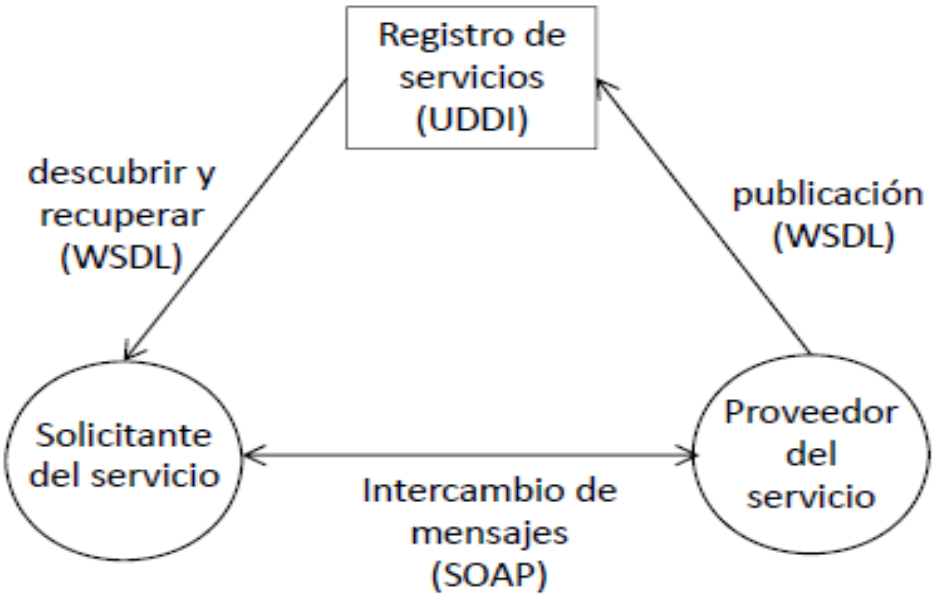
Servicios web

- Hay una primera generación de servicios web caracterizada por el uso de:
  - *Web Services Description Language* (WSDL)
  - *Simple Object Access Protocol* (SOAP)
  - *Universal Description, Discovery and Integracion* (UDDI)
  - *WS-I Basic Profile*

Servicios web



Primera generación de servicios web

Servicios web

- Hay una segunda generación de servicios web basados en extensiones a los servicios web de primera generación. Entre otras:
  - *WS-Security Specification and Frameworks*
  - *WS-Addressing Specification*
  - *WS-Reliable Messaging Specifications*
  - *WS-Business Process Execution Language*
  - *WS-Choreography Definition Language* (*)
  - *WS-Metadata Exchange Specifications*

* **La Coreografía de Servicios Web (WS-Choreography**) es una especificación establecida por el W3C que define procesos de modelo de negocio basados en XML, los cuales **describen protocolos de colaboración entre participantes de un servicio Web**, en el cual los servicios actúan como pares, y las interacciones pueden tener un ciclo de vida y estados de larga duración.

Servicios web

- Hay dos implementaciones de servicios web:
  - Servicios web REST (**RE**presentational **S**tate **T**ransfer)
  - Servicios web SOAP (**S**imple **O**bject **A**ccess **P**rotocol)

	REST	SOAP
Formato del mensaje	XML	XML dentro de SOAP
Definición del interfaz	no es necesario	WSDL
Transporte	HTTP	HTTP, JMS, FTP, etc.

Diferencias entre servicios web REST y SOAP

Servicios web REST

- Los servicios web REST están basados en la tesis doctoral de Roy Fielding (*)
- Se habla de servicios web REST-style, o RESTful
- Se basan en una serie de principios:
  - *Los servicios RESTful no tienen estado*. Cada petición del cliente al servidor debe contener toda la información necesaria para entender la petición, y no puede aprovecharse de ningún contexto almacenado en el servidor

