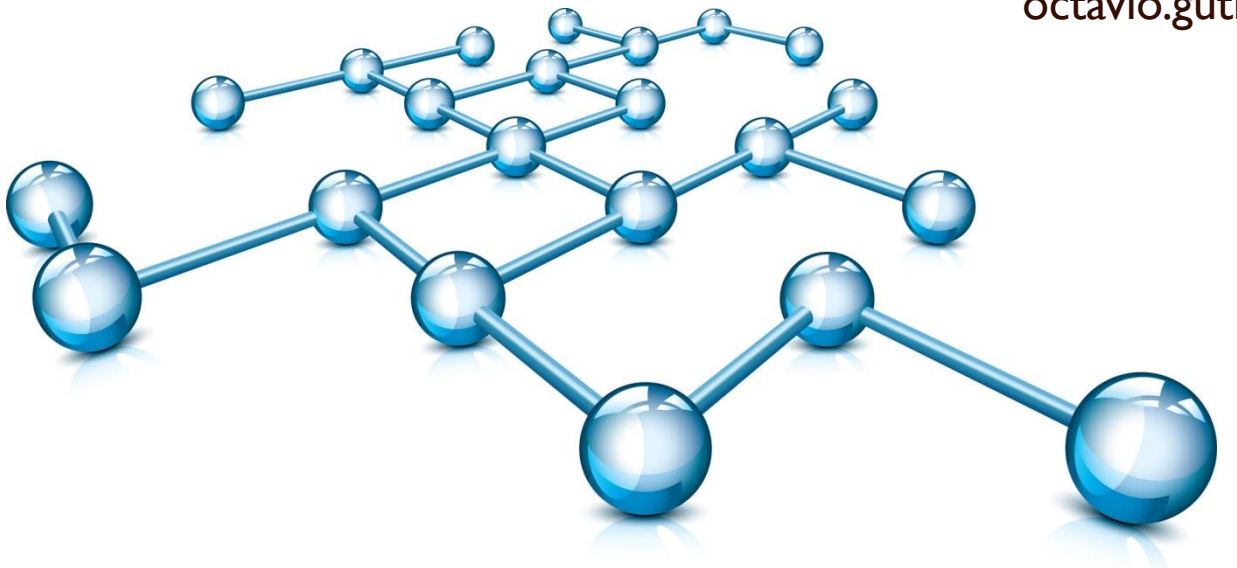
Servicios Web

Profesor:

Dr. J. Octavio Gutiérrez García

octavio.gutierrez@itam.mx



Problemas de aplicaciones Web

Diversas tecnologías:

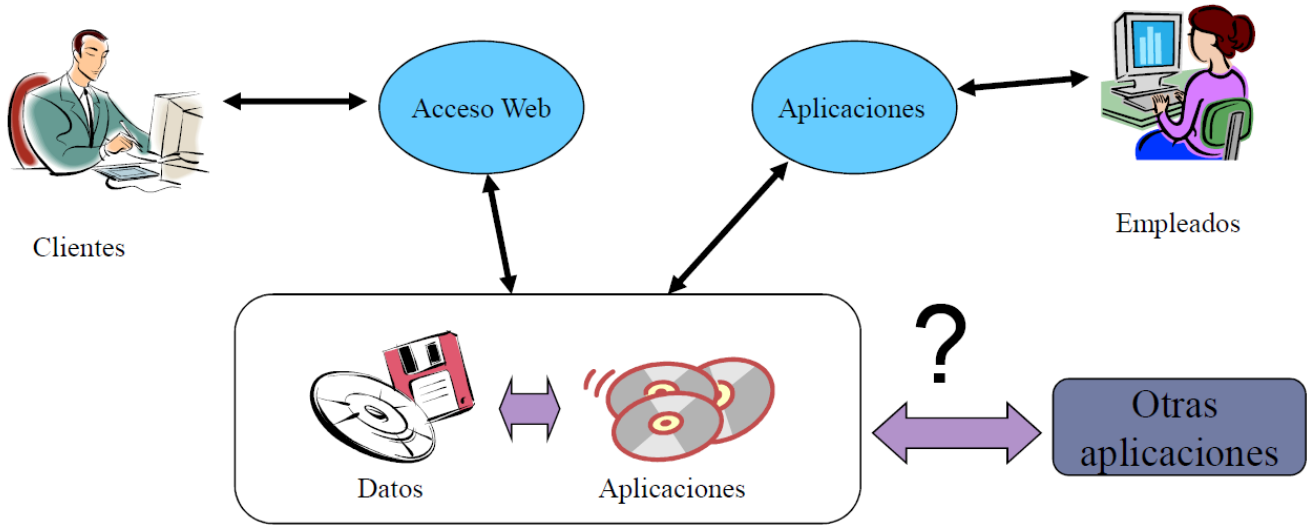
Applets,

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

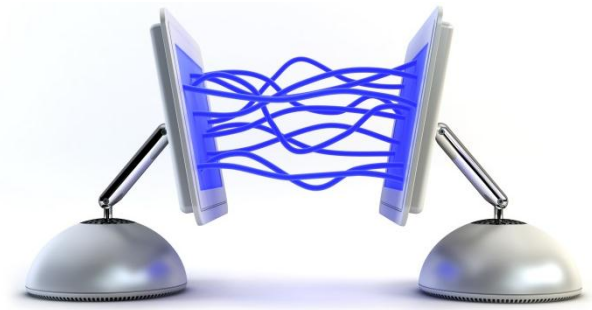
- **Objetivo**: ofrecer una plataforma para construir aplicaciones distribuidas utilizando un software que enmascare la heterogeneidad.

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

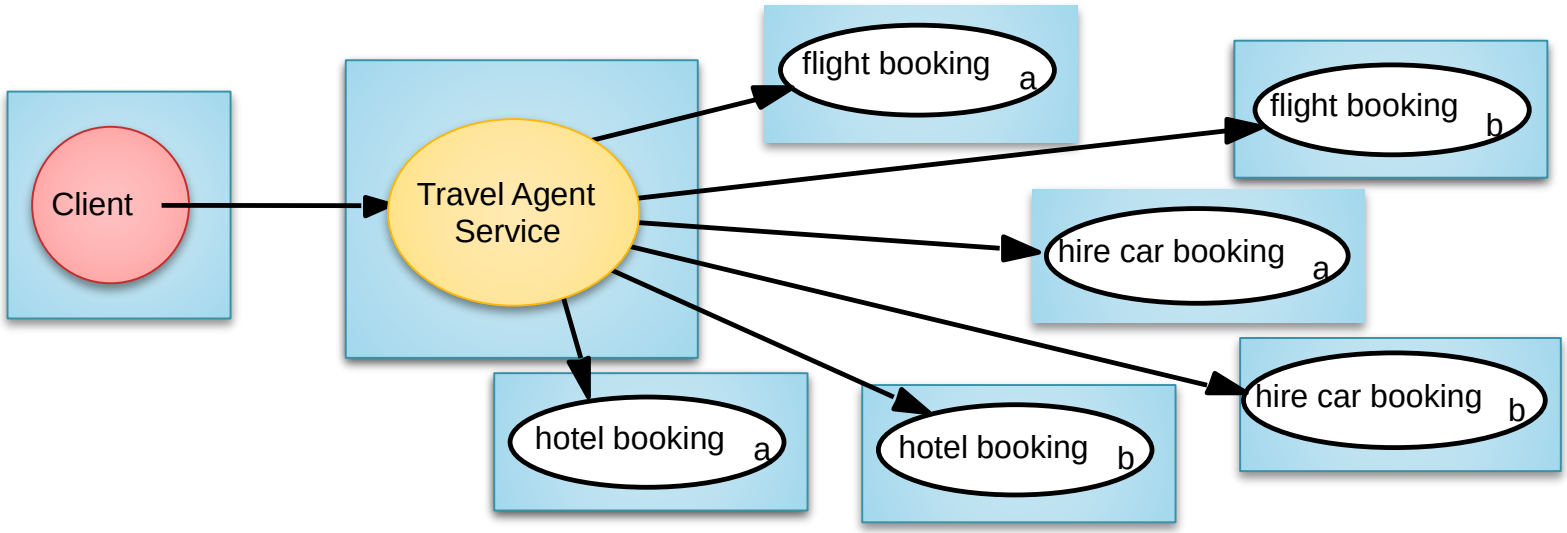
Un servicio web es una colección de protocolos y estándares abiertos que sirven para intercambiar datos entre aplicaciones

- 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

http://www.zdnet.com/news/xml-web-services-need-a-firewall/298402

Aplicaciones		Servicios de directorio	Seguridad
Web Services		WSDL	
SOAP			
URIs	XML	HTTP, SMTP u otros	

Interoperabilidad en entornos heterogéneos

- Servicios basados en protocolos abiertos y mecanismos estándar
  - HTTP

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

Interoperabilidad en entornos heterogéneos

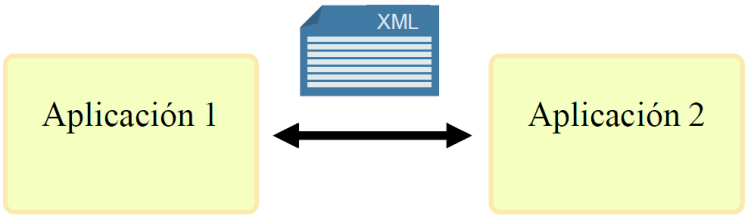
- Servicios basados en protocolos abiertos y mecanismos estándar

  - 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



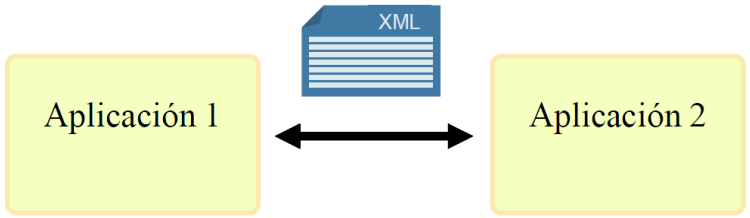
- **Simple Object Access Protocol** permite intercambiar mensajes basados en XML sobre redes de computadoras

- Define un esquema para:
  - Usa XML para representar el contenido de mensajes

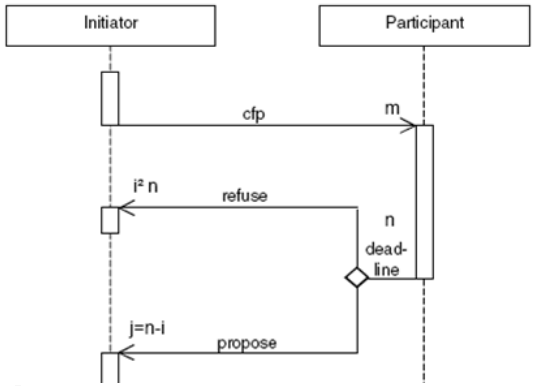
- SOAP (versión 1.2) usa:
  - HTTP, SMTP, TCP o UDP

http://www.w3.org/TR/soap/

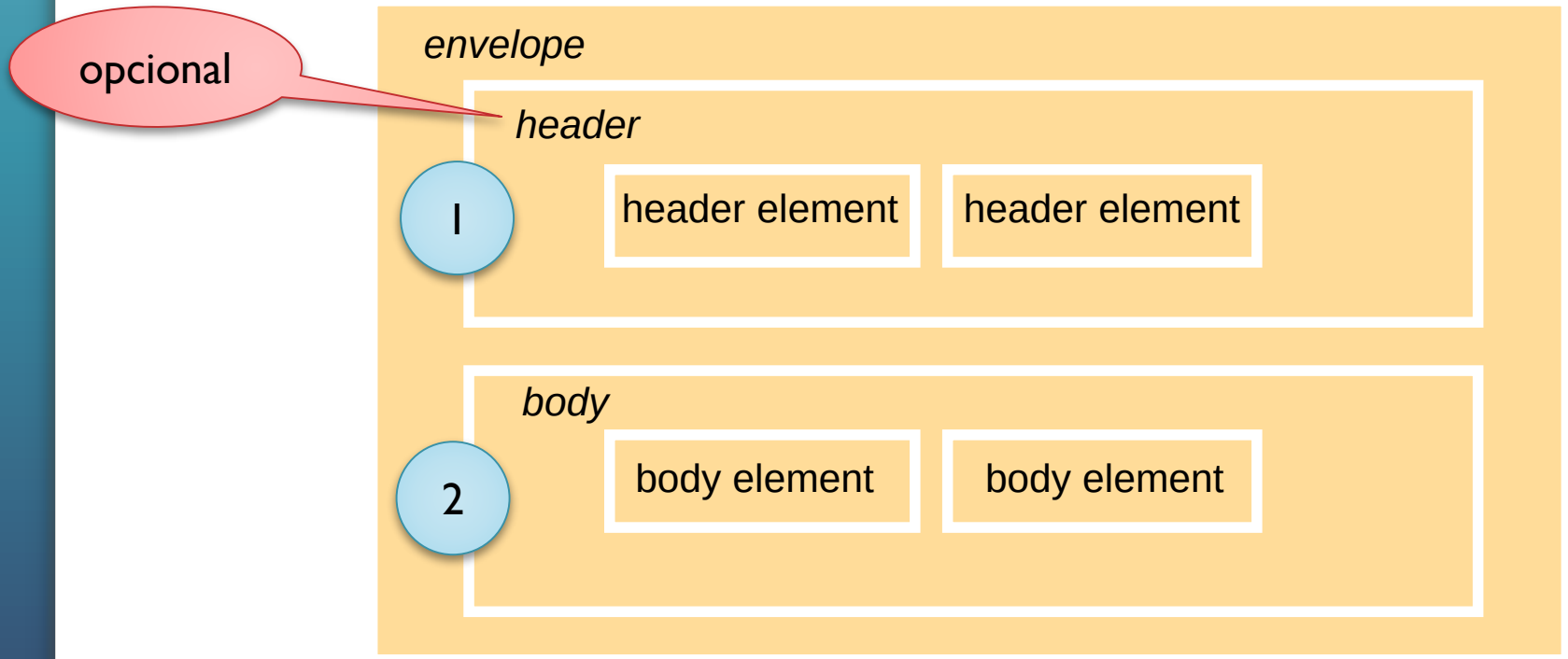
SOAP



- SOAP específica:
  - 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 



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

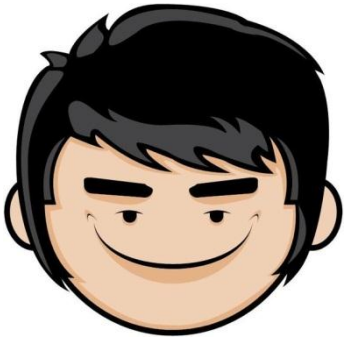
Sobre (Envelope)

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.

