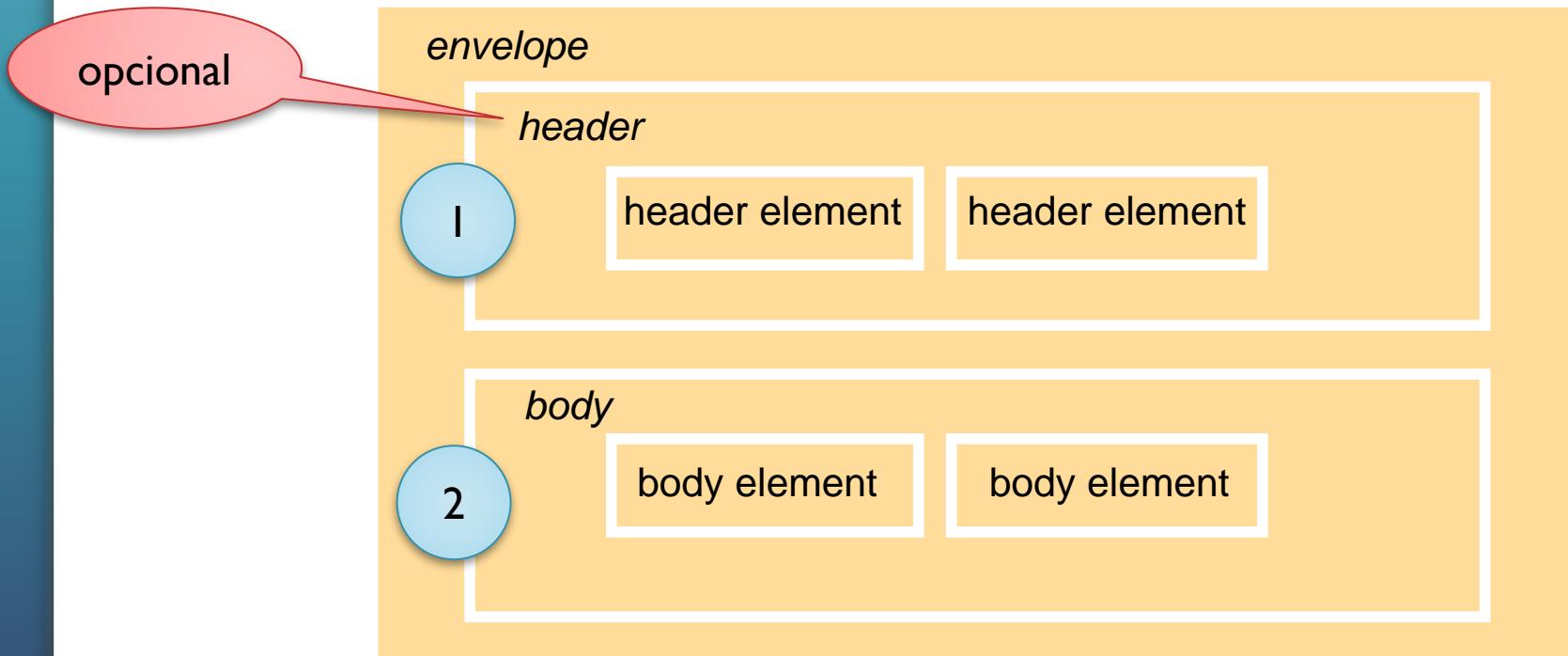
- SOAP especifica:

- Cómo representar los mensajes de texto en XML
- Cómo procesar los elementos de los mensajes
- Cómo se puede combinar un par de mensajes para reproducir un modelo petición-respuesta
- Cómo utilizar el protocolo de aplicación (HTTP, SMTP, ...) para enviar mensajes SOAP



SOAP

Un **mensaje SOAP** es transportado **en un sobre** (envelope)



Mensaje



```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        ...
    </soap:Header>

    <soap:Body>
        ...
        <soap:Fault>
            ...
        </soap:Fault>
    </soap:Body>

</soap:Envelope>
```

Sobre

Cabecera

Cuerpo

Sobre (Envelope)

Define el documento XML como un
mensaje SOAP

Obligatorio

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    ...
    Message information goes here
    ...
</soap:Envelope>
```

EncodingStyle

define los tipos de
datos usados en el
documento.



Mensaje SOAP

Estilo **RPC** y **literal**

FORMAT

```
<soap:envelope>
  <soap:body>
    <myMethod>
      <x>5</x>
      <y>5.0</y>
    </myMethod>
  </soap:body>
</soap:envelope>
```

Mensaje SOAP

Estilo *RPC* y *encoding*



FORMAT

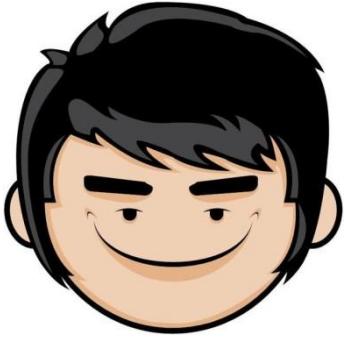
```
<soap:envelope>
  <soap:body>
    <myMethod>
      <x xsi:type="xsd:int">5</x>
      <y xsi:type="xsd:float">5.0</y>
    </myMethod>
  </soap:body>
</soap:envelope>
```

Mensaje SOAP

Estilo ***document*** y ***literal***



```
<soap:envelope>
  <soap:body>
    <x>5</x>
    <y>5.0</y>
  </soap:body>
</soap:envelope>
```



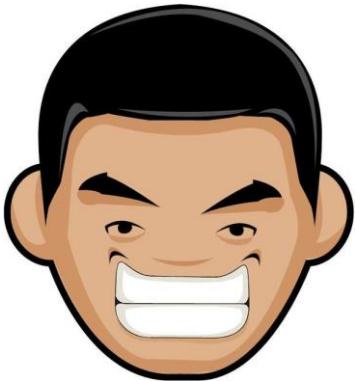
Encabezado (Header)

- Elemento **opcional**
- Incluye **información de control**
 - **Identificador de transacción** para su uso con un servicio de transacciones
 - Un **identificador de mensajes** para relacionar mensajes entre sí
 - Un nombre de usuario, una clave pública, **etc.**



Botella 1

Botella 2



Encabezado (Header)

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Header>
    <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
      soap:mustUnderstand="1">234
    </m:Trans>
  </soap:Header>
  ...
</soap:Envelope>
```

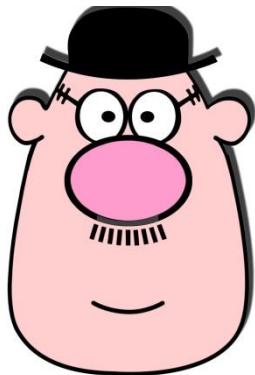


Encabezado (Header)

Etiqueta XML
definida por el
usuario (aplicación)

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
            soap:mustUnderstand="1">234
        </m:Trans>
    </soap:Header>
    ...
</soap:Envelope>
```



Encabezado (Header)

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
            soap:mustUnderstand="1">234
        </m:Trans>
    </soap:Header>
    ...
</soap:Envelope>
```

Si el valor es 1, el
elemento header
DEBE ser procesado

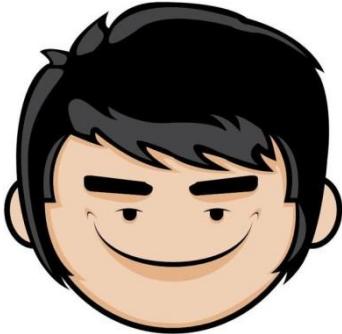


Encabezado (Header)

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Header>
    <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
      soap:mustUnderstand="1">234
    </m:Trans>
  </soap:Header>
  ...
</soap:Envelope>
```

Valor de la etiqueta
XML trans. Ejemplo:
Transacción 234



Encabezado (Header)

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
            soap:actor="http://www.w3schools.com/appml/">234
        </m:Trans>
    </soap:Header>
    ...
</soap:Envelope>
```

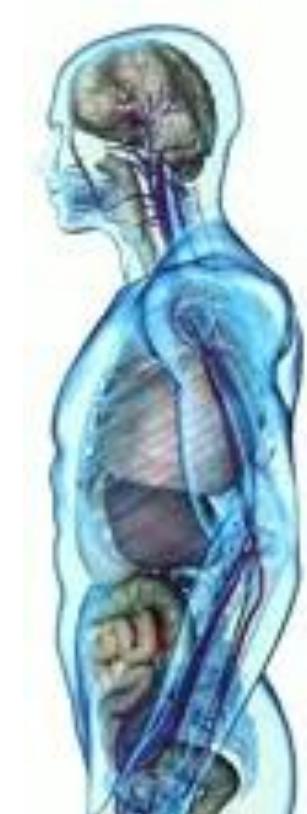
Si el mensaje SOAP pasa por varios endpoints, sólo el endpoint indicado en el elemento **actor** debe de procesar el encabezado

A pink rounded rectangle containing the word "SOAP" in white, with a dark brown border.