(*) Uno de los principales autores de la especificación del protocolo HTTP

Servicios web REST

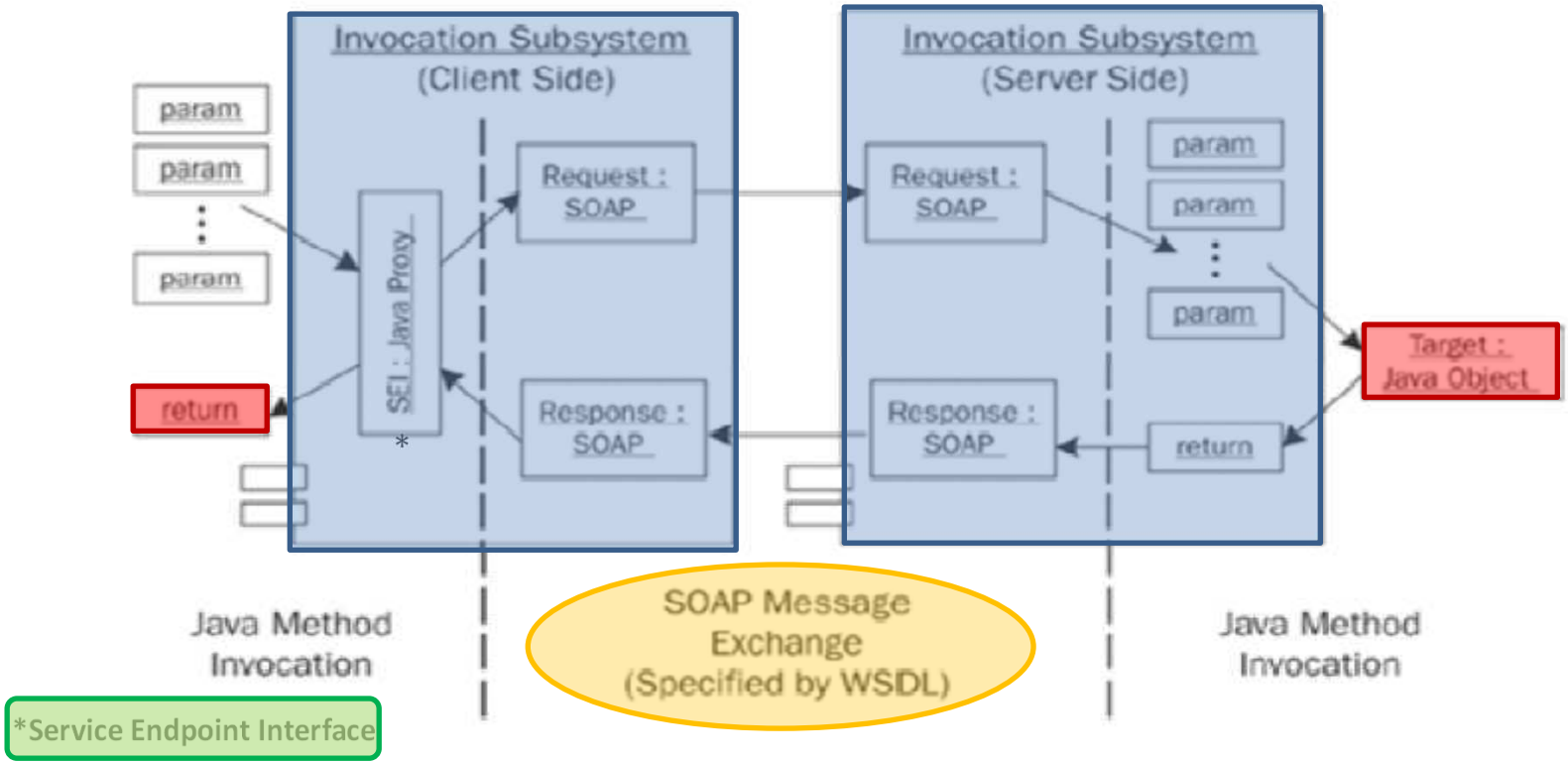
- *Los servicios web tienen un interfaz común.* Esto suele significar que las únicas operaciones permitidas son aquellas proporcionadas por el protocolo HTTP: `get`, `post`, `put` y `delete`
- *Las arquitecturas REST se construyen sobre recursos que son identificados de manera unívoca por URIs*(*). Así, cada URI representa una función distinta y por extensión a un dato distinto
- *Los componentes REST manipulan recursos intercambiando representaciones de los mismos.* Así, la información se intercambia en formato XML

(*) Un identificador de recursos uniforme o URI (Uniform Resource Identifier) es una cadena de caracteres que identifica los recursos de una red de forma unívoca. La diferencia respecto a un localizador de recursos uniforme (URL) es que estos últimos hacen referencia a recursos que, de forma general, **pueden variar en el tiempo**.