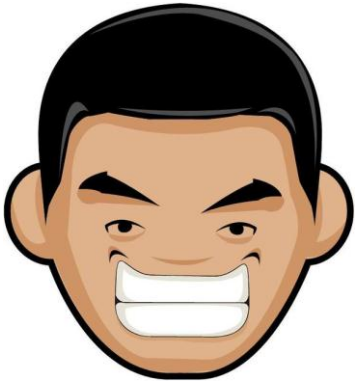


Encabezado (Header)

- 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)

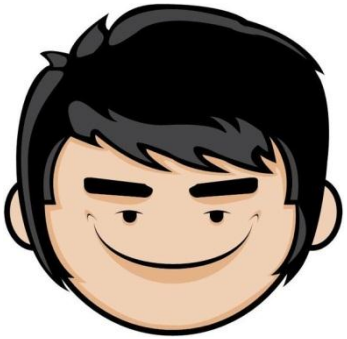
```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<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

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

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

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>
```

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.

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>
```

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>
```

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.

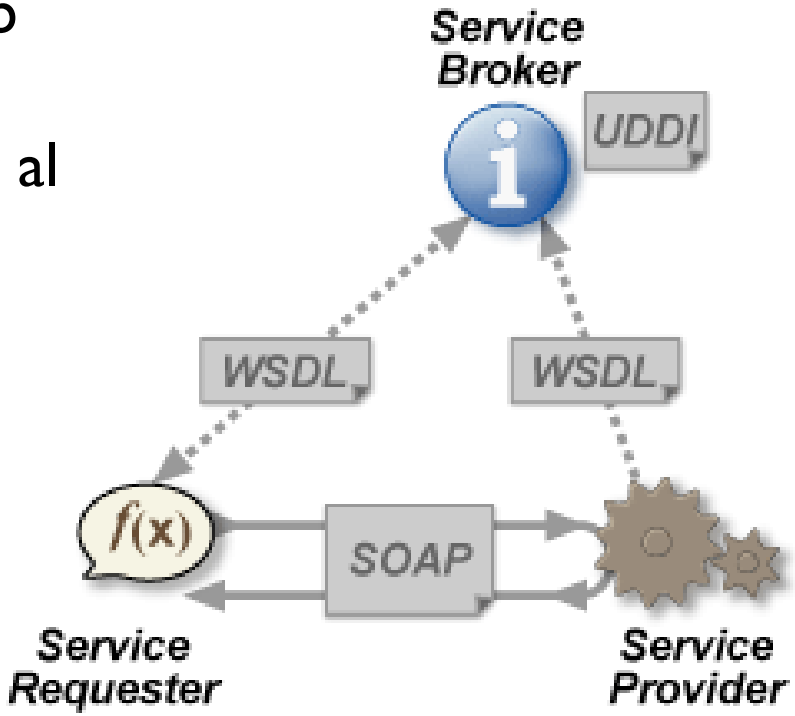
anything

WSDL

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

Describe al servicio web

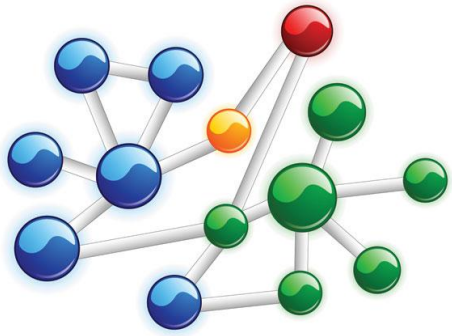
Describe como acceder al servicio web



WSDL – Elementos

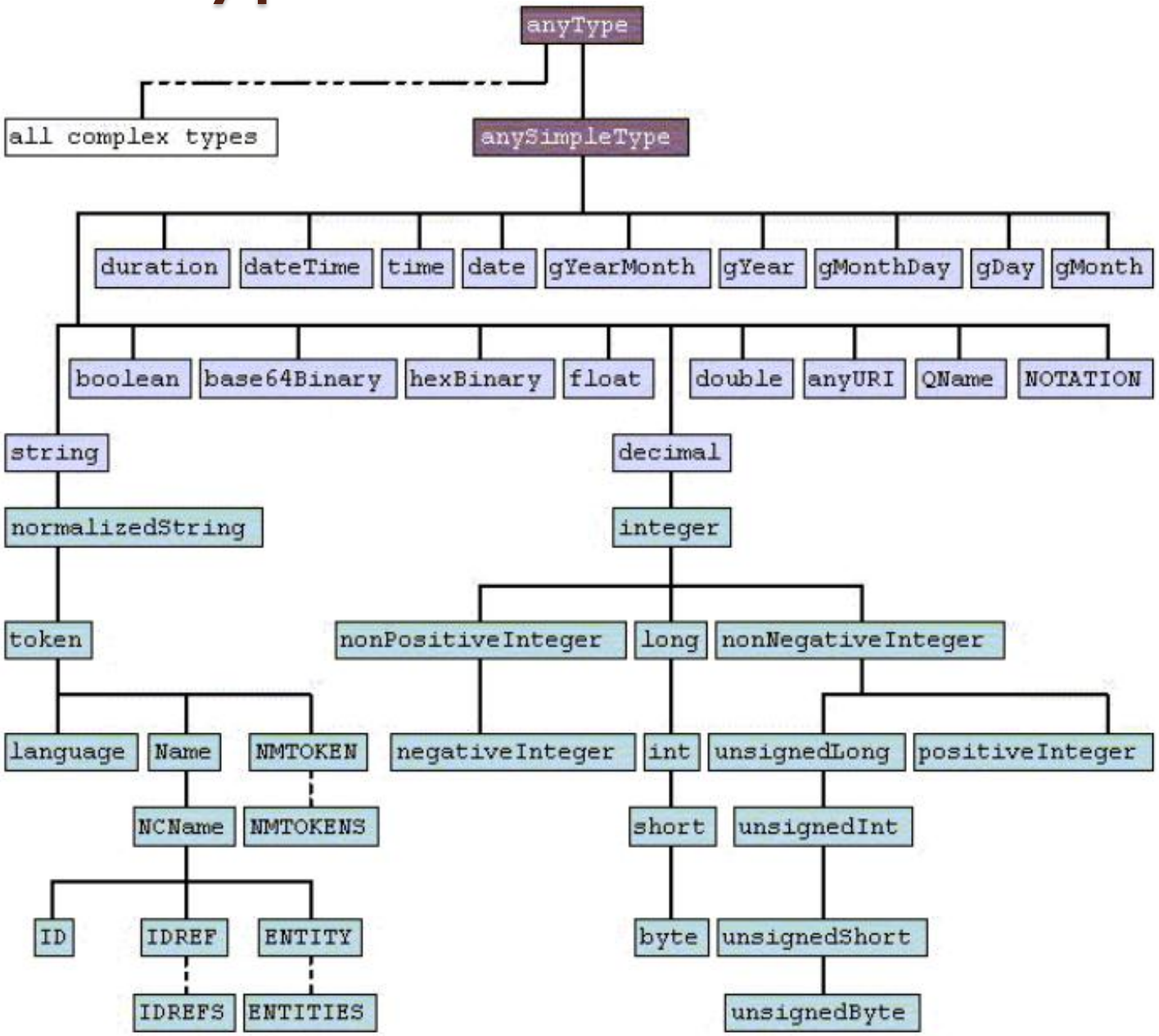
<types>	Contenedor para definiciones de tipos de datos usados por el servicio web
<message>	Definición de los datos a ser comunicados
<portType>	Conjunto de operaciones soportadas por uno o mas puntos finales
<binding>	Define el formato del mensaje y el protocolo

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


anyType
all complex types
anySimpleType
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizedString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte

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 >
```

El Servicio web envía un mensaje

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>
```

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

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

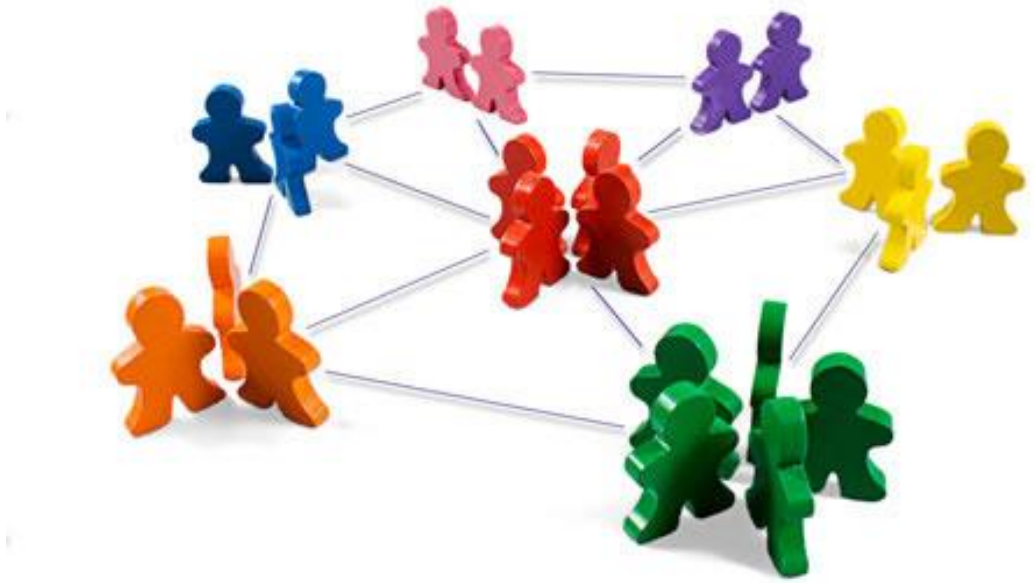
1 `<output message="getExpirationDateResponse"/>`

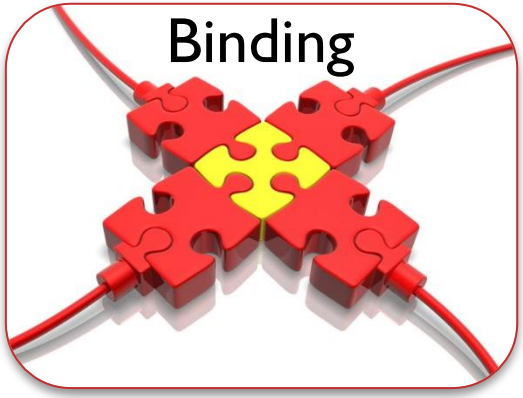
2 `<input message="getExpirationDateSolicit"/>`

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

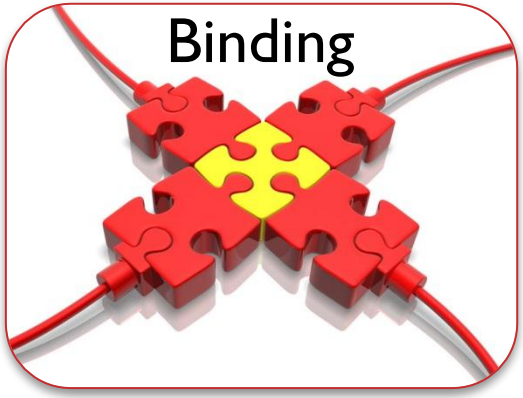
WSDL – Vinculación (Binding)

Define el formato del
mensaje y el protocolo



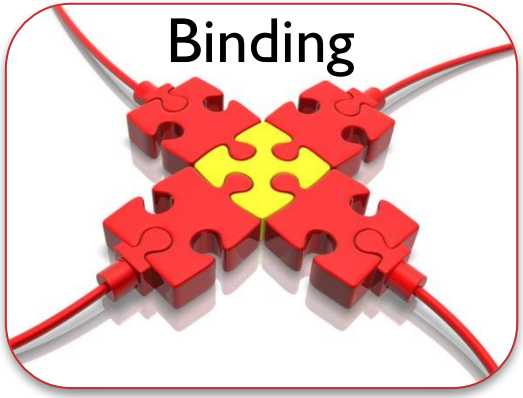



```
<message name="getTermRequest"> … </message>

<message name="getTermResponse" > … </message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermRespon
  </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.
   <operation>
     <soap:operation soapAction
     <input><soap:body use="li
     <output><soap:body use=
  </operation>
</binding>
```

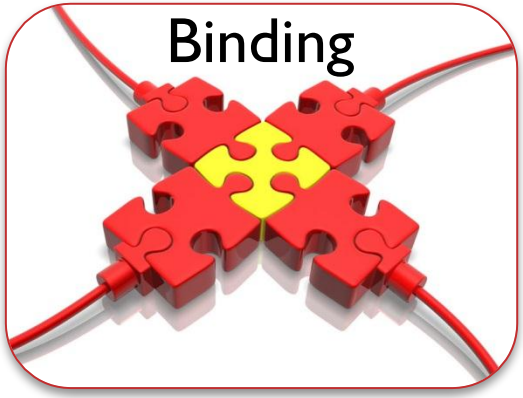


El atributo STYLE se refiere a como traducir el binding a un mensaje SOAP.