SOAP

Cuerpo (Body)

- Elemento **obligatorio**
- En el elemento body se incluye:
 - Mensaje
 - Referencia al esquema XML que describe el servicio
- Comunicación cliente-servidor
 - El elemento *body* contiene una petición o una respuesta





SOAP

Cuerpo (Body)

PETICIÓN

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
    </m:GetPrice> ←
  </soap:Body>
</soap:Envelope>
```



Elemento específico
de la aplicación





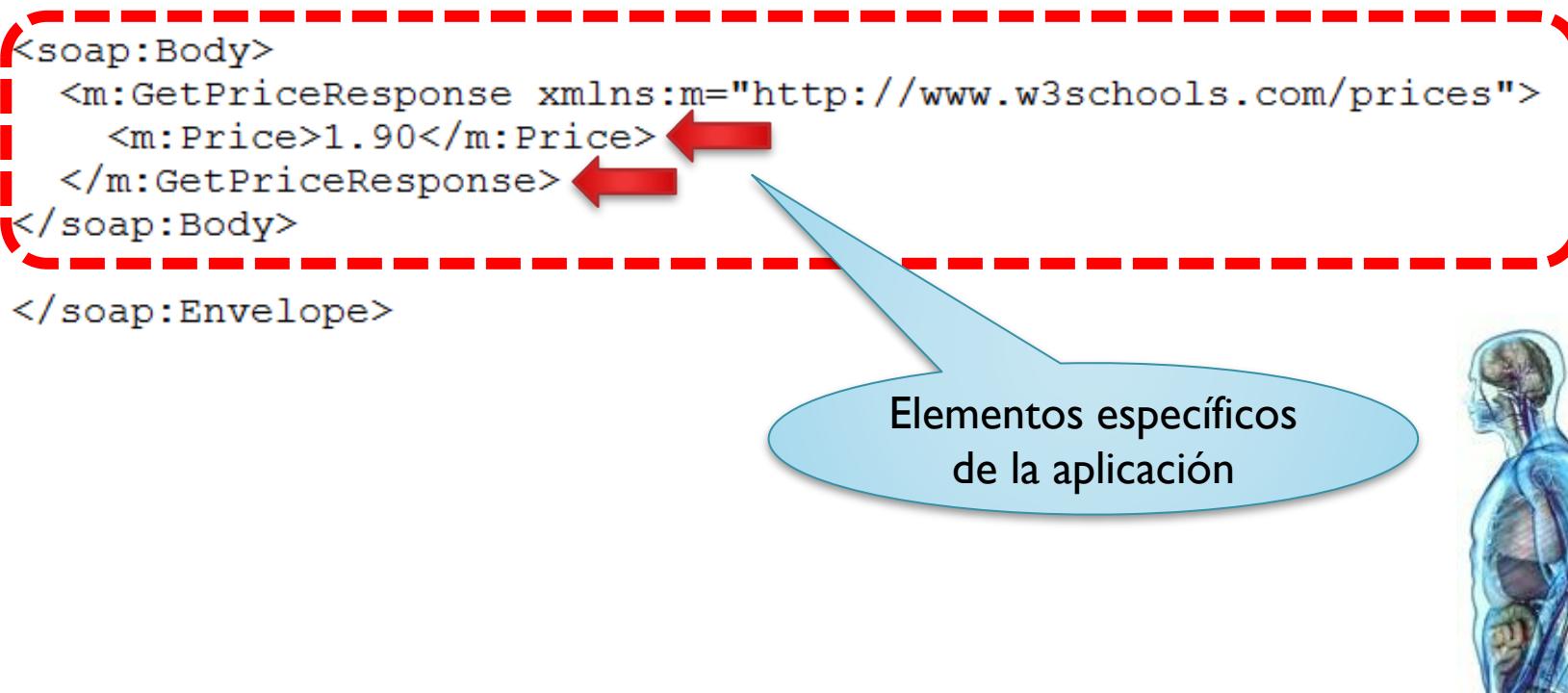
SOAP

Cuerpo (Body)

RESPUESTA

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body>
    <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
      <m:Price>1.90</m:Price>
    </m:GetPriceResponse>
  </soap:Body>
</soap:Envelope>
```



The diagram illustrates the SOAP message structure. A red dashed box encloses the `<soap:Body>` and `<m:GetPriceResponse>` elements. Two red arrows point from these specific application-specific elements to a light blue callout bubble containing the text "Elementos específicos de la aplicación". A stylized human figure is partially visible on the right.

Elementos específicos
de la aplicación



SOAP & HTTP

- **HTTP + XML = SOAP**
- Un método SOAP es una solicitud/respuesta HTTP que cumple con las reglas de SOAP.



A pink rounded rectangle button with the word "SOAP" in white, bold, sans-serif font.

SOAP

Ejemplo de solicitud

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Body
        xmlns:m="http://www.example.org/stock">
        <m:GetStockPrice>
            <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
    </soap:Body>

</soap:Envelope>
```

A pink rounded rectangle button with the word "SOAP" in white, bold, sans-serif font.

SOAP

Ejemplo de respuesta

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/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>
```

HTTP Status Codes



1XX
INFORMATIONAL

2XX
SUCCESS

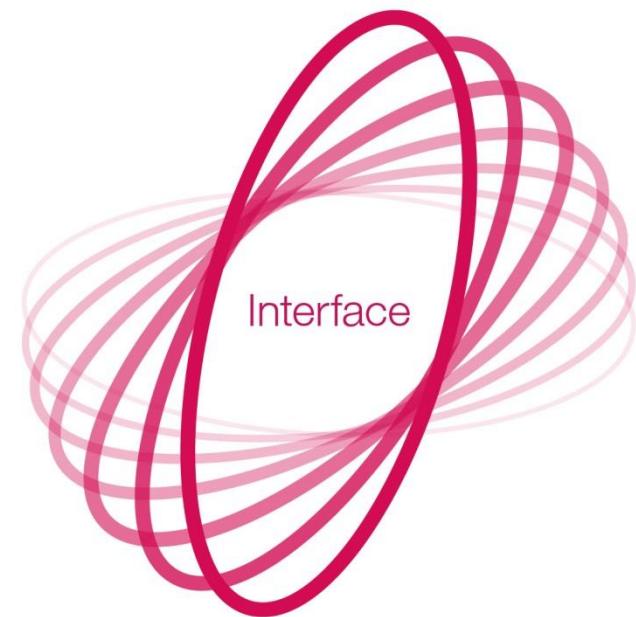
3XX
REDIRECTION

4XX
CLIENT ERROR

5XX
SERVER ERROR

Servicios Web – Interfaz

- Una **interfaz** de servicio web consta de un **conjunto de operaciones** que pueden ser accedidas por un cliente en Internet
- El **conjunto de operaciones** en un servicio web pueden ser **ofrecidas por programas, objetos**, bases de datos, etc.