Servicios web SOAP

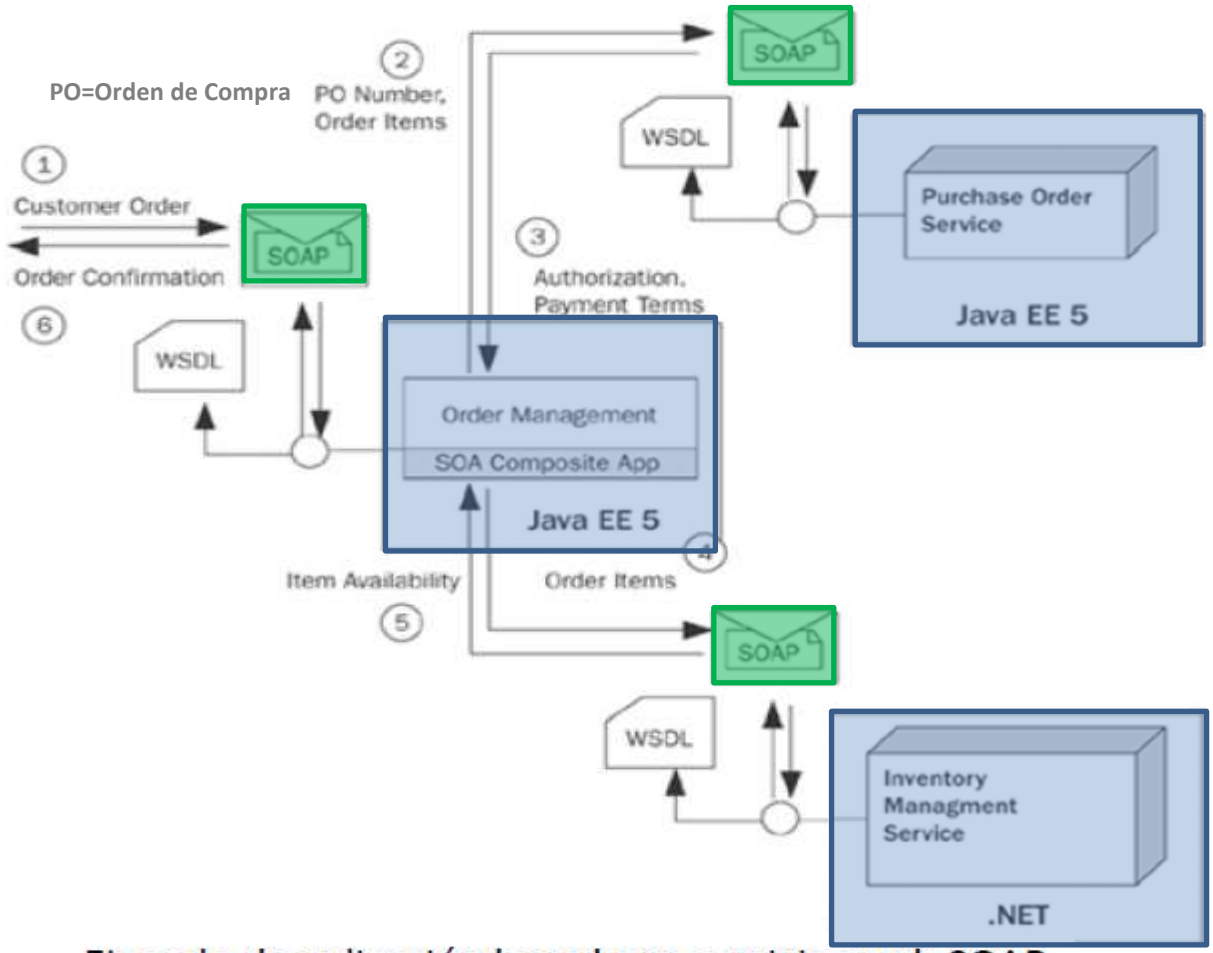
- Los servicios web SOAP utilizan SOAP para
  - Invocar servicios web descritos a través de interfaces WSDL **(Web Services Description Language)**
  - Codificar los parámetros de entrada y salida del servicio invocado en formato XML
- Requieren un mecanismo para traducir:
  - Peticiones de servicio en elementos de lógica
  - Parámetros de un lenguaje de programación en otros