Document: mensaje que contiene documentos.

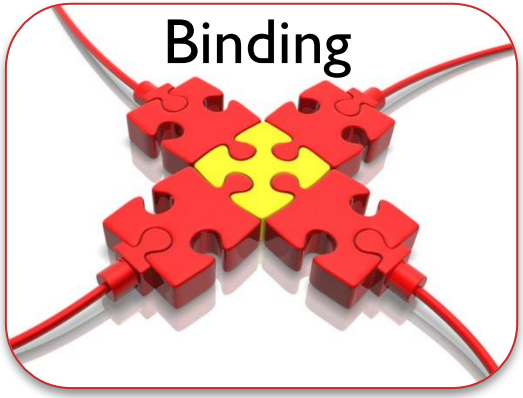
RPC: mensaje con parámetros y valores de retorno.

```
<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"/></outp
  </operation>
</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

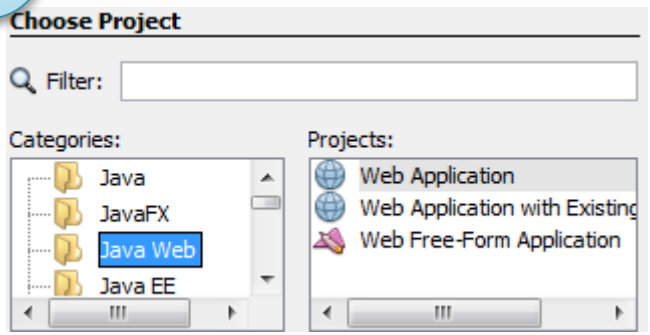
WSDL y UDDI

- 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.

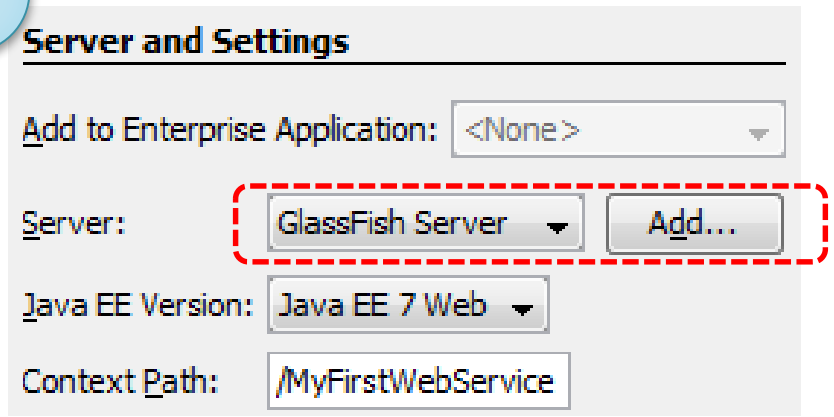
Servicios Web SOAP

Crear aplicación web

1

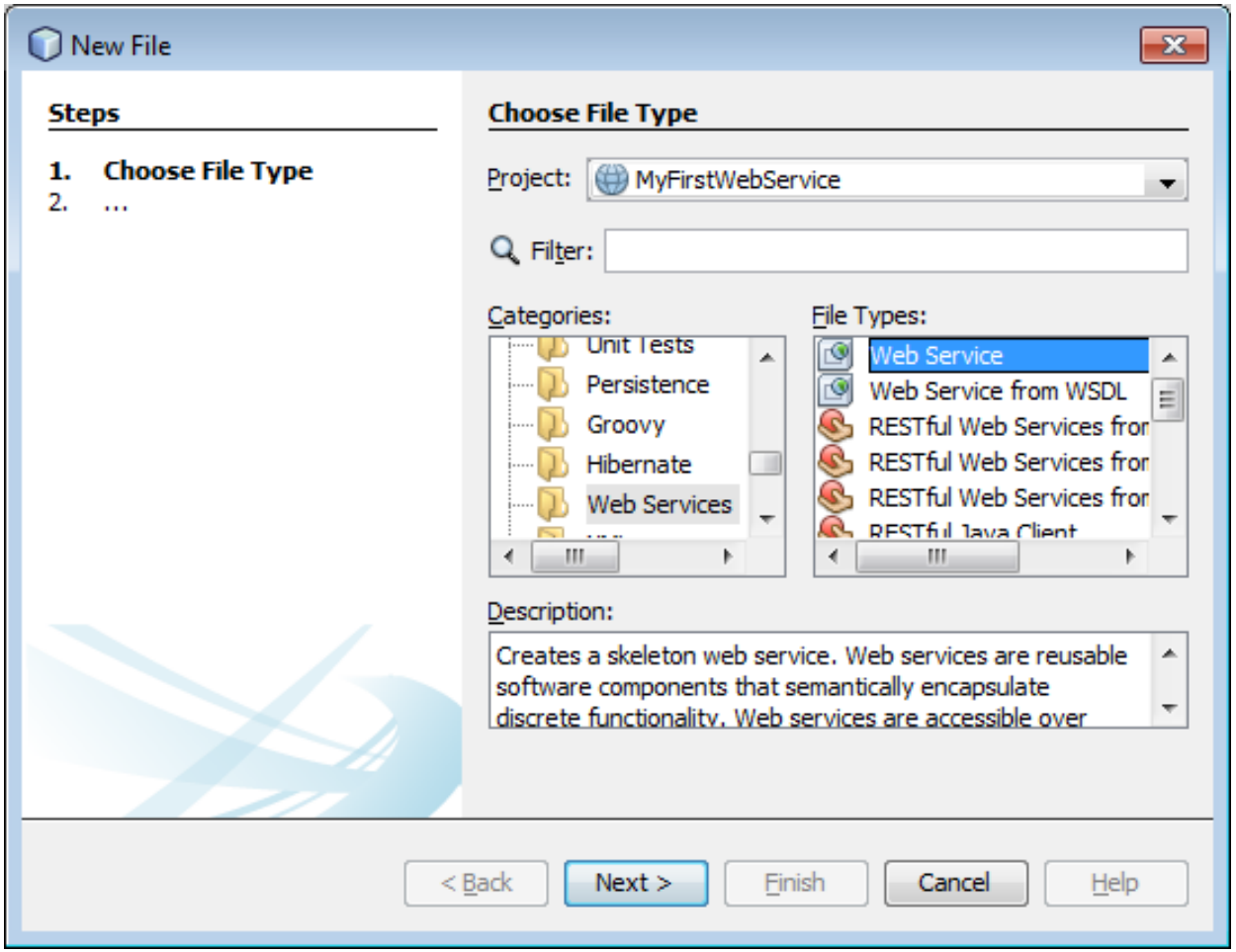


2



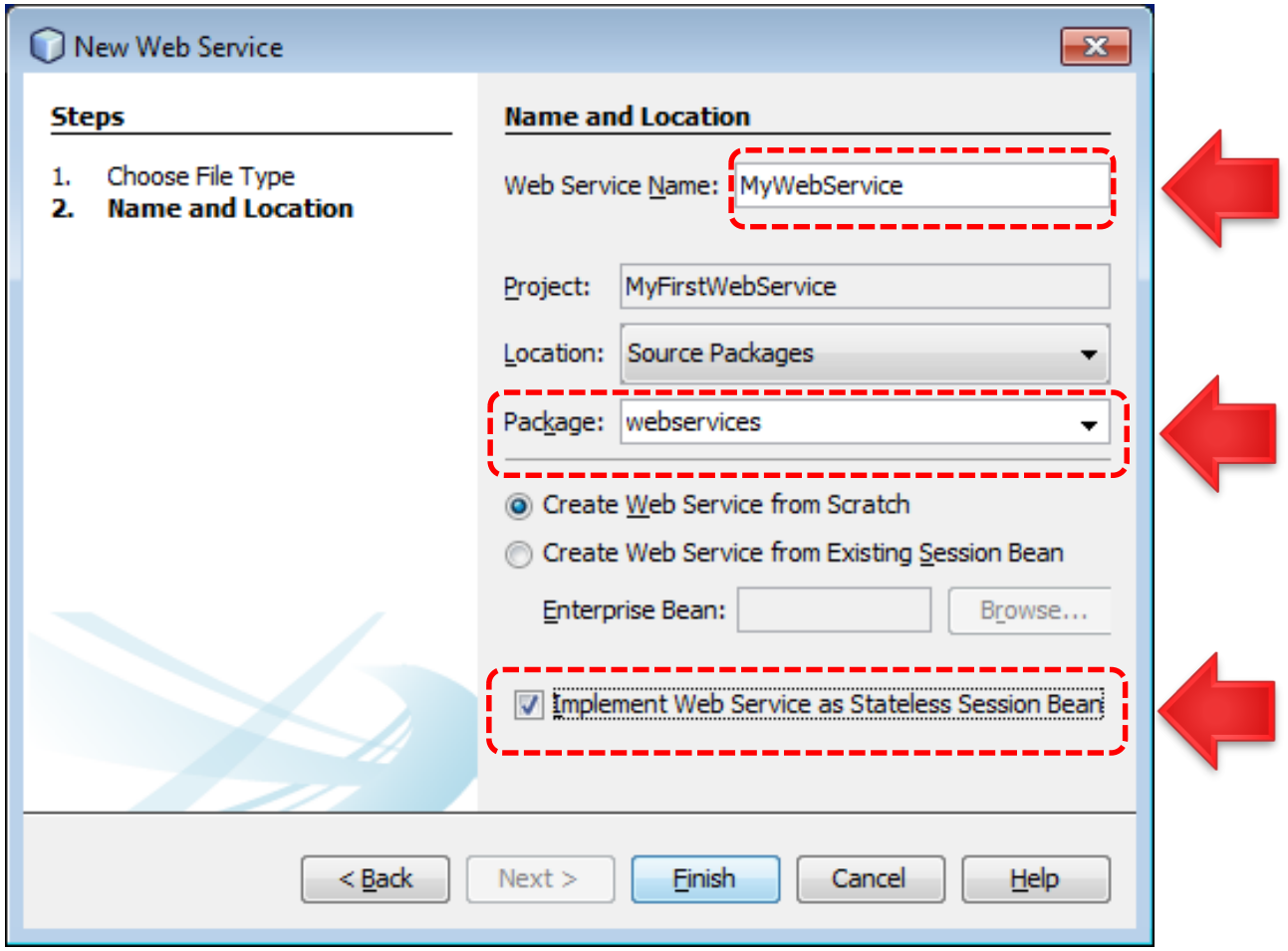
Servicios Web SOAP

Crear servicio web