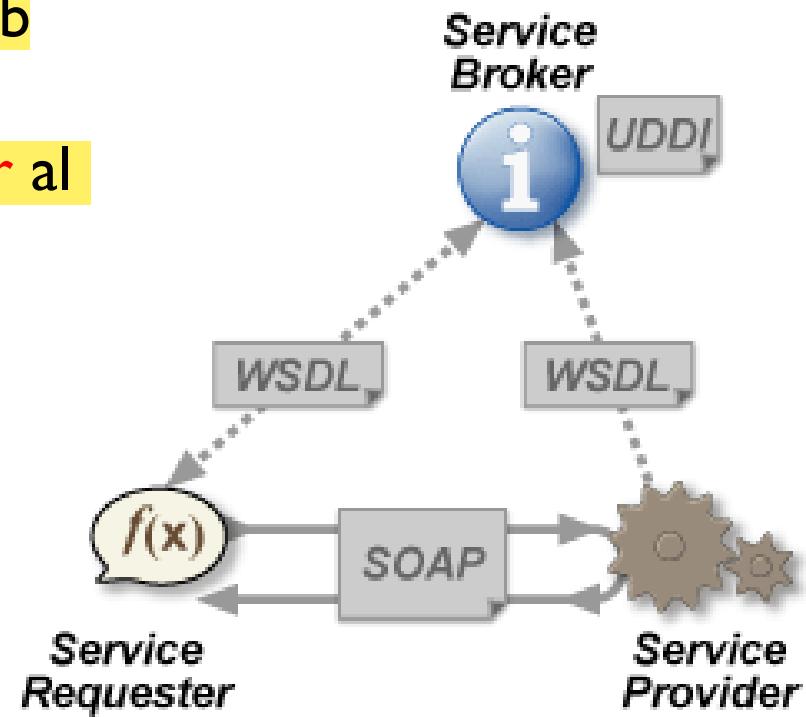
anything

WSDL

- Web Services Description Language es un lenguaje de descripción de interfaz (IDL) para servicios Web en XML

Describe al servicio web

Describe como acceder al servicio web



WSDL – Elementos

| | |
|-------------------------|--|
| <types> | Contenedor para definiciones de tipos de datos usados por el servicio web |
| <message> | Definición de los datos a ser comunicados |
| <portType> | Conjunto de operaciones soportadas por uno o mas puntos finales |
| <binding> | Define el formato del mensaje y el protocolo |

Estructura principal de un documento WSDL

<definitions>

<types>

Definiciones de tipos de datos....

</types>

<message>

Definición de los datos a ser comunicados...

</message>

<portType>

Conjunto de operaciones.....

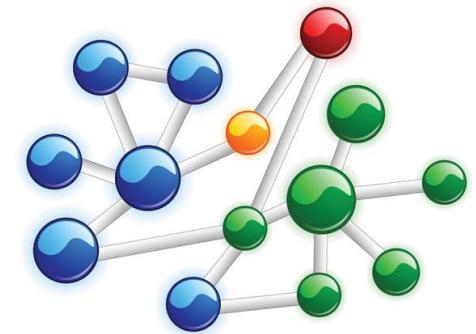
</portType>

<binding>

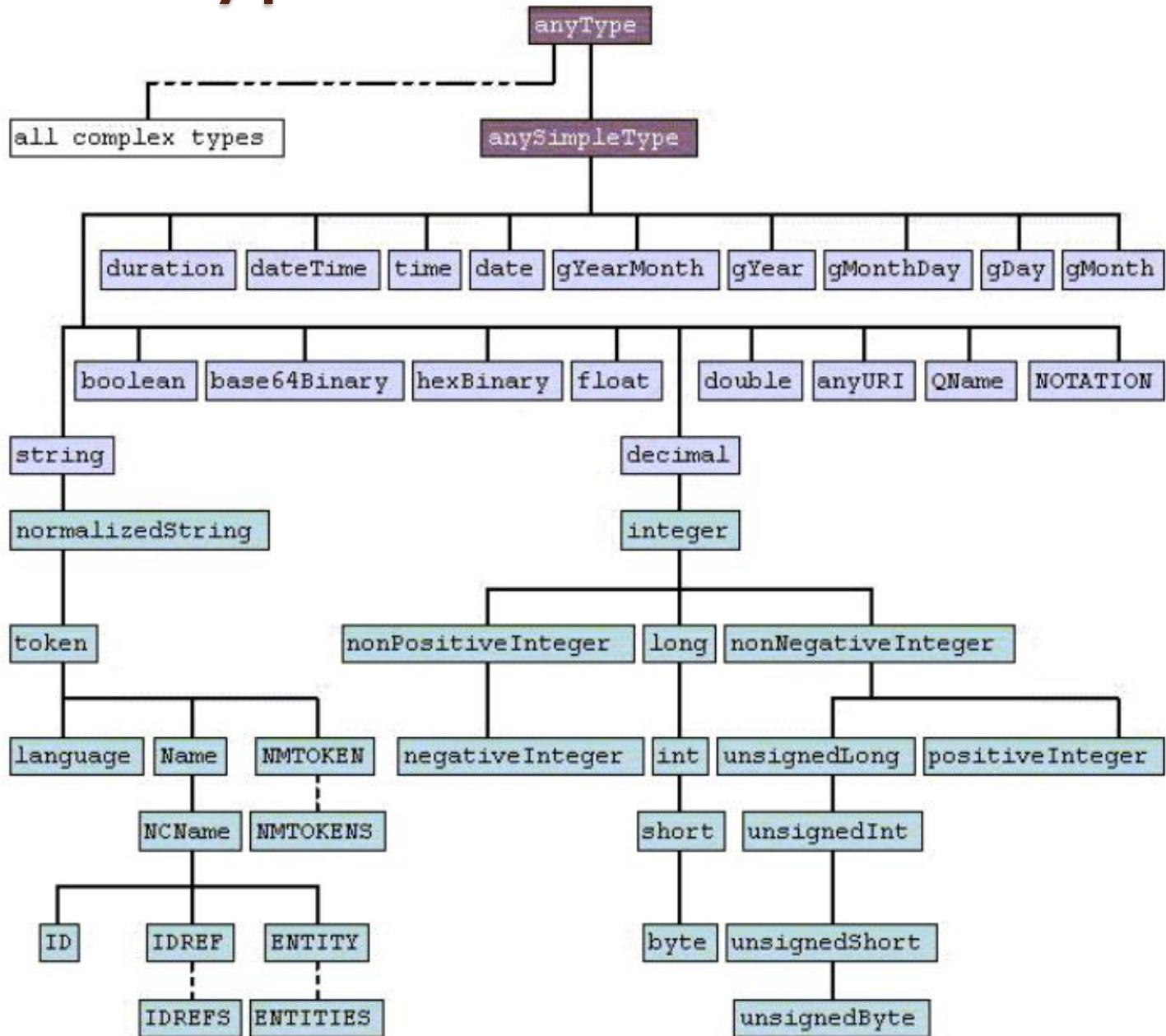
Protocolo y especificación del formato de datos....

</binding>

</definitions>



WSDL - Types



WSDL - Ejemplo

```
<message name="getTermRequest">  
  <part name="term" type="xs:string"/>  
</message>
```

```
<message name="getTermResponse">  
  <part name="value" type="xs:string"/>  
</message>
```

```
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>
```

Función

Param
entrada
y salida

WSDL – PortType [1/2]

Define las operaciones del servicio web y los mensajes que están involucrados.

- Tipos de operaciones



- Sólo ida (**One-way**)
 - El punto final recibe un mensaje



- Notificación (**Notification**)
 - El punto final envía un mensaje

WSDL – PortType [2/2]

Define las operaciones del servicio web y los mensajes que están involucrados.