Servicios web SOAP



Invocación y respuesta de servicios web SOAP

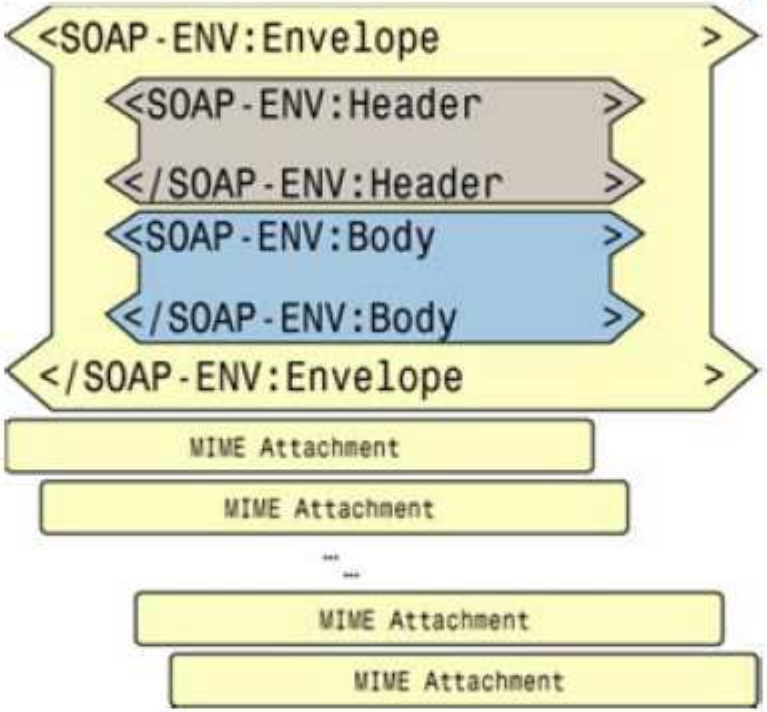
Servicios web SOAP



Ejemplo de aplicación basada en servicios web SOAP

Servicios web SOAP

- La estructura básica de un mensaje SOAP es:



Estructura básica de un mensaje SOAP

Servicios web SOAP

- Donde:
  - `Envelope`: es el elemento raíz. Guarda la información de espacios de nombres y de estilo de codificación. El Sobre (envelope) define qué hay en el mensaje y cómo procesarlo
  - `Header`: es opcional, aunque no suele omitirse. Proporcionan servicios a la carga transportada en el elemento `Body`
  - `Body`: es obligatorio. Contiene la carga del mensaje SOAP que será procesada por el punto final destino

Servicios web SOAP

- Ejemplo de petición SOAP

```
<env1:Envelope
xmlns:env1="http://schemas.xmlsoap.org/soap/envelope">
 <env1:Body>
      <getord:getOrdersDates
xmlns:getord="http://www.example.com/oms/getorders">
        <getord:startDate>2005-11-19</getord:startDate>
        <getord:endDate>2005-11-22</getord:endDate>
      </getord:getOrdersDates>
 </env1:Body>
</env1:Envelope>
```

Servicios web SOAP

- Ejemplo de respuesta SOAP

```
<env1:Envelope
xmlns:env1="http://schemas.xmlsoap.org/soap/envelope">
  <env1:Body>
    <getord:getOrdersDatesResponse

xmlns:getord="http://www.example.com/oms/getorders">
      <Orders xmlns="http://www.example.com/oms"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xsi:schemaLocation="http://www.example.com/oms
        http://soabook.com/example/oms/orders.xsd">
```

xmlns: The namespace to use (for XHTML documents)

(*)

* Un archivo XSD (XML Schema Definition) muestra la relación entre los diferentes elementos de un archivo XML (E**Xt**ensible **M**arkup **L**anguage)

Servicios web SOAP

```
<Order>
   <OrderKey>ENT1234567</OrderKey>
      <OrderHeader>
         <SALES_ORG>NE</SALES_ORG>
         <PURCH_DATE>2005-11-20</PURCH_DATE>
         <CUST_NO>ENT0072123</CUST_NO>
         <PYMT_METH>PO</PYMT_METH>
         <PURCH_ORD_NO>PO-72123-0007</PURCH_ORD_NO>
        <WAR_DEL_DATE>2006-12-20</WAR_DEL_DATE>
       </OrderHeader>
.....
```