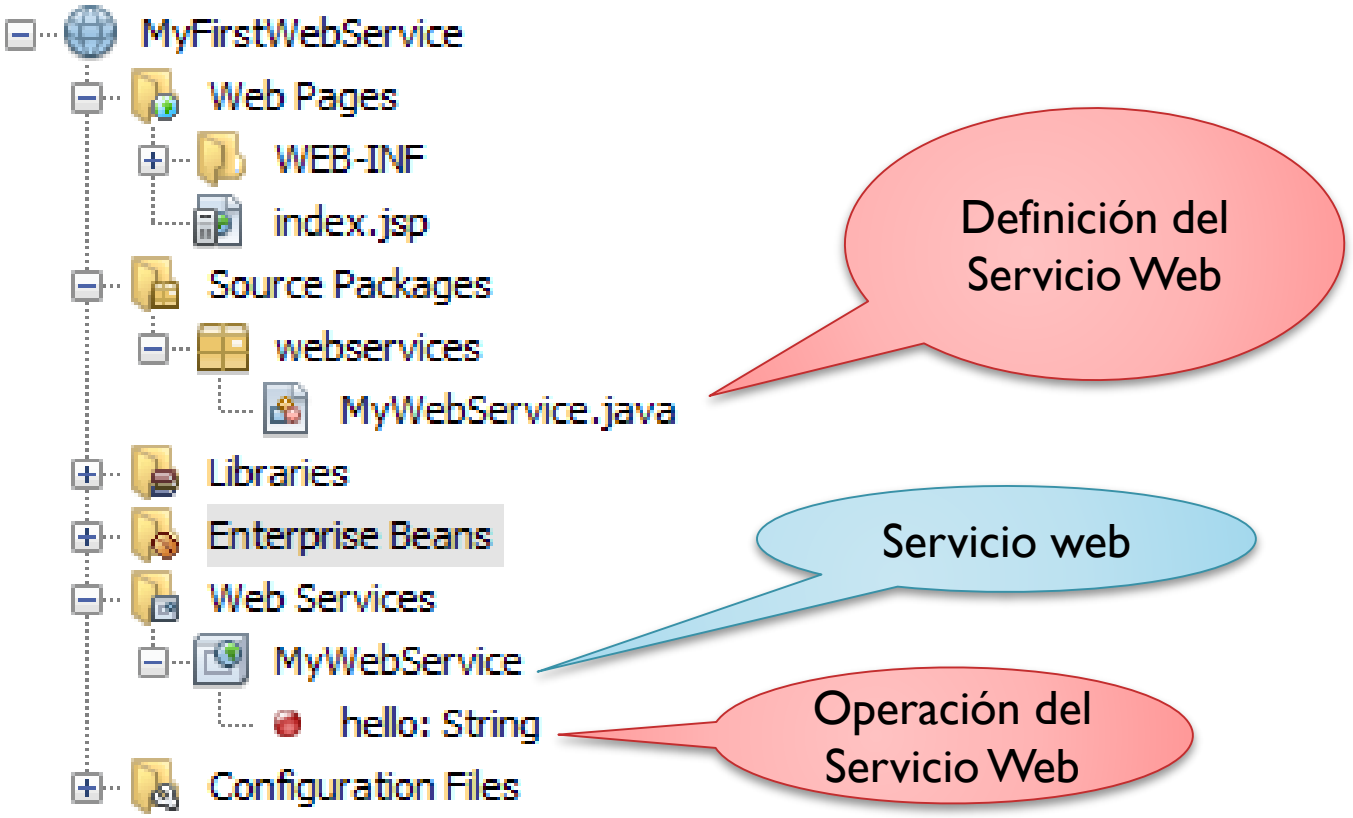


Servicios Web SOAP

Crear servicio web

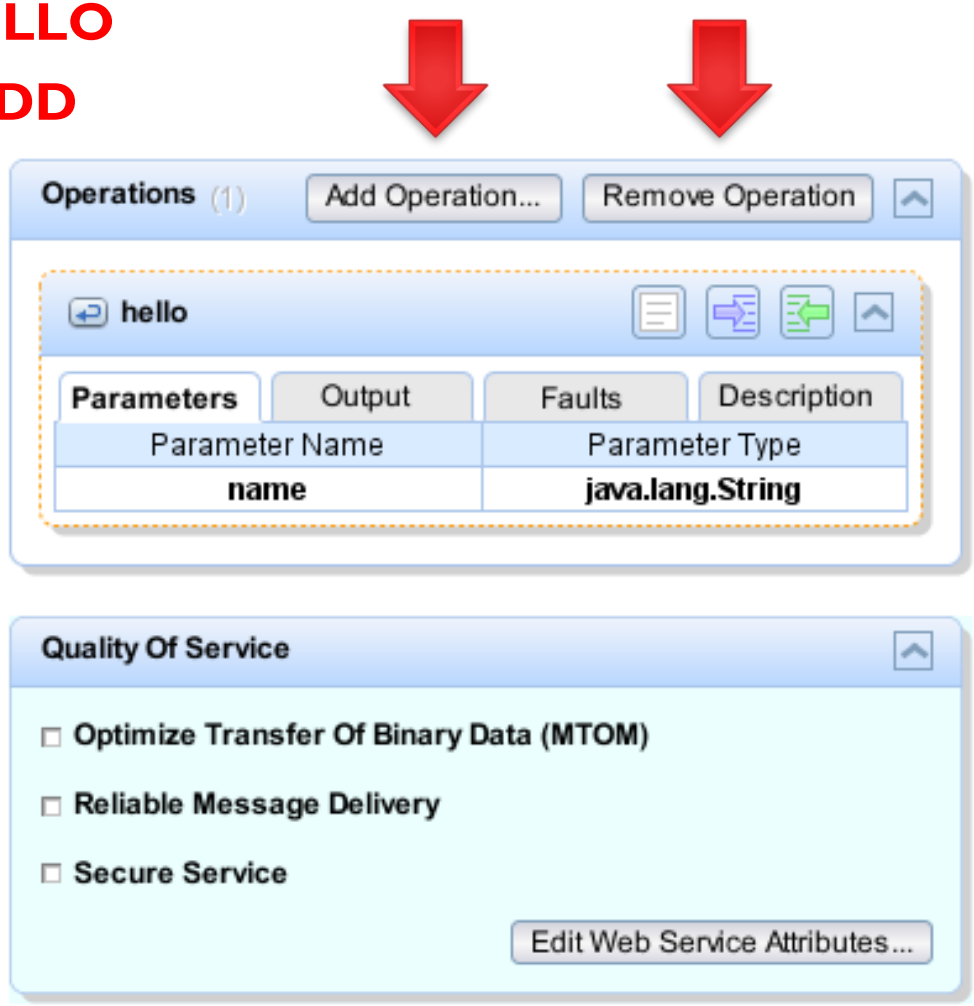


Servicios Web SOAP



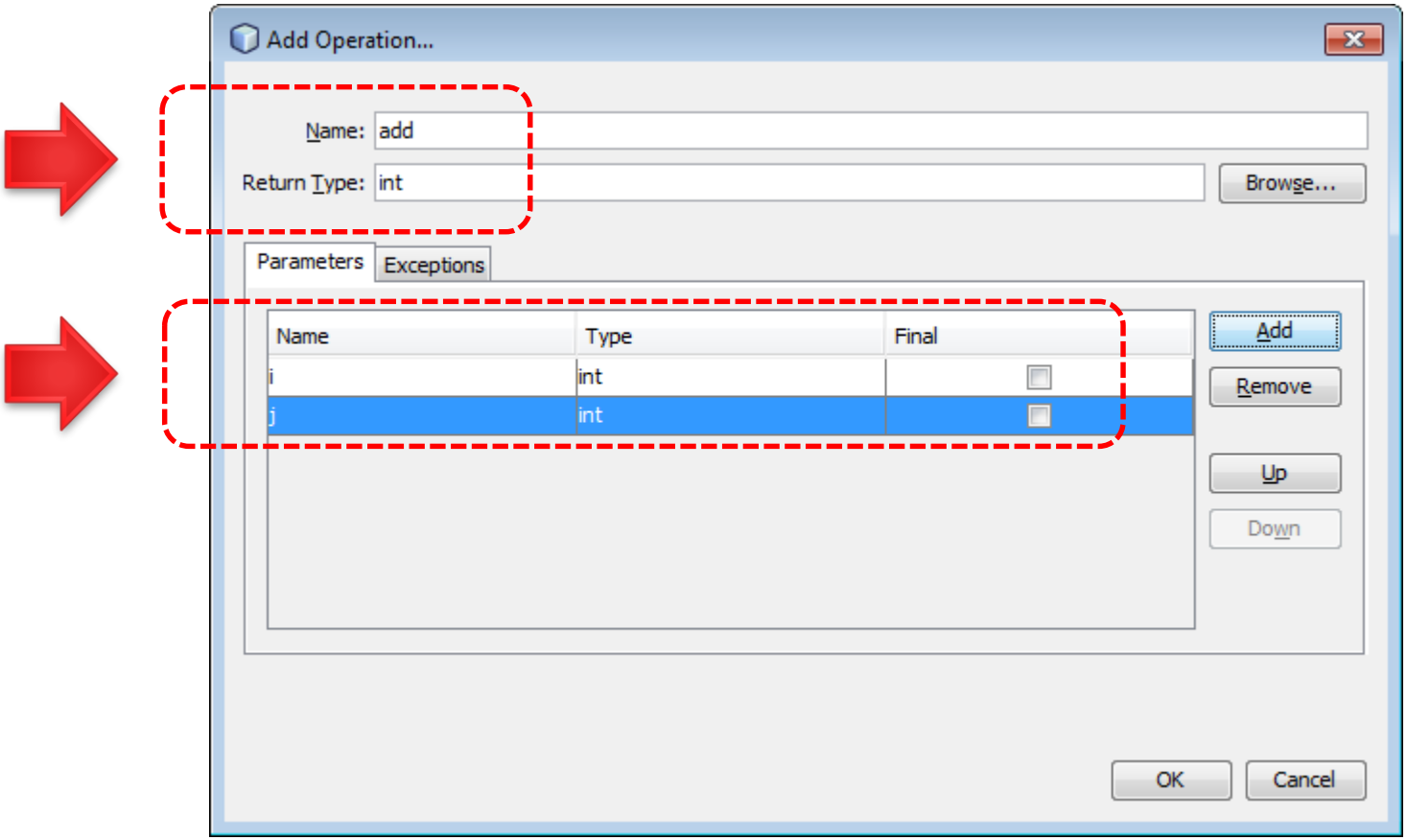
Servicios Web SOAP

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



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

add

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...

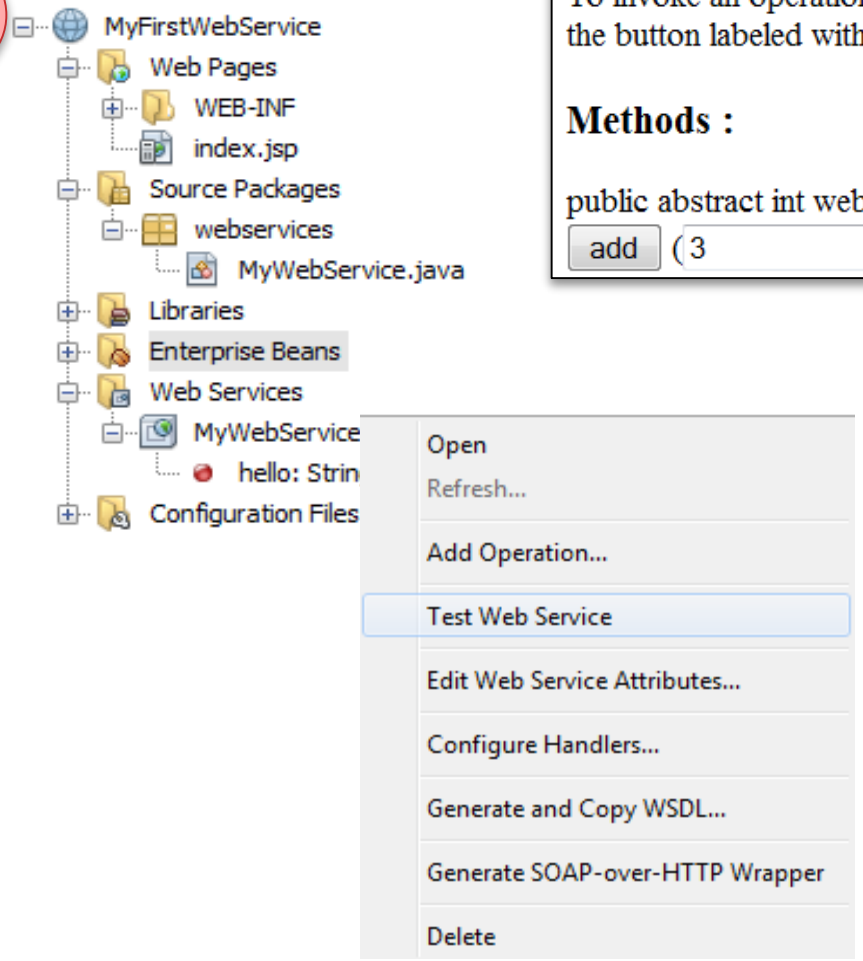
Servicios Web SOAP

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

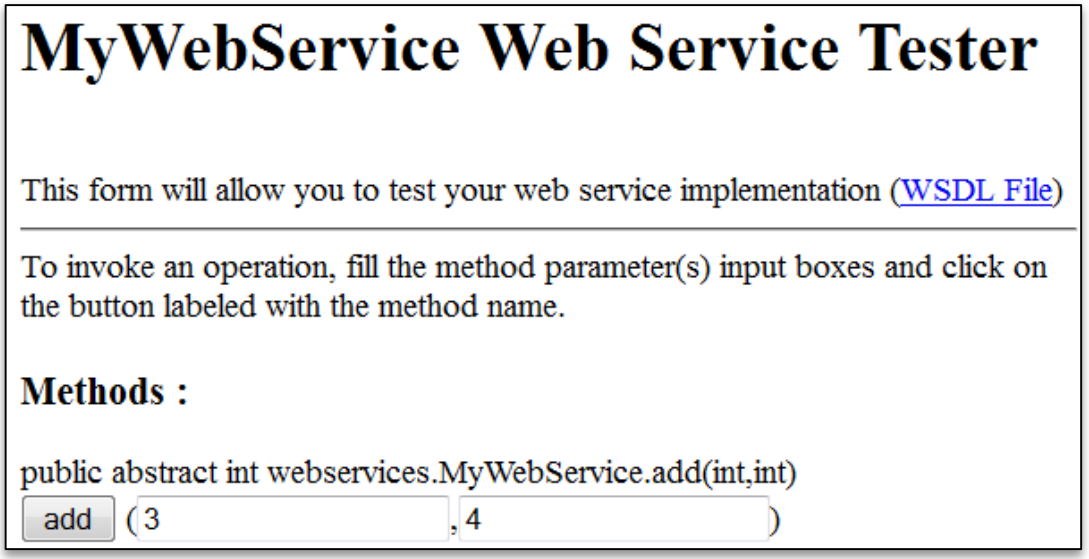
```
@WebService(serviceName = "MyWebService")
@Stateless()
public class MyWebService {