- Tipos de operaciones



- Solicitud-respuesta (**Request-Response**)
 - El punto final **recibe un mensaje y envía un mensaje correlacionado.**



- Petición-respuesta (**Solicit-Response**)
 - El punto final **envía un mensaje y recibe un mensaje correlacionado.**

One-way

```
<message name="newTermValues">  
  <part name="term" type="xs:string"/>  
  <part name="value" type="xs:string"/>  
</message>
```

```
<portType name="glossaryTerms">
```

```
  <operation name="setTerm">
```

```
    <input name="newTerm" message="newTermValues"/>
```

```
  </operation>
```

```
</portType >
```



Notification



```
<message name="notifyNewVersion ">  
  <part name="version" type="xs:string"/>  
</message>
```

```
<portType name="glossaryTerms">  
  <operation name="newVersion">
```

```
    <output name="version" message="notifyNewVersion"/>
```

```
  </operation>
```

```
</portType >
```



Request-Response



1

```
<message name="getTermRequest">  
  <part name="term" type="xs:string"/>  
</message>
```

2

```
<message name="getTermResponse">  
  <part name="value" type="xs:string"/>  
</message>
```

```
<portType name="glossaryTerms">  
  <operation name="getTerm">
```

1 <input message="getTermRequest"/>

2 <output message="getTermResponse"/>

```
  </operation>  
</portType>
```

El servicio web
recibe un
mensaje y envía
un mensaje
correlacionado

Solicit-Response



1

```
<message name="getExpirationDateResponse">  
  <part name="value" type="xs:string"/>  
</message>
```

2

```
<message name="getExpirationDateSolicit">  
  <part name="term" type="xs:string"/>  
</message>
```

```
<portType name="glossaryTerms">  
  <operation name="getExpirationDate">
```

1

```
    <output message="getExpirationDateResponse"/>
```

2

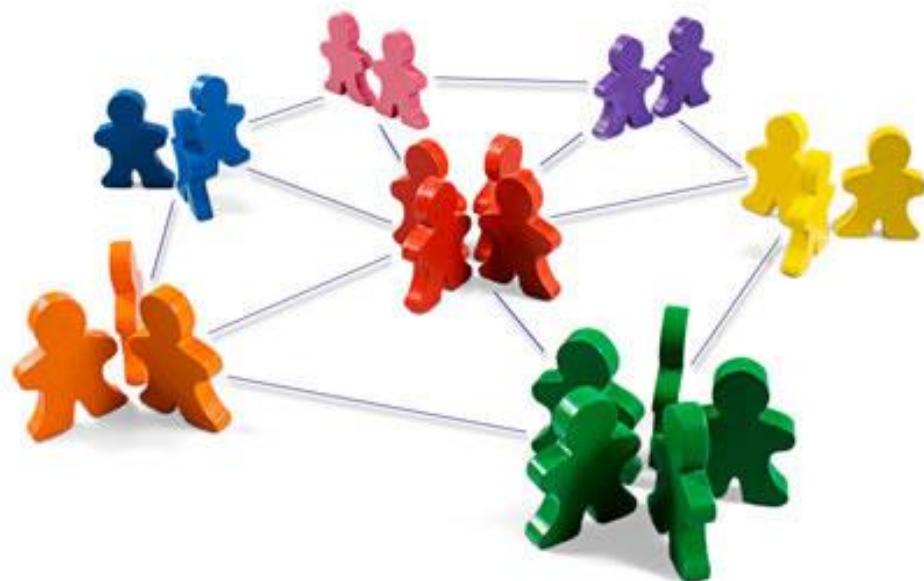
```
    <input message="getExpirationDateSolicit"/>
```

```
  </operation>  
</portType>
```

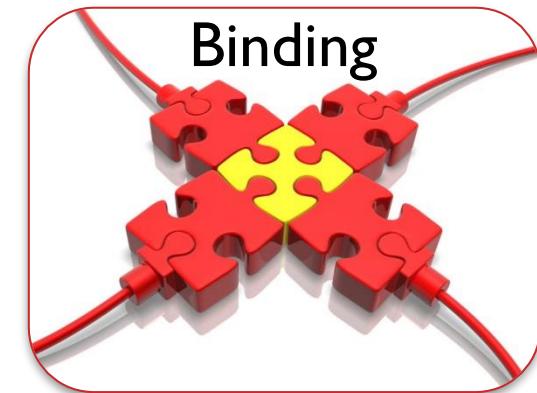
El servicio web
envía un mensaje
y recibe un
mensaje
correlacionado.

WSDL – Vinculación (Binding)

Define el formato del
mensaje y el protocolo



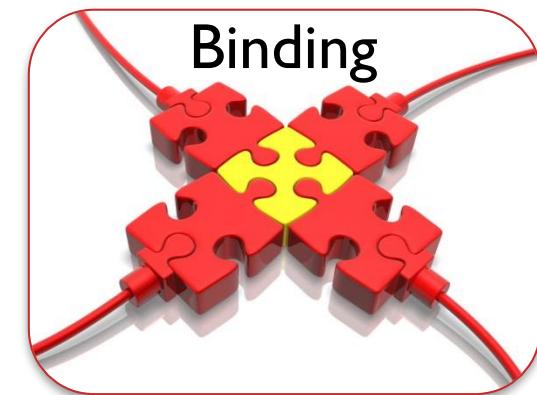
```
<message name="getTermRequest"> ... </message>  
  
<message name="getTermResponse" > ... </message>  
  
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>  
  
<binding type="glossaryTerms" name="MyBinding">  
  <soap:binding style="document"  
    transport="http://schemas.xmlsoap.org/soap/http" />  
  <operation>  
    <soap:operation soapAction="http://example.com/getTerm"/>  
    <input><soap:body use="literal"/></input>  
    <output><soap:body use="literal"/></output>  
  </operation>  
</binding>
```



```
<message name="getTermRequest"> ... </message>  
  
<message name="getTermResponse" > ... </message>
```

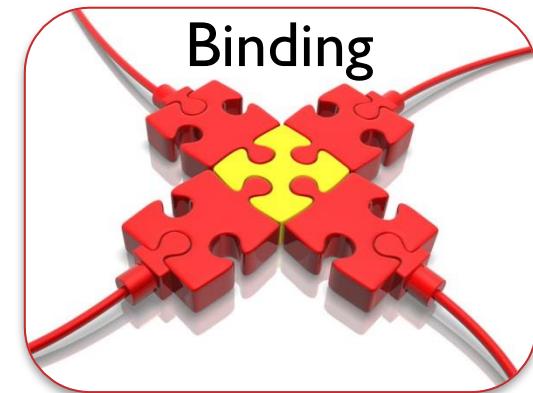
```
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>
```

```
<binding type="glossaryTerms" name="MyBinding">  
  <soap:binding style="document"  
    transport="http://schemas.xmlsoap.org/soap/http" />  
  <operation>  
    <soap:operation soapAction="http://example.com/getTerm"/>  
    <input><soap:body use="literal"/></input>  
    <output><soap:body use="literal"/></output>  
  </operation>  
</binding>
```



```
<message name="getTermRequest"> ... </message>  
  
<message name="getTermResponse" > ... </message>  
  
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>
```

```
<binding type="glossaryTerms" name="MyBinding">  
  <soap:binding style="document"  
    transport="http://schemas.xmlsoap.org/soap/http">  
    <operation>  
      <soap:operation soapAction="http://tempuri.org/IEdmService/getTerm" />  
      <input><soap:body use="literal" />  
      <output><soap:body use="literal" />  
    </operation>  
</binding>
```



El atributo STYLE se refiere a como traducir el binding a un mensaje SOAP.

Document: mensaje que contiene documentos.