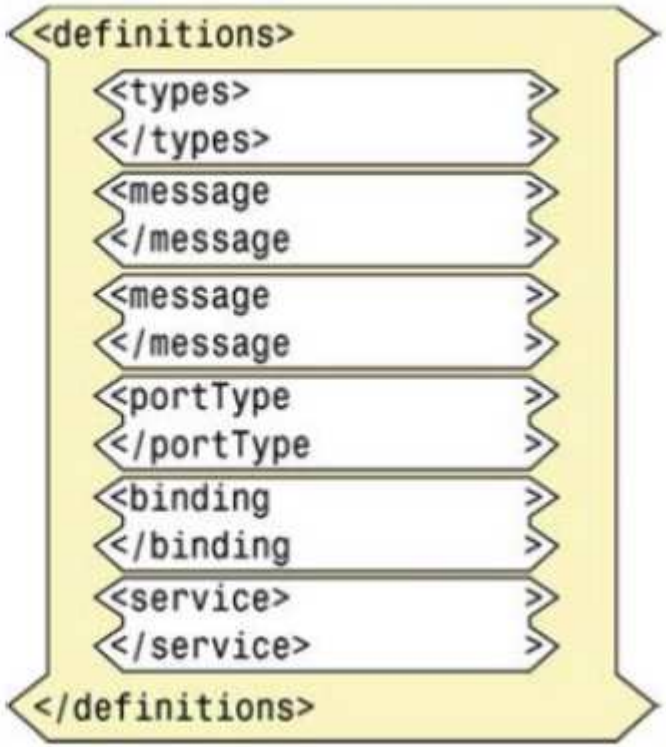
Servicios web SOAP

- **WSDL** describe la interfaz pública a los servicios Web.
- Está basado en XML y **describe la forma de comunicación**, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo.
- **Las operaciones y mensajes que soporta se describen en abstracto** y **se enlazan después al protocolo concreto de red y al formato del mensaje**.

Servicios web SOAP

- **"WSDL** se usa a menudo en combinación con SOAP y XML Schema."
- **Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor**.
- Los **tipos de datos** especiales se incluyen en el archivo WSDL en forma de XML Schema.
- El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.
- El WSDL permite tener una descripción de un servicio web. **"Especifica la interfaz abstracta a través de la cual un cliente puede acceder al servicio y los detalles de cómo se debe utilizar"**.

- La estructura de un interfaz WSDL es:



Estructura interfaz WSDL

Servicios web SOAP

- Donde:
  - `definitions:` es el elemento raíz y contiene los elementos que definen a un servicio web
  - `types:` actúa como un contenedor de las definiciones de los tipos de datos utilizando definiciones de esquemas XML. Estos esquemas definen los tipos de datos de la información utilizando elementos `message`

Servicios web SOAP

- `message`: estos elementos caracterizan definiciones tipadas abstractas de los datos intercambiados. Estos elementos referencian los tipos de datos definidos en `types`
- `portType`: este elemento describe uno o más conjunto de operaciones abstractas soportadas por uno o más puntos finales. Utiliza elementos `operation` para describir dichas operaciones, que hacen referencia a los valores definidos en `message` para caracterizar los parámetros de las funciones

Servicios web SOAP

- `binding:` especifica un protocolo concreto para la vinculación de transporte y una especificación de formato de datos para un elemento `portType`
- `service:` identifica el servicio web en su atributo `name`. Modela múltiples servicios web a través de una colección de elementos `port`, que indican puntos finales como combinación de vinculaciones de transporte y direcciones de red.
  - `port:` proporciona el nombre y la localización del sistema en el que se encuentra el servicio web. Modela un servicio web individual

```
<definitions name="StockQuote"
         targetNamespace="http://example.com/stockquote.wsdl"
         xmlns:tns="http://example.com/stockquote.wsdl"
         xmlns:xsd1="http://example.com/stockquote.xsd"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns="http://schemas.xmlsoap.org/wsdl/">

   <types>
      <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
          <element name="TradePriceRequest">
             <complexType>
                 <all>
                     <element name="tickerSymbol" type="string"/>
                 </all>
             </complexType>
          </element>
          <element name="TradePrice">
             <complexType>
                 <all>
                     <element name="price" type="float"/>
                 </all>
             </complexType>
          </element>
      </schema>
   </types>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
           <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>

</definitions>
```