@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

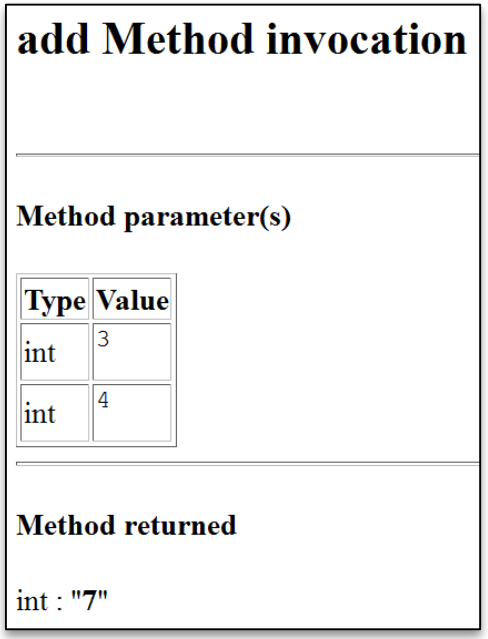
1



2

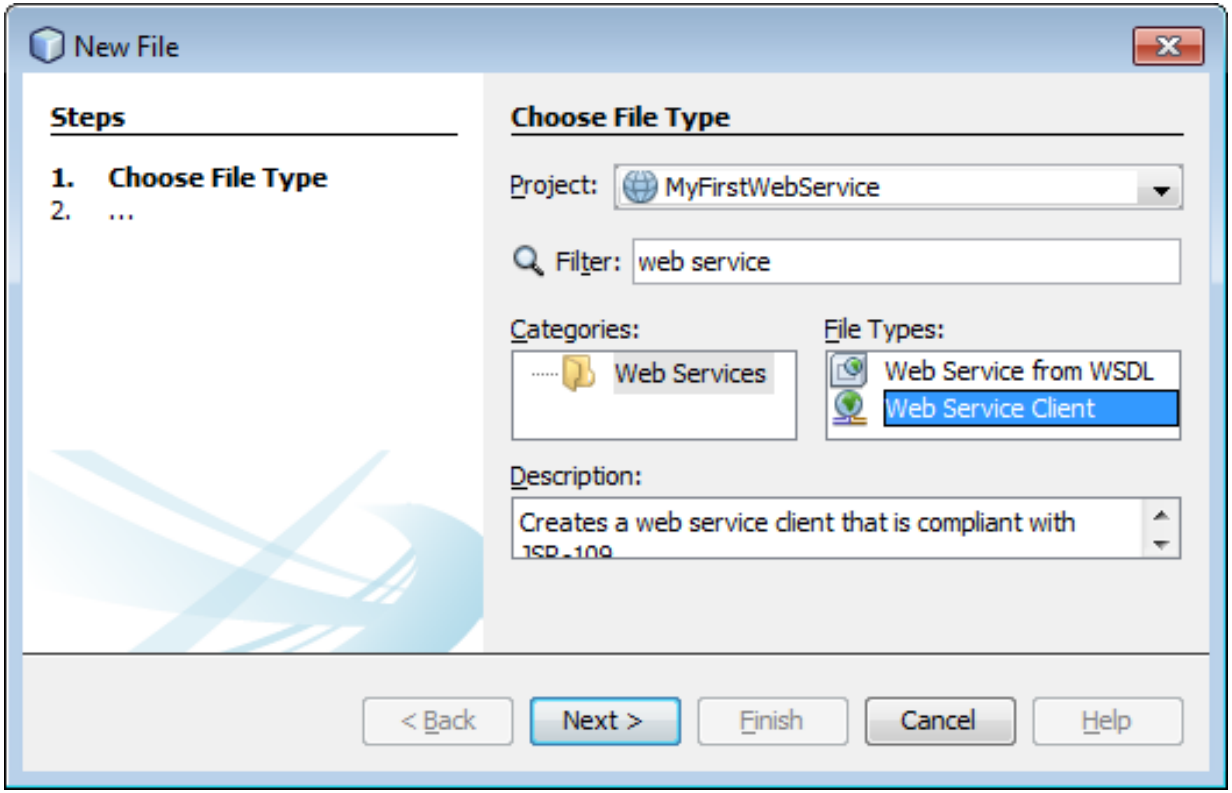


3



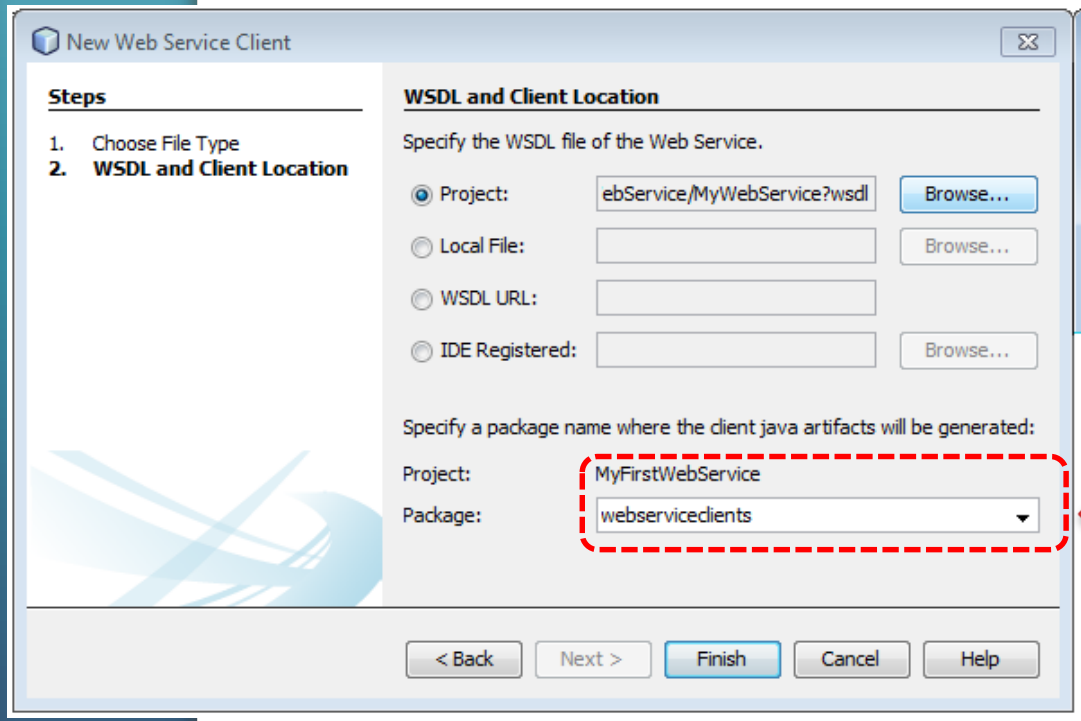
Servicios Web SOAP

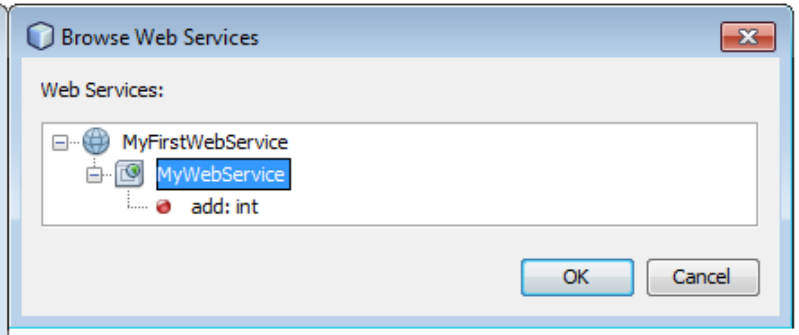
- Crear cliente en [Aplicación Web, Aplicación Java]



Servicios Web SOAP

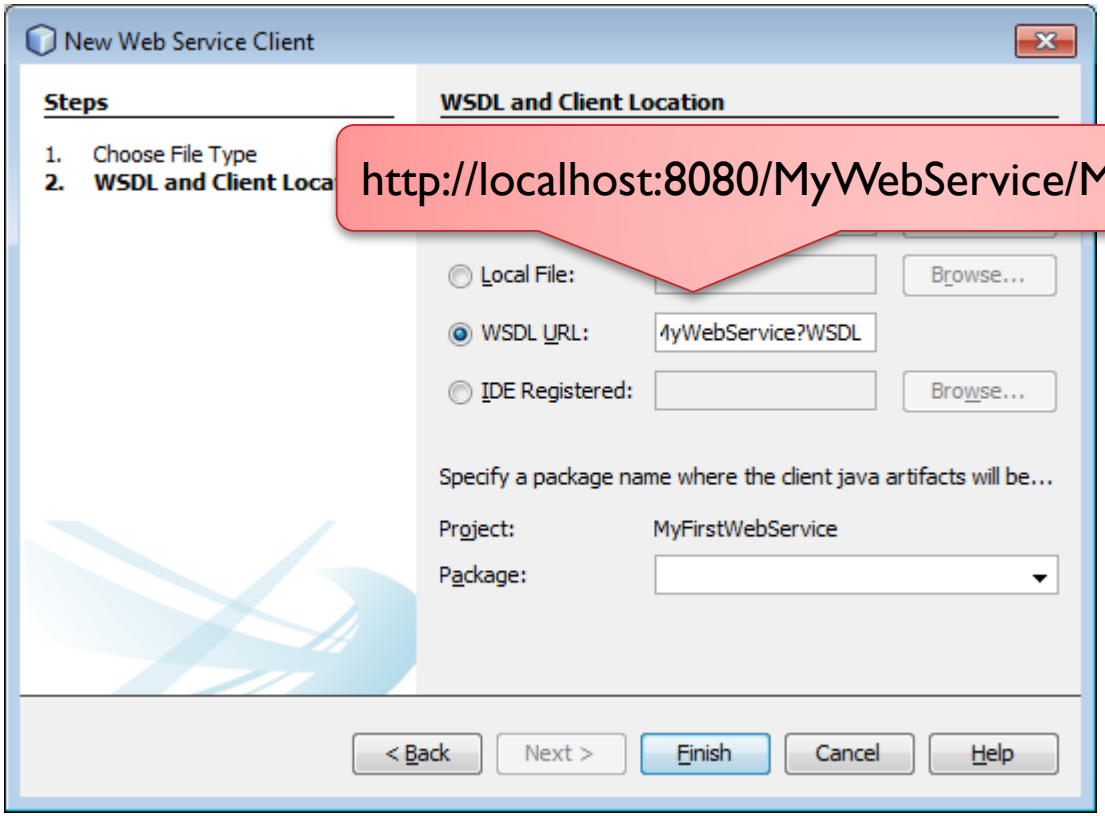
- Crear cliente de servicio web ubicado en el mismo proyecto