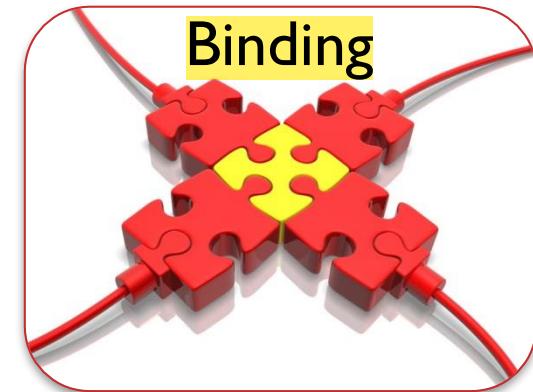
RPC: mensaje con parámetros y valores de retorno.

```
<message name="getTermRequest"> ... </message>
```

```
<message name="getTermResponse" > ... </message>
```

```
<portType name="glossaryTerms">  
    <operation name="getTerm">  
        <input message="getTermRequest"/>  
        <output message="getTermResponse"/>  
    </operation>  
</portType>
```

```
<binding type="glossaryTerms" name="MyBinding">  
    <soap:binding style="document"  
    transport="http://schemas.xmlsoap.org/soap/http" />  
    <operation>  
        <soap:operation soapAction="http://example.c  
        <input><soap:body use="literal"/></input>  
        <output><soap:body use="literal"/></output>  
    </operation>  
</binding>
```



Binding

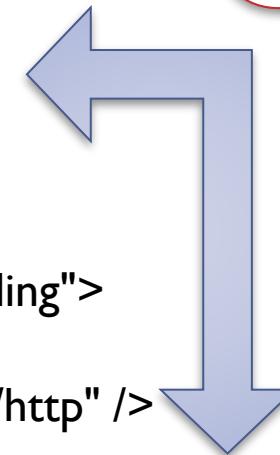
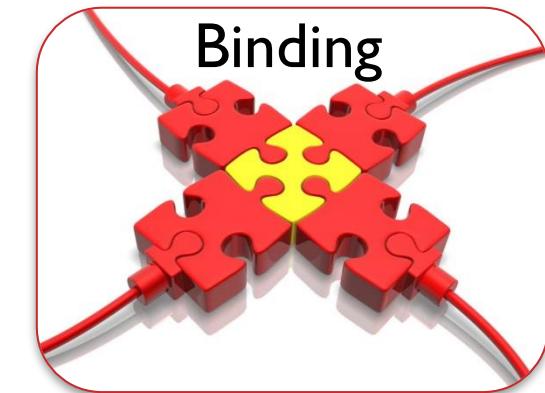
El atributo
TRANSPORT define el
protocolo a utilizar

```
<message name="getTermRequest"> ... </message>
```

```
<message name="getTermResponse" > ... </message>
```

```
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>
```

```
<binding type="glossaryTerms" name="MyBinding">  
  <soap:binding style="document"  
    transport="http://schemas.xmlsoap.org/soap/http" />  
  <operation>  
    <soap:operation soapAction="http://example.com/getTerm"/>  
    <input><soap:body use="literal"/></input>  
    <output><soap:body use="literal"/></output>  
  </operation>  
</binding>
```



encoded o literal

WSDL y UDDI

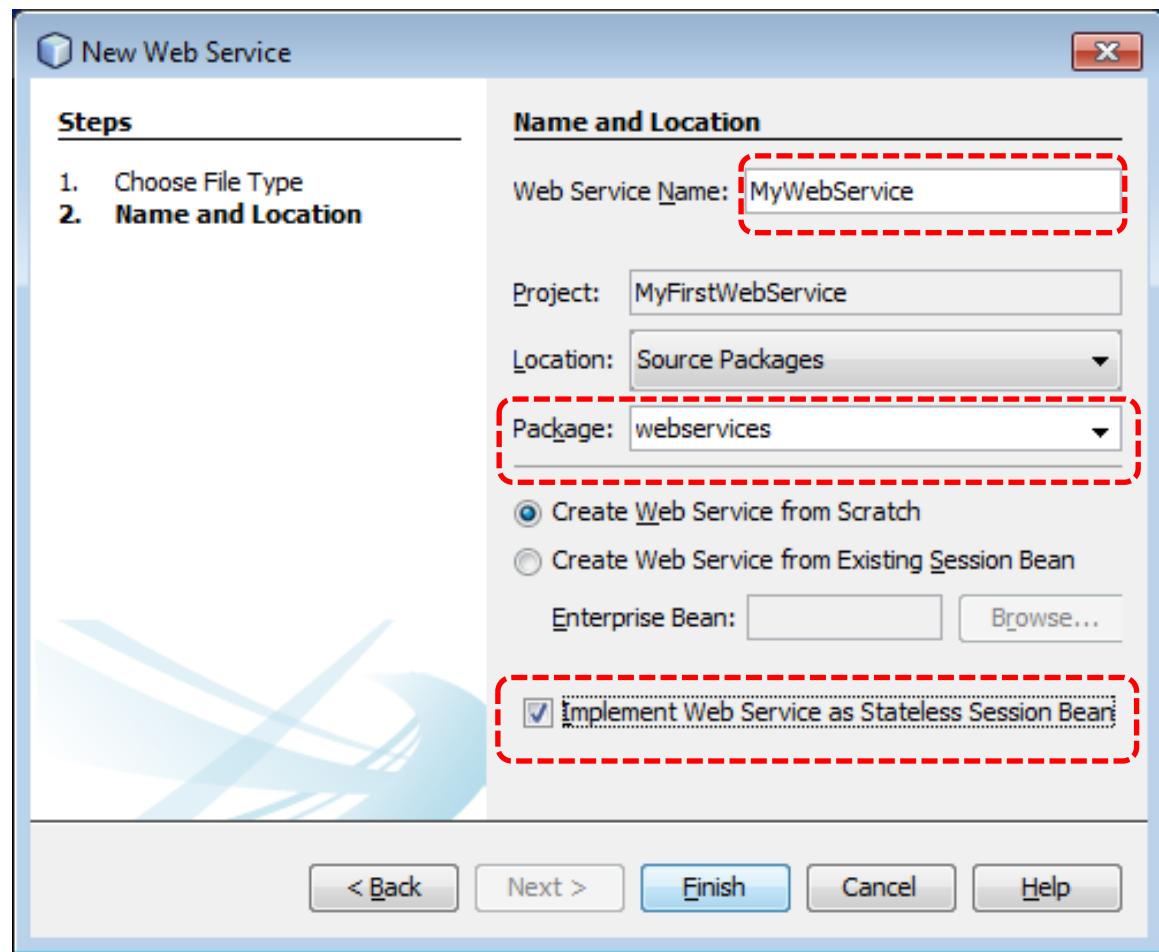


- **Universal Description, Discovery and Integration (UDDI)** es un servicio de directorio en el que se pueden registrar y buscar servicios Web.
- Directorio de interfaces WSDL
- Comunicación a través de SOAP
- Servicio de páginas blancas, amarillas y verdes.