Ejemplo de definición de Interfaz WSDL

Java WS

- Java-WS permite definir y utilizar servicios web de forma sencilla
- Utilizando `@anotaciones` Java, herramientas automáticas pueden publicar los interfaces WSDL necesarios para su uso
- Se utiliza junto a JAXB (*Java Architecture for XML Bindind*), que mapea clases Java a esquemas XML y viceversa

Java WS

- En el lado servidor hay una serie de anotaciones clave:
  - `@WebService`: indica la clase Java responsable de implementar el servicio web. Puede utilizarse también con interfaces para *representar Service Endpoint Interface* (SEI)
  - `@WebMethod`: indica que un método Java se expone como una operación del servicio web

Java WS

- `@WebParam`: personaliza el parámetro de un método de un servicio web a la parte del mensaje WSDL correspondiente
- `@WebResult`: especifica el maped del tipo de retorno de un método de un servicio web
- `@SOAPBinding`: especifica mapear el servicio web al protocolo de mensaje SOAP

Java WS

- Ejemplo:

```
@WebService(name = "Suma",
            targetNamespace =
"http://servicioAplicacion.negocio/")

public interface Suma {
  public TransferResultado sumar (TransferSuma ts);
}
```

Java WS

```
public class TransferSuma {
      float x;
      float y;
……}

public class TransferResultado {
      float resultado;
……}
```

Java WS

```
@WebService(
targetNamespace= "http://servicioAplicacion.negocio/",
endpointInterface = "negocio.servicioAplicacion.Suma",
portName = "SumaImplementacionPort",
serviceName = "SumaImplementacionService")

public class SumaImplementacion implements Suma {

  public TransferResultado sumar (TransferSuma ts)
  {
   return  new TransferResultado(ts.getX()+ts.getY());
  }

}
```

Java WS

- En el lado cliente también se utilizan anotaciones en clases o interfaces (SEI) para utilizar los servicios web
- En su mayor parte, coinciden con las anotaciones utilizadas en el lado servidor

Java WS

- Ejemplo:

```
@WebService(targetNamespace =
"http://servicioAplicacion.negocio/",
            name = "Suma")

public interface Suma {

    @WebResult(name = "return", targetNamespace = "")
    @RequestWrapper(localName = "sumar", targetNamespace
= "http://servicioAplicacion.negocio/", className =
"negocio.servicioaplicacion.Sumar")
    @WebMethod
    @ResponseWrapper(localName = "sumarResponse",
targetNamespace = "http://servicioAplicacion.negocio/",
className = "negocio.servicioaplicacion.SumarResponse")
```

Java WS

```
 public negocio.servicioaplicacion.TransferResultado
sumar(@WebParam(name = "arg0",
                targetNamespace = "")
negocio.servicioaplicacion.TransferSuma arg0
);
}
```

Java WS

```
@WebServiceClient(name =
"SumaImplementacionService",
                  wsdlLocation =
"http://localhost:55555/servicioSuma/services/SumaIm
plementacionPort?wsdl",
                  targetNamespace =
"http://servicioAplicacion.negocio/")
public class SumaImplementacionService extends
Service {
……. }
```

Java WS

```
public final class Main {
    private static final QName SERVICE_NAME = new
QName("http://servicioAplicacion.negocio/",
"SumaImplementacionService");

    public static void main(String args[]) throws
Exception {
        URL wsdlURL =
SumaImplementacionService.WSDL_LOCATION;
        SumaImplementacionService ss = new
SumaImplementacionService(wsdlURL, SERVICE_NAME);
        Suma port = ss.getSumaImplementacionPort();
```

Java WS

```
        TransferSuma ts= new TransferSuma();
        ts.setX(4);
        ts.setY(3);

        TransferResultado tr= port.sumar(ts);

        System.out.println (tr.getResultado());
        System.exit(0);
    }
}
```

Referencias

- Thomas Erl, *SOA Principles of Service Design*, Prentice Hall, 2008
- Mark D. Hansen, *SOA Using Java Web Services*, Prentice Hall, 2007
- B.V. Kumar, P. Narayan, T. Ng, *Implementing SOA Using Java EE*, Addison-Wesley 2010
- Alur, D., Malks, D., Crupi. J. *Core J2EE Design Patterns: Best Practices and Design Strategies. 2nd Edition*. Prentice Hall, 2003
- Fowler, M. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003