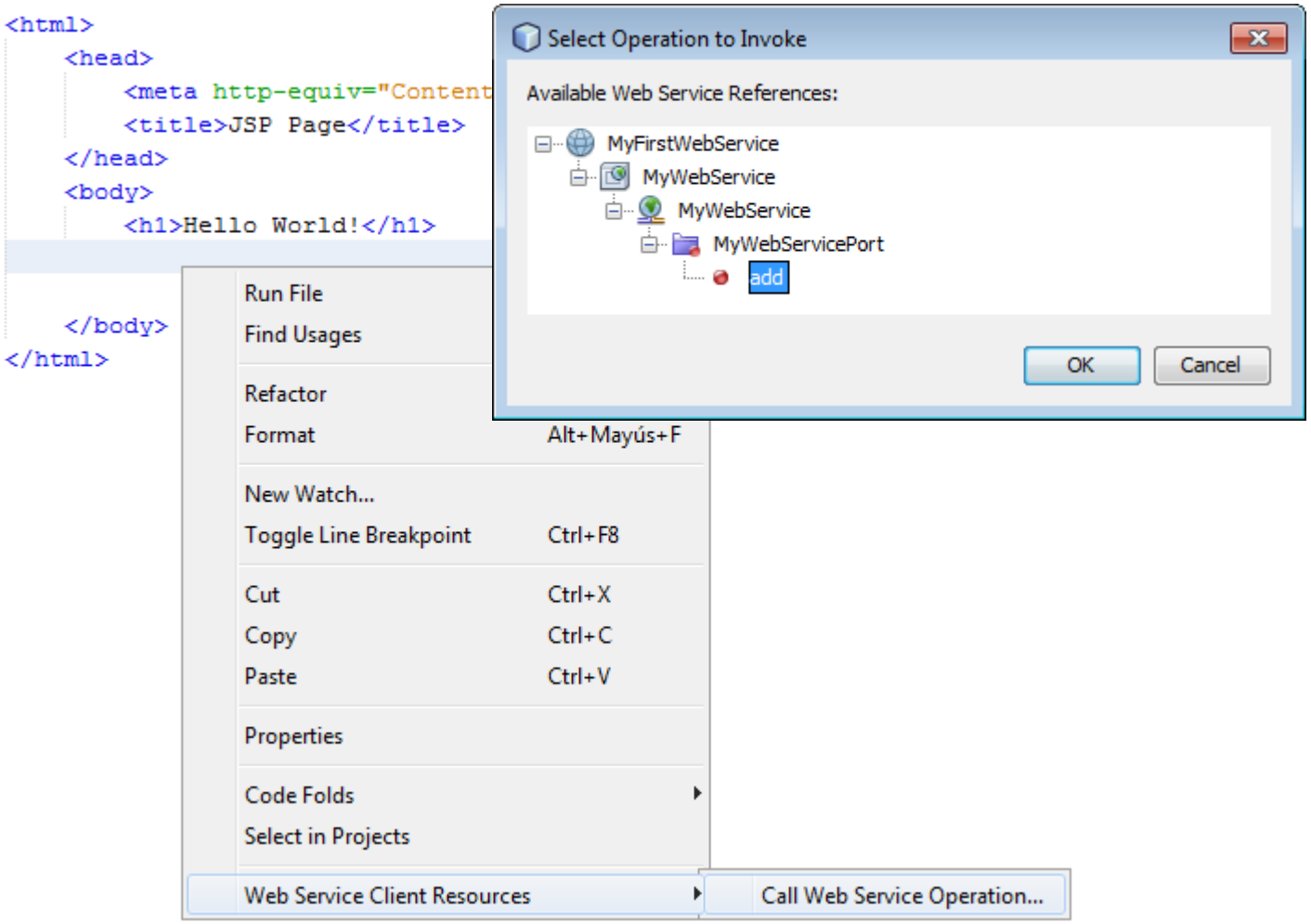
Servicios Web SOAP

- Crear cliente de servicio web desplegado

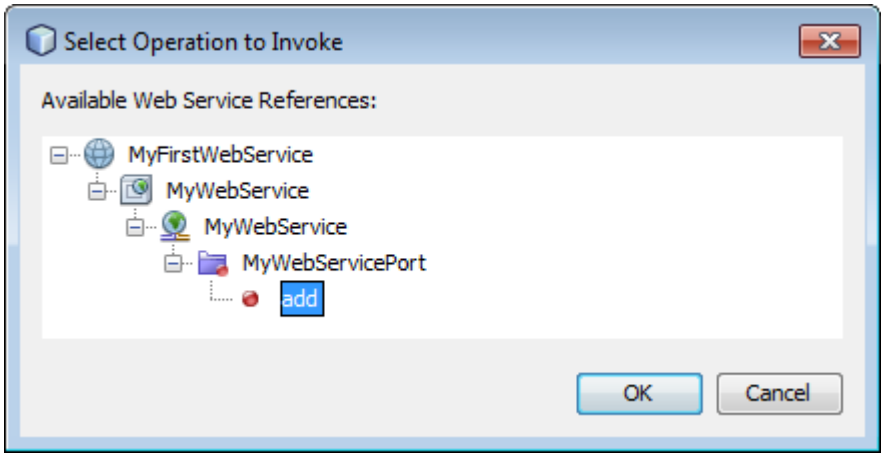


Servicios Web SOAP

- Agregar operación del Servicio Web



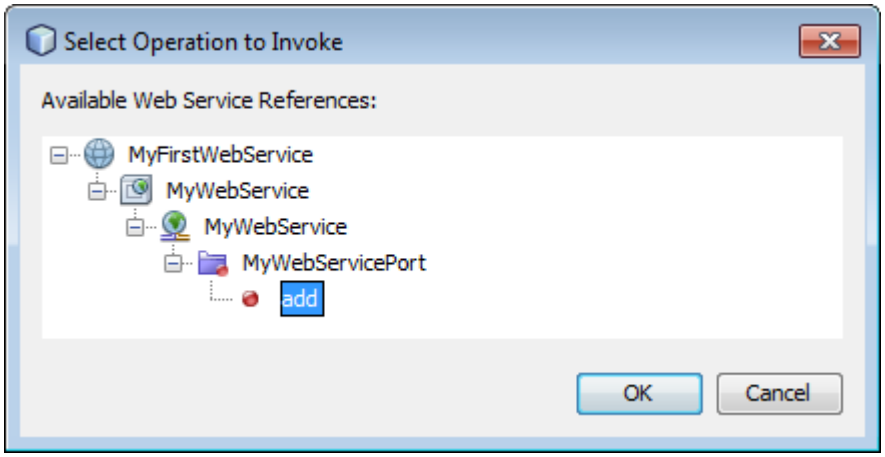
Servicios Web SOAP




```
<body>
        <h1>Hello World!</h1>

<%
    try {
            webserviceclients.MyWebService_Service service =
                                    new webserviceclients.MyWebService_Service();
            webserviceclients.MyWebService port = service.getMyWebServicePort();
            int i = 0;
            int j = 0;
            int result = port.add(i, j);
            out.println("Result = "+result);
    } catch (Exception ex) {
}
    %>

</body>
```

JSP

Servicios Web SOAP




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

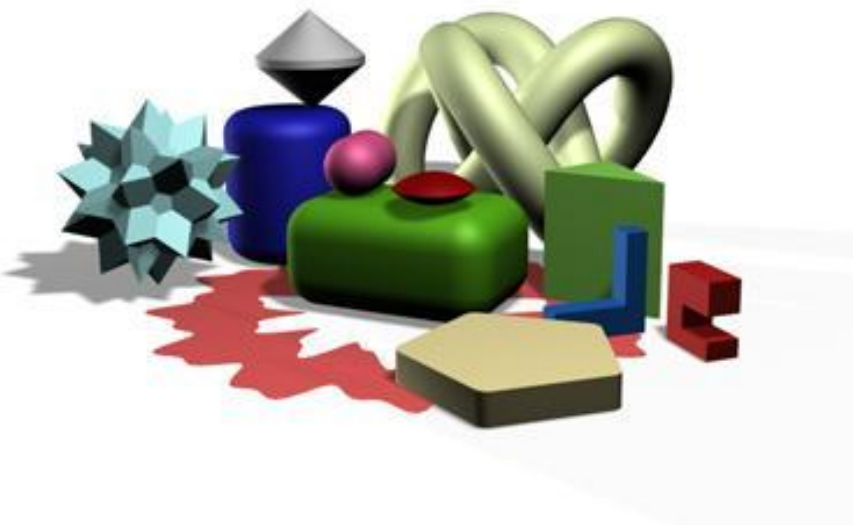
```
private static int adding(int i, int j) {
        webservices.MyWebService_Service service = new
                                                webservices.MyWebService_Service();
        webservices.MyWebService port = service.getMyWebServicePort();
        return port.add(i, j);
    }
```

Aplicación Java

Servicios Web SOAP:
Enviando y recibiendo objetos (complejos)

$$z = \underbrace{a}_{\text{Real Part}} + \underbrace{bi}_{\text{Imaginary Part}}$$

- WebCalculator
  - Web Pages
  - Source Packages
    - webservices
      - Calculator.java
      - ComplexNumber.java
  - Libraries
  - Enterprise Beans
  - Web Services
    - Calculator
      - add: ComplexNumber



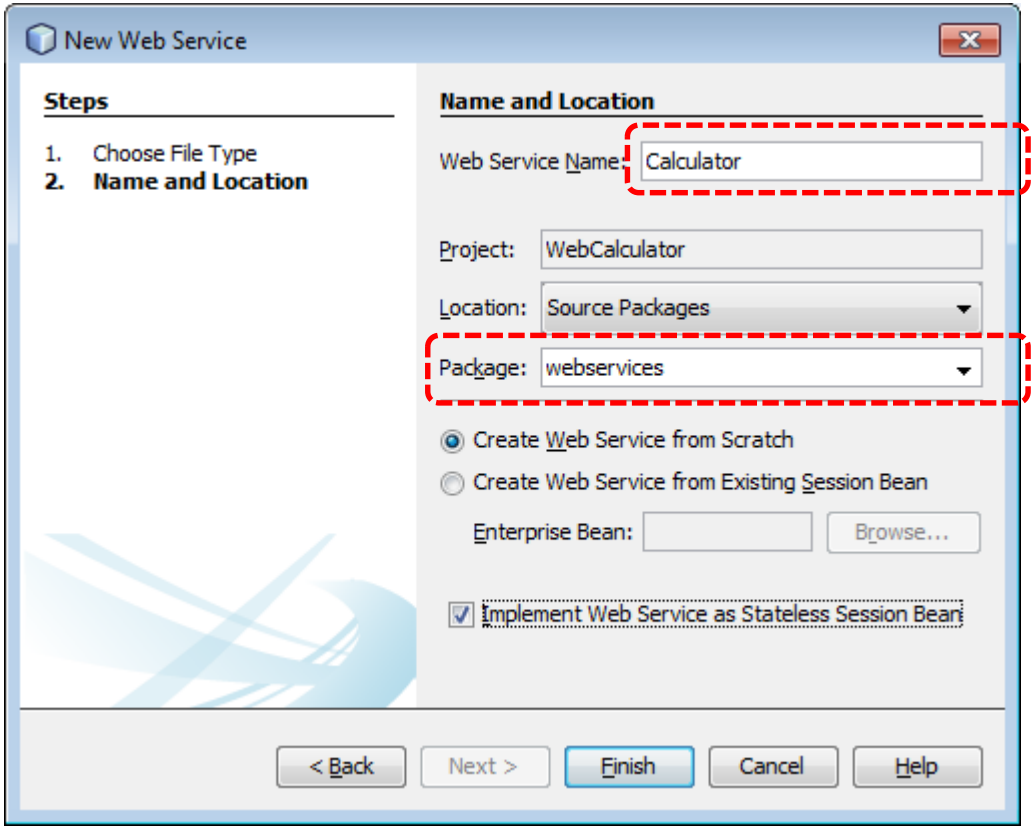
- WebCalculatorClient
  - Source Packages
    - META-INF
    - META-INF.wsdl.localhost_8080.Calculator
    - webcalculatorclient
      - WebCalculatorClient.java
  - Generated Sources (jax-ws)
  - Web Service References
    - Calculator
      - Calculator
        - CalculatorPort
          - add

NetBeans

- WebCalculator
- WebCalculatorClient

Servicios Web SOAP:
Enviando y recibiendo objetos (complejos)

$$z = \underbrace{a}_{\text{Real Part}} + \underbrace{bi}_{\text{Imaginary Part}}$$

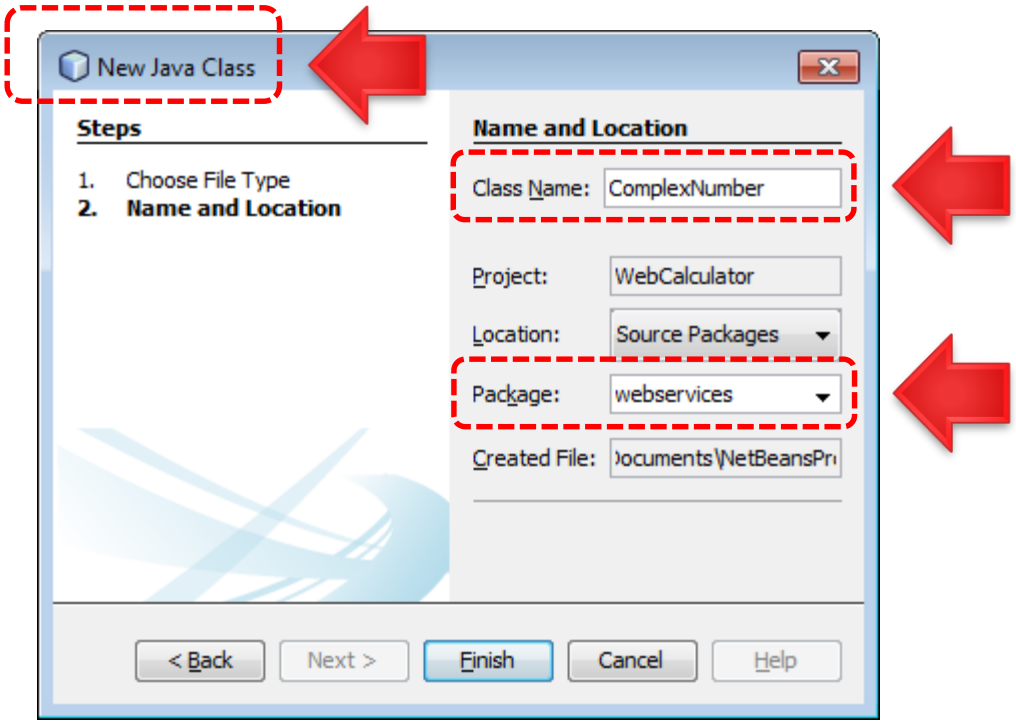


NetBeans

WebCalculator
WebCalculatorClient

Servicios Web SOAP:

Enviando y recibiendo objetos (complejos)



NetBeans

WebCalculator
WebCalculatorClient

Servicios Web SOAP:

Enviando y recibiendo objetos (complejos)

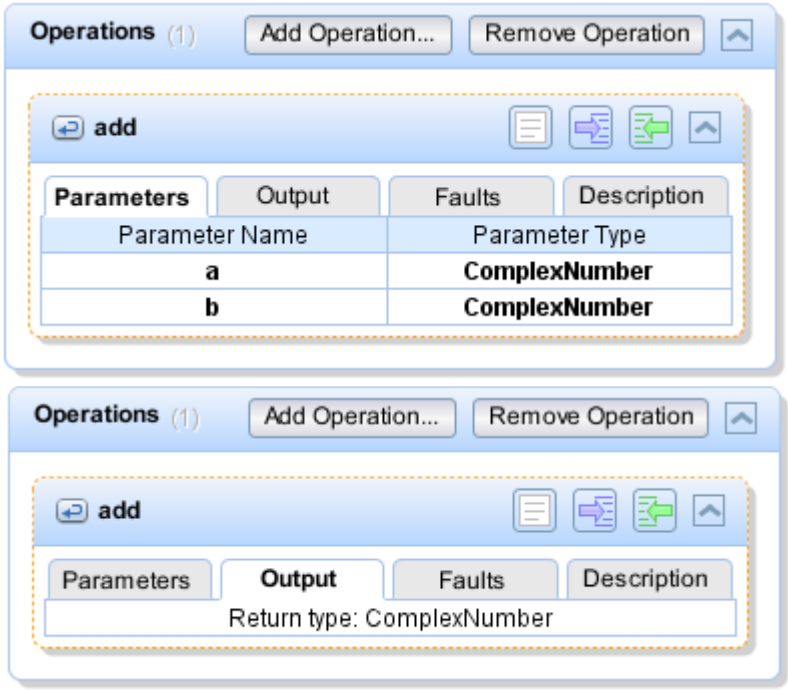
$$z = a + bi$$

Real Part — Imaginary Part

1

```
public class ComplexNumber {
    int i;
    int j;

    public ComplexNumber() {
        i=0;
        j=0;
    }
    public int getI() {
        return i;
    }

    public void setI(int i) {
        this.i = i;
    }

    public int getJ() {
        return j;
    }

    public void setJ(int j) {
        this.j = j;
    }

}
```