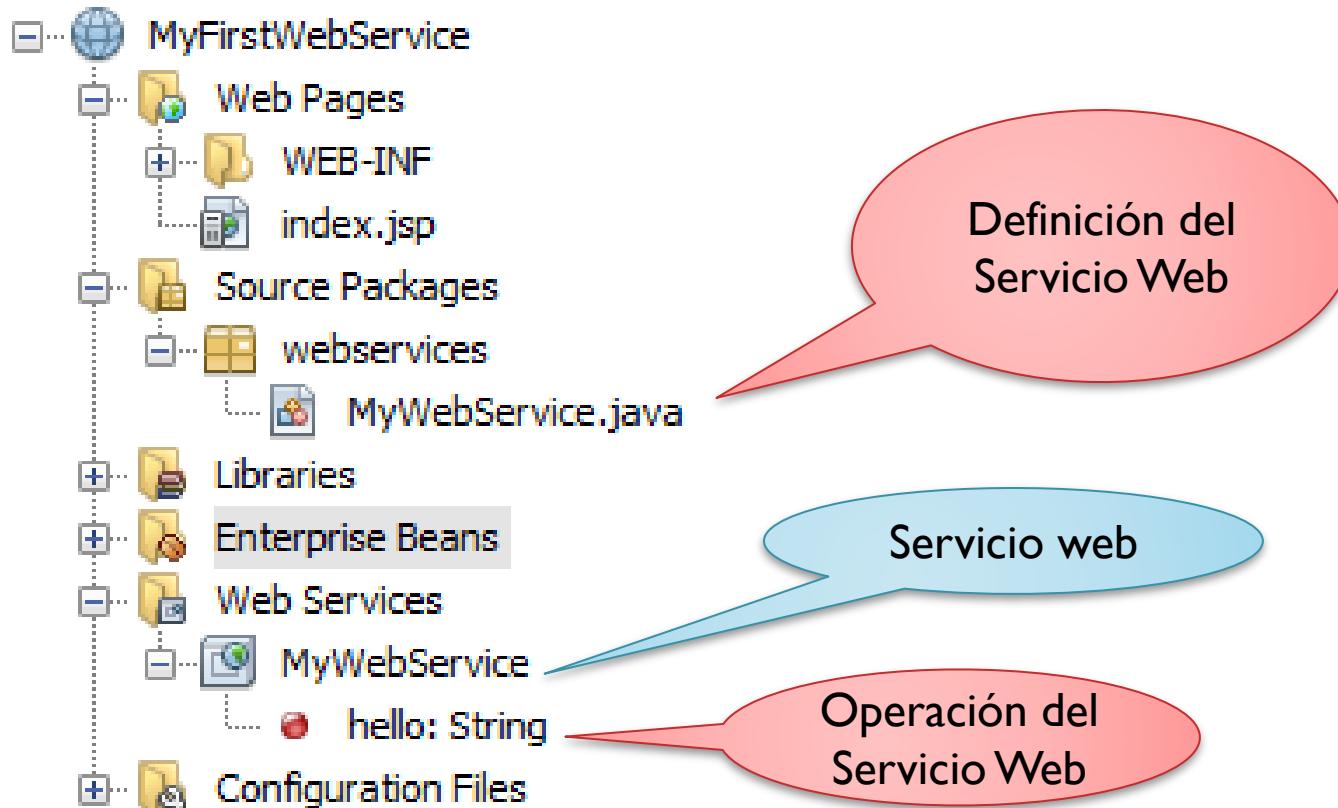


Servicios Web SOAP

Crear servicio web



Servicios Web SOAP



Servicios Web SOAP

- En vista de diseño del servicio web
 - Quitar operación **HELLO**
 - Agregar operación **ADD**



Operations (1) Add Operation... Remove Operation ▲

| | | | | |
|----------------|------------------|--------|-------------|--|
| hello | | | | |
| Parameters | Output | Faults | Description | |
| Parameter Name | Parameter Type | | | |
| name | java.lang.String | | | |

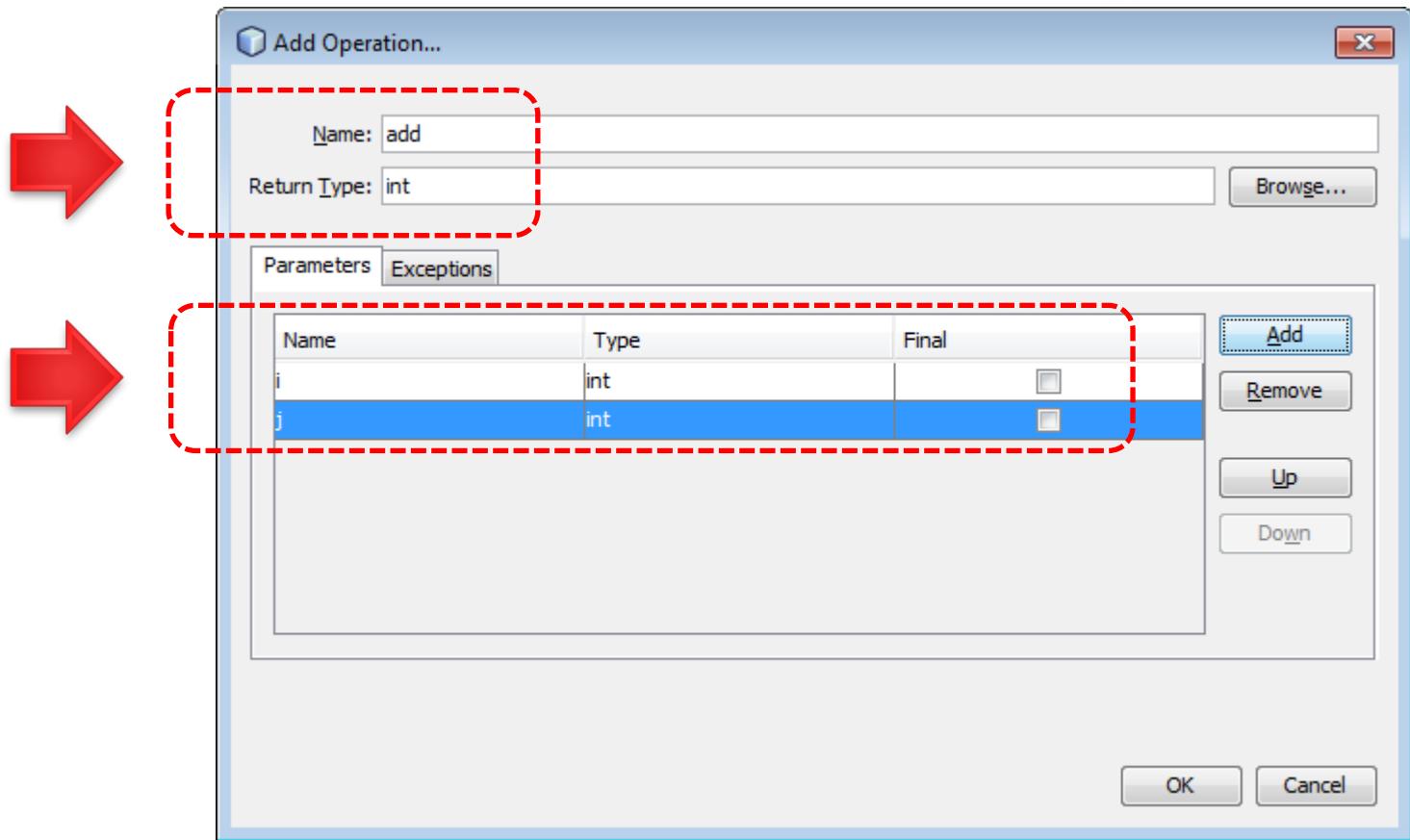
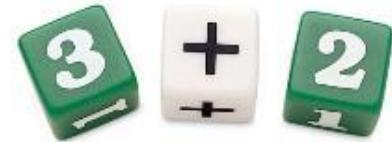
Quality Of Service ▲

- Optimize Transfer Of Binary Data (MTOM)
- Reliable Message Delivery
- Secure Service

Edit Web Service Attributes...

Servicios Web SOAP

- Definición de operación ADD



Servicios Web SOAP

- Vista de diseño del servicio web
 - Operación **ADD**

MyWebService

Operations (1) Add Operation... Remove Operation 

| Parameters | Output | Faults | Description |
|----------------|----------------|--------|-------------|
| Parameter Name | Parameter Type | | |
| i | int | | |
| j | int | | |

Quality Of Service

- Optimize Transfer Of Binary Data (MTOM)
- Reliable Message Delivery
- Secure Service

Edit Web Service Attributes...