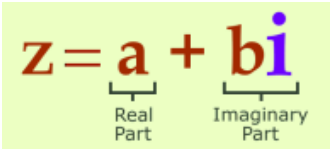
2



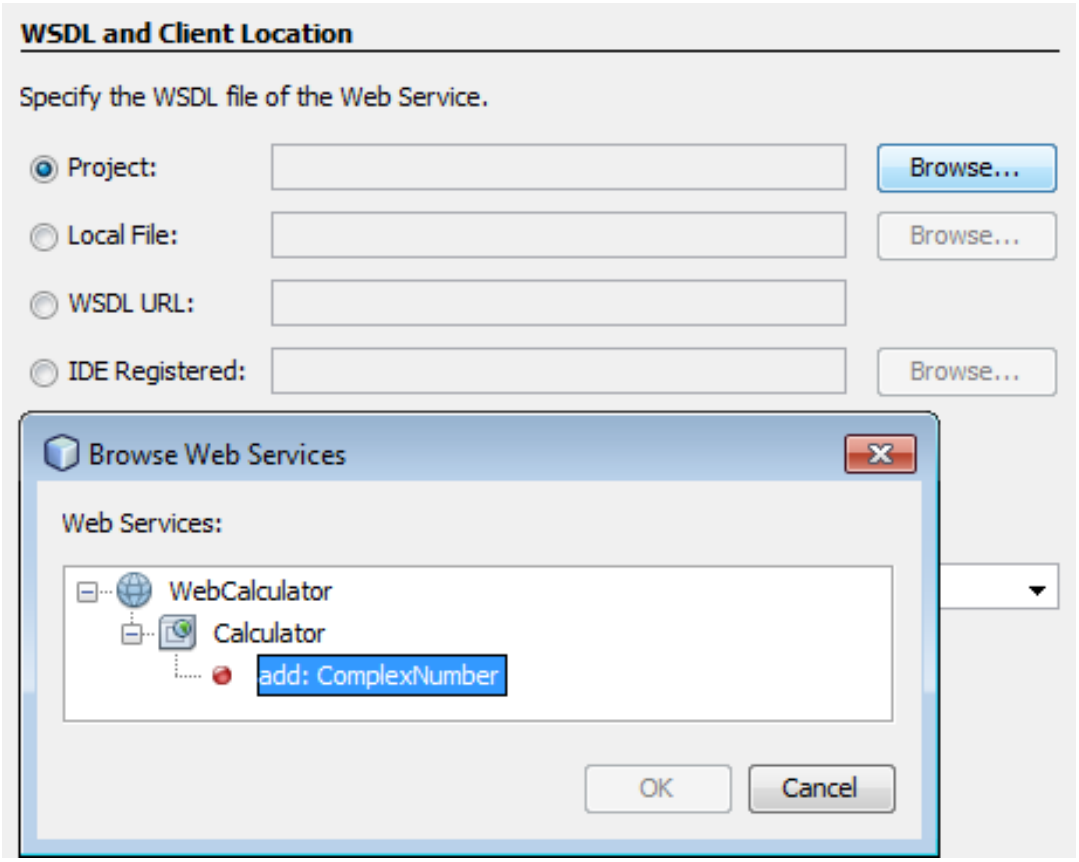
3

```
@WebMethod(operationName = "add")
public ComplexNumber add(@WebParam(name = "a") ComplexNumber a,
                         @WebParam(name = "b") ComplexNumber b) {
    ComplexNumber c = new ComplexNumber();
    c.setI(a.getI()+b.getI());
    c.setJ(a.getJ()+b.getJ());
    return c;
}
```

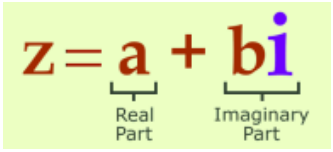
Servicios Web SOAP:
Enviando y recibiendo objetos (complejos)



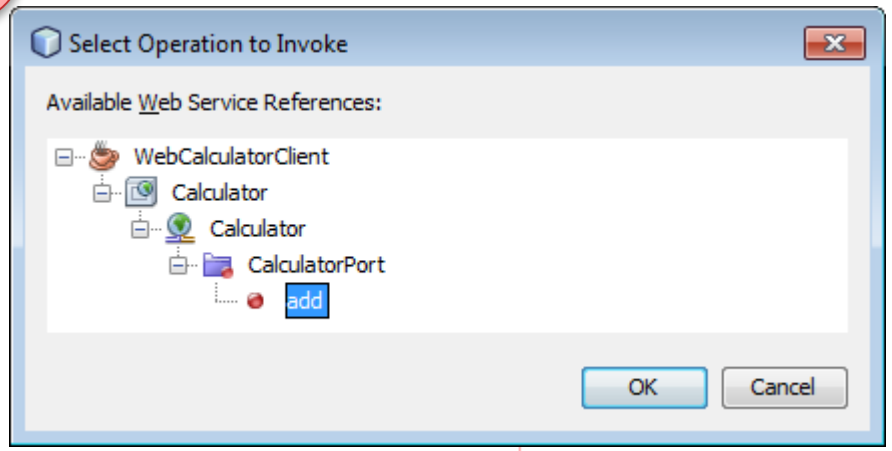
- Aplicación java: **WebCalculatorClient**



Servicios Web SOAP:
Enviando y recibiendo objetos (complejos)

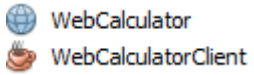


1



2

```
public static void main(String[] args) {
    ComplexNumber a = new ComplexNumber();
    ComplexNumber b = new ComplexNumber();
    a.setI(1);
    b.setI(2);  // 3

    a.setJ(3);
    b.setJ(4);  // 7
    ComplexNumber c= add(a,b);
    System.out.println(c.getI() + " " + c.getJ());  // 3 7
}

private static ComplexNumber add(webservices.ComplexNumber a, webservices.ComplexNumber b) {
    webservices.Calculator_Service service = new webservices.Calculator_Service();
    webservices.Calculator port = service.getCalculatorPort();
    return port.add(a, b);
}
```



Consider "Martin Lawrence" as your data

SOAP

Client

+

=

<- ->

Server

Data + SOAP Standards

Your data sent to SERVER would become huge as Big mama

REST

Client

<- ->

Rest is like sending the DATA as such

Server

Servicios Web RESTful

REST: REpresentational State Transfer

- No es un estándar.
- Usa métodos de HTTP (GET, POST, PUT, etc.)
- Cada operación en un servicio web RESTful es identificada por un URL único.
  - Eso los hace más rápidos que los servicios web SOAP.

Servicios Web RESTful

Mensajes en servicios web RESTful no requieren ser empaquetados en "sobres", como es el caso de servicios web SOAP.

Servicios web SOAP regresan datos en XML.

Servicios web RESTful regresan datos en XML, JSON, HTML y texto plano

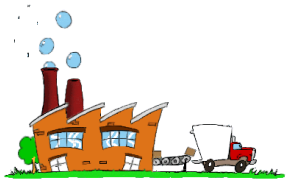
¿Quién usa servicios web RESTful?

REST Business Logic

Anotaciones

- **@Path** – define la ruta de acceso del servicio web.

- **@GET / @POST / @PUT / @DELETE**
  Indica que el método va a responder a una solicitud HTTP Get / Post / Put / Delete



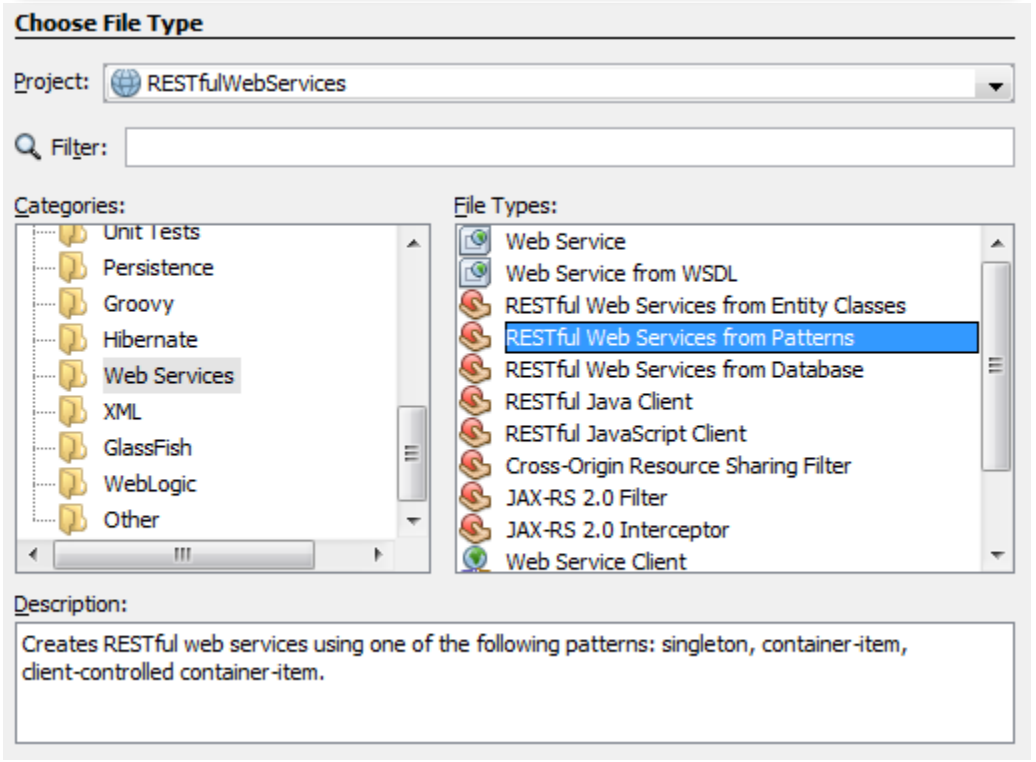
- **@Produces** – indica el tipo de dato que produce un cierto método.

- **@Consumes** – indica el tipo de dato que consume un cierto método.

Mi primer Servicio Web RESTful

0 Crear aplicación *Web* ***RESTfulWebServices***

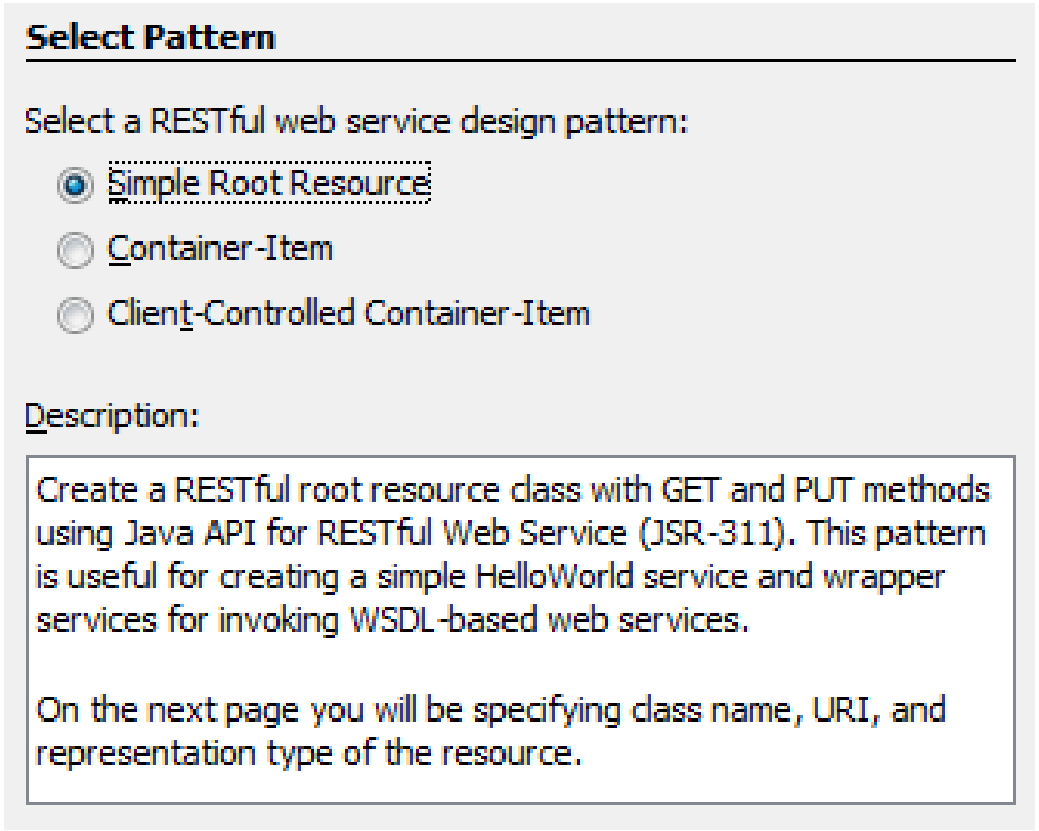
1 Crear Servicio *Web* ***RESTful***



Mi primer Servicio Web RESTful

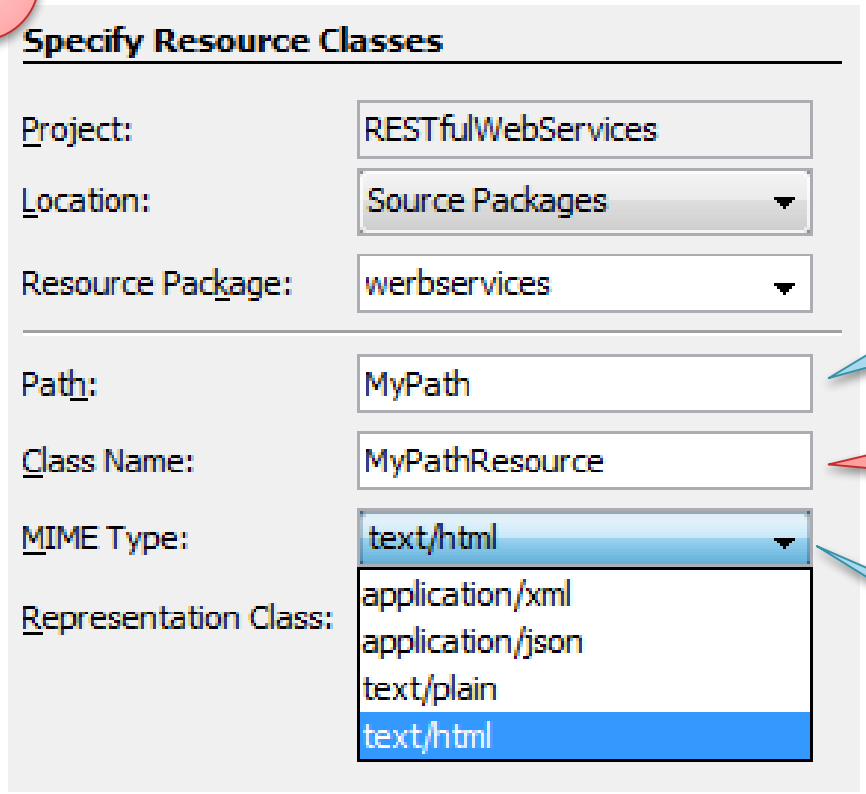
2

Seleccionar Patrón de Diseño



Mi primer Servicio Web RESTful

3



Parte de la URL por donde se consumirá el servicio web

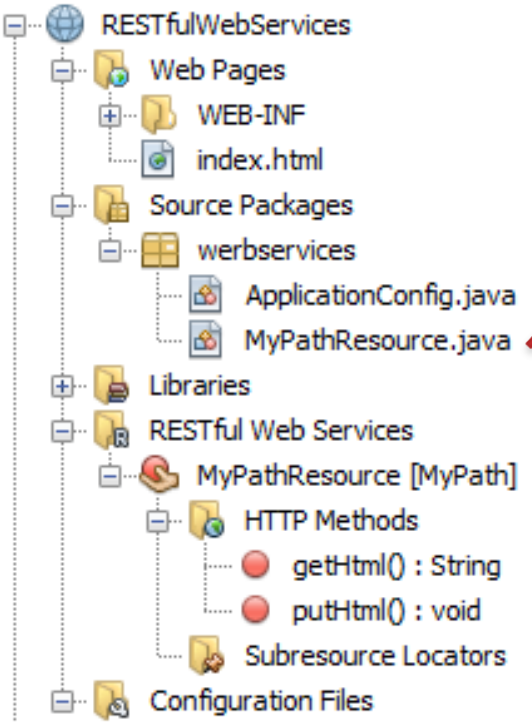
Clase donde se implementarán los métodos del servicio web

Tipo de datos a consumir/producir

Mi primer Servicio Web RESTful

4

Implementar el servicio web




```
@GET
@Produces("text/html")
public String getHtml() {
    //TODO return proper representation object
    throw new UnsupportedOperationException();
}
```

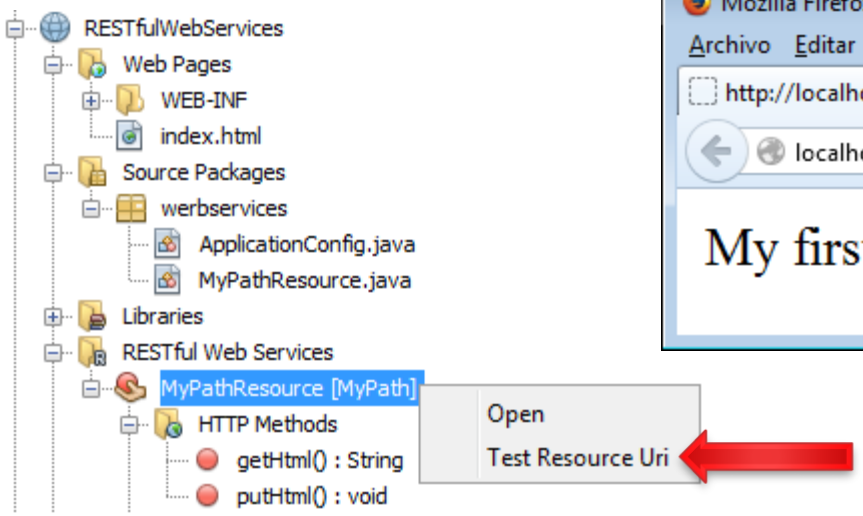
```
@GET
@Produces("text/html")
public String getHtml() {

    return "My first RESTful web service";
}
```

```
@PUT
@Consumes("text/html")
public void putHtml(String content) {
        System.out.println(content);
}
```

Mi primer Servicio Web RESTful

5

Probar el servicio web



http://localhost:8080/RESTfulWebServices/webresources/MyPath

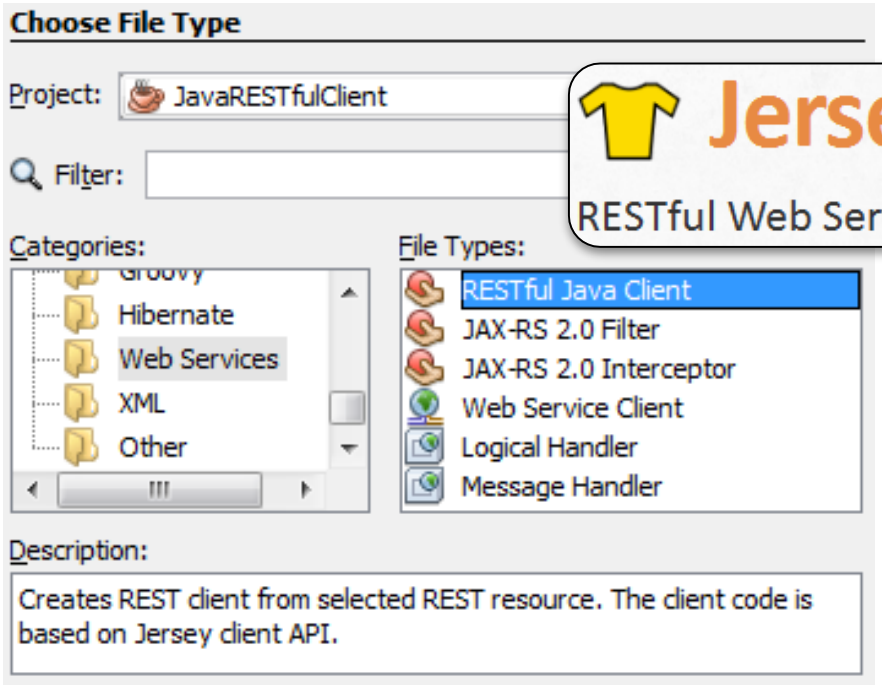
NetBeans

RESTfulWebServices
GlassFish

Cliente de Servicios Web RESTful
Aplicación Java

0 Crear Aplicación *Java* ***JavaRESTfulClient***

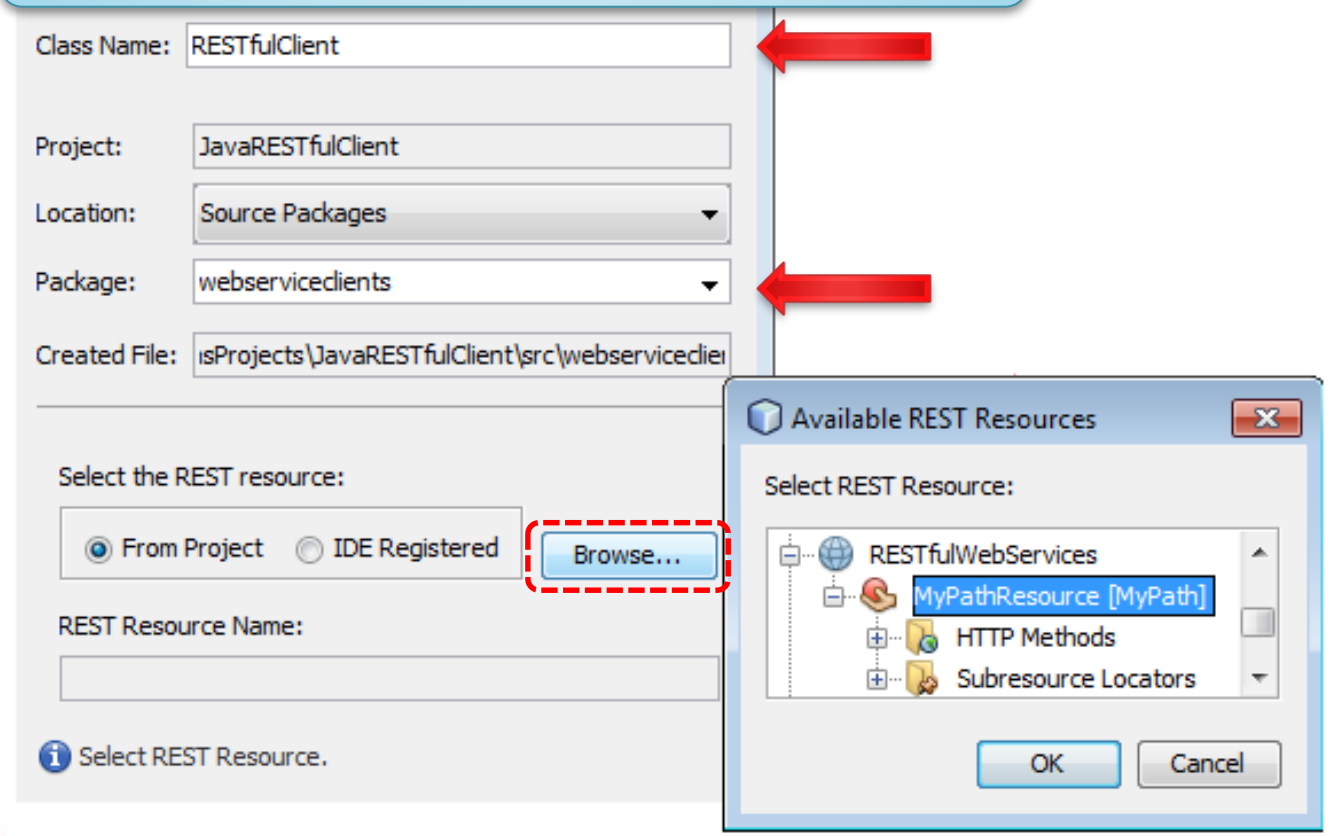
I Crear ***RESTful Java Client***



Cliente de Servicios Web RESTful
Aplicación Java

2

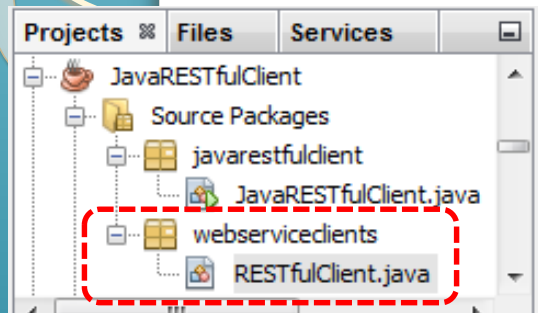
Seleccionar ***Servicio Web RESTful***

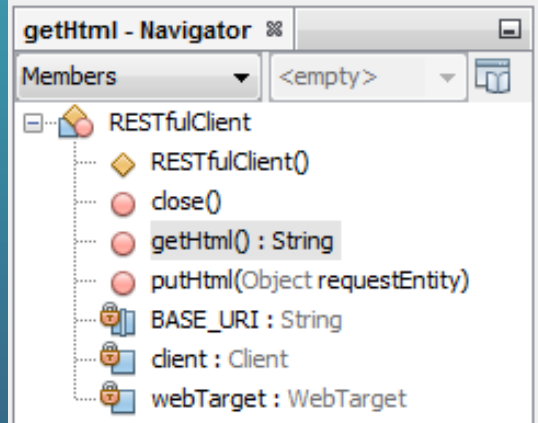