Servicios Web SOAP

Vista de código fuente del servicio web Operación **ADD**

```
@WebService(serviceName = "MyWebService")
@Stateless()
public class MyWebService {

    @WebMethod(operationName = "add")

    public int add(@WebParam(name = "i") int i, @WebParam(name = "j") int j) {
        //TODO write your implementation code here:
        return 0;
    }
}
```

Servicios Web SOAP

2

MyWebService Web Service Tester

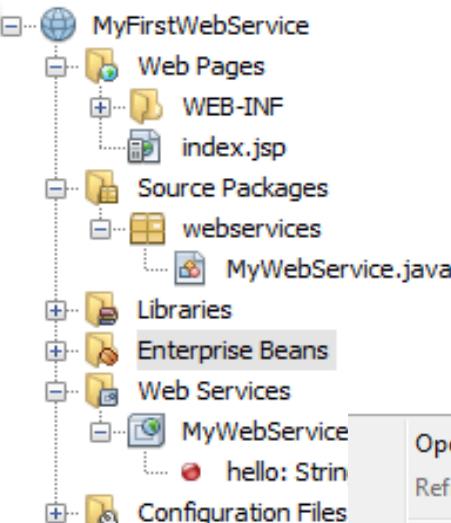
This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract int webservices.MyWebService.add(int,int)

add (3 , 4)



3

add Method invocation

Method parameter(s)

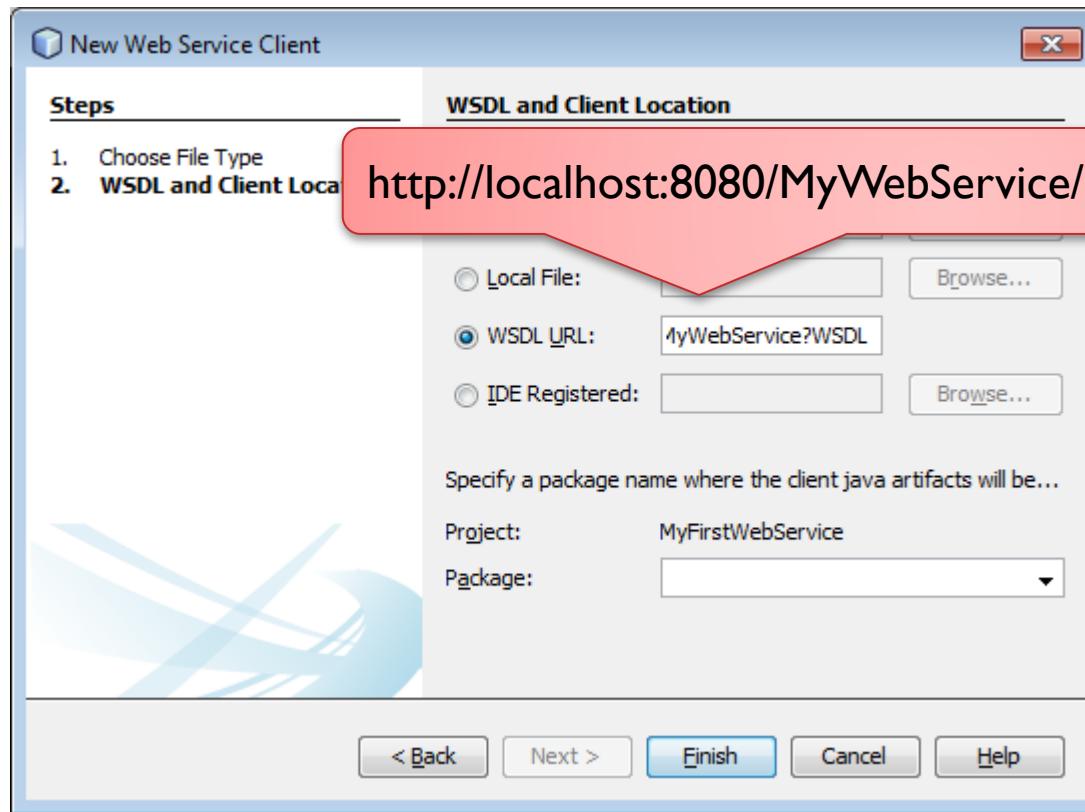
| Type | Value |
|------|-------|
| int | 3 |
| int | 4 |

Method returned

int : "7"

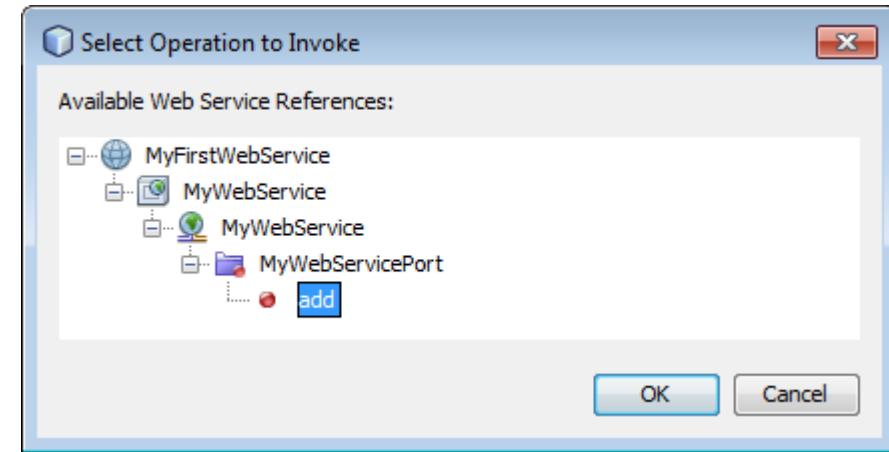
Servicios Web SOAP

• Crear cliente de servicio web desplegado



Servicios Web SOAP

```
public static void main(String[] args) {  
    System.out.println(add(3, 5));  
}
```



```
private static int add(int i, int j) {  
    webservices.MyWebService_Service service = new  
        webservices.MyWebService_Service();  
    webservices.MyWebService port = service.getMyWebServicePort();  
    return port.add(i, j);  
}
```

Aplicación
Java

Práctica de Laboratorio

Composición de Servicios Turísticos



Composición de Servicios Turísticos

Crea **cuatro** servicios web SOAP

1. Servicio “**Agencia**” principal que ofrezca una solución compuesta de:
 2. Servicio para reservar vuelos
 3. Servicio para reservar hotel
 4. Servicio para rentar carros



Composición de Servicios Turísticos

- *Servicio Vuelo*

- Dos operaciones:

- *creaReservacionVuelo*

- Entrada: (1) origen (String), (2) destino (String), (3) fecha inicial (String) y (4) fecha final (String)
- Salida: (1) Costo (double) e (2) identificador de reservación (String)

- *confirmaReservacionVuelo*

- Entrada: (1) Tarjeta de crédito (String) (2) identificador de reservación (String)

Salida: (1) Estado de la reserva (String) - Fallido o Exitoso

Se necesitará
un objeto
Reservacion



Composición de Servicios Turísticos

- **Servicio Hotel**

Se necesitará
un objeto
Reservacion

- **Dos** operaciones:

- **creaReservacionHotel**

- Entrada: (1) ciudad (String), (2) fecha inicial (String) y (3) fecha final (String)
- Salida: (1) Costo (double) e (2) identificador de reservación (String)

- **confirmaReservacionHotel**

- Entrada: (1) Tarjeta de crédito (String) (2) identificador de reservación (String)
- Salida: (1) Estatus de operación: Éxito o Falla.

Notas de Implementación:

El primer método puede regresar un costo y un identificador aleatorio. El segundo método puede siempre regresar Éxito



Composición de Servicios Turísticos

- **Servicio Carro**

Se necesitará
un objeto
Reservacion

- **Dos** operaciones:

- **creaReservacionCarro**

- Entrada: (1) ciudad (String), (2) fecha inicial (String) y (3) fecha final (String)
- Salida: (1) Costo (double) e (2) identificador de reservación (String)

- **confirmaReservacionCarro**

- Entrada: (1) Tarjeta de crédito (String) (2) identificador de reservación (String)
- Salida: (1) Estatus de operación: Éxito o Falla.

Notas de Implementación:

El primer método puede regresar un costo y un identificador aleatorio. El segundo método puede siempre regresar Éxito



Composición de Servicios Turísticos

- *Servicio Agencia*

Se necesitará
un objeto
Reservacion

- Dos operaciones:

- **creaReservacion**

- Entrada: (1) origen (String), (2) destino (String), (3) fecha inicial (String) y (4) fecha final (String)
- Salida: (1) Costo (double) e (2) identificador de reservación (String)

- **confirmaReservacion**

- Entrada: (1) Tarjeta de crédito (String) (2) identificador de reservación (String)
- Salida: (1) Estatus de operación: Éxito o Falla.

Notas de Implementación:

El primer método puede regresar un costo y un identificador aleatorio. El segundo método puede siempre regresar Éxito



Composición de Servicios Turísticos

- *Implementación de Servicio Agencia*

creaReservacion

```
id1 <- ReservaHotel().getId()  
id2 <- ReservaVuelo().getId()  
id3 <- ReservaCarro().getId()  
Id = id1 + '-' + id2 + '-' + id3  
costo1 <- ReservaHotel().getCosto()  
costo2 <- ReservaVuelo().getCosto()  
costo3 <- ReservaCarro().getCosto()  
costo = costo1 + costo2 + costo3  
return Reservacion(id, costo)
```



Composición de Servicios Turísticos

- *Implementación de Servicio Agencia*

confirmaReservacion

```
Si (confirmaHotel(tarjeta, id) &&
    confirmaCarro(tarjeta, id) &&
    confirmaVuelo(tarjeta, id))
```

```
    return “Exito”
```

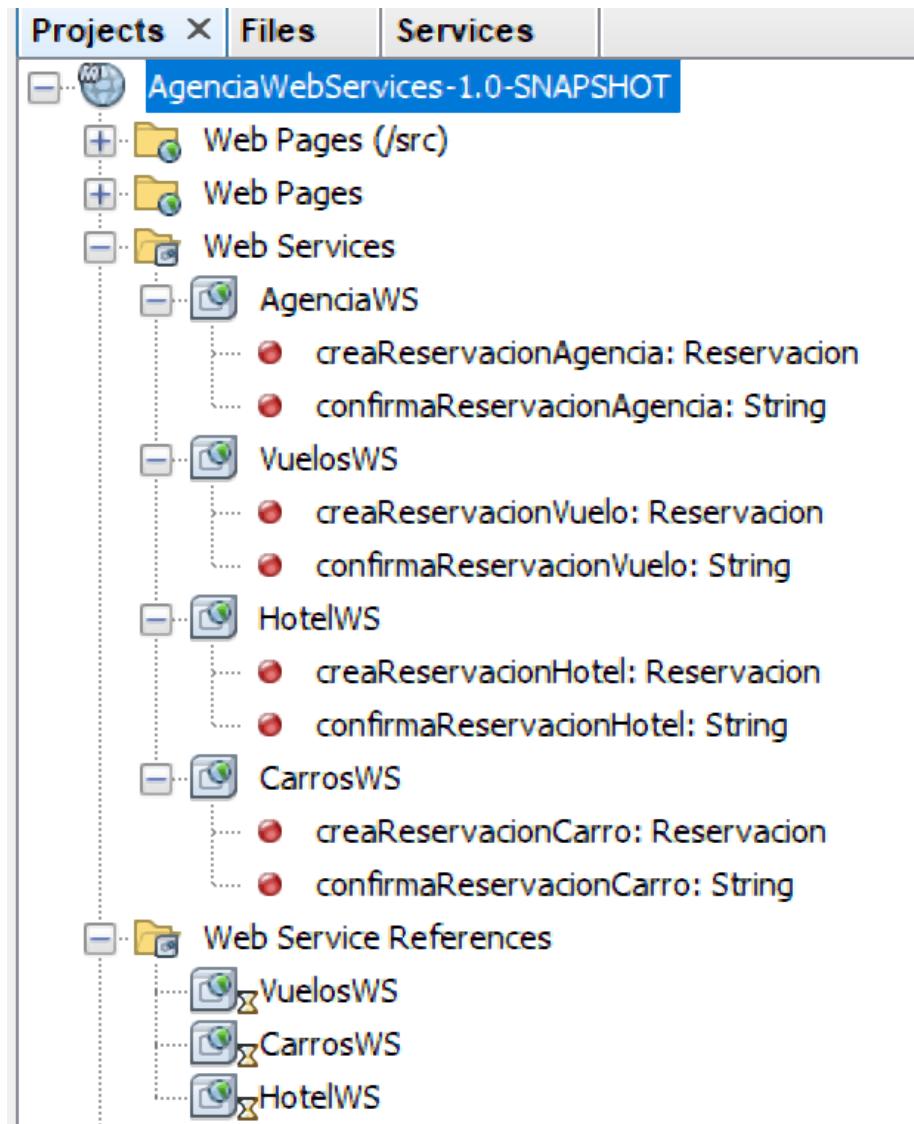
```
Si no
```

```
    return “Fallo”
```



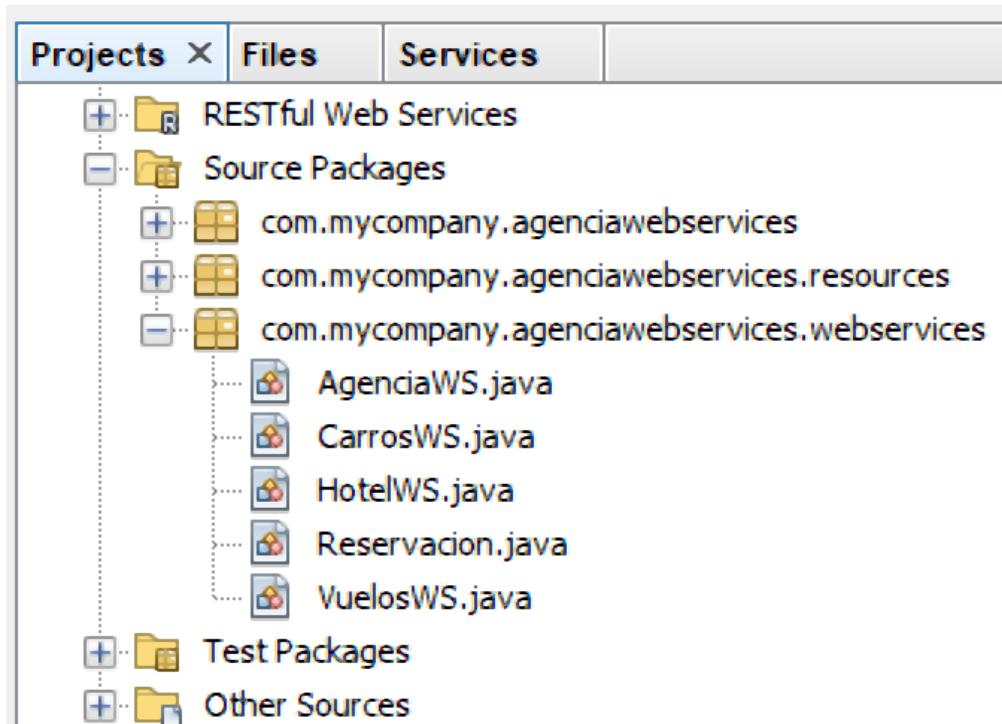
Composición de Servicios Turísticos

- *Estructura del proyecto parte I*



Composición de Servicios Turísticos

- *Estructura del proyecto parte 2*



Composición de Servicios Turísticos

- *Estructura del proyecto parte 3*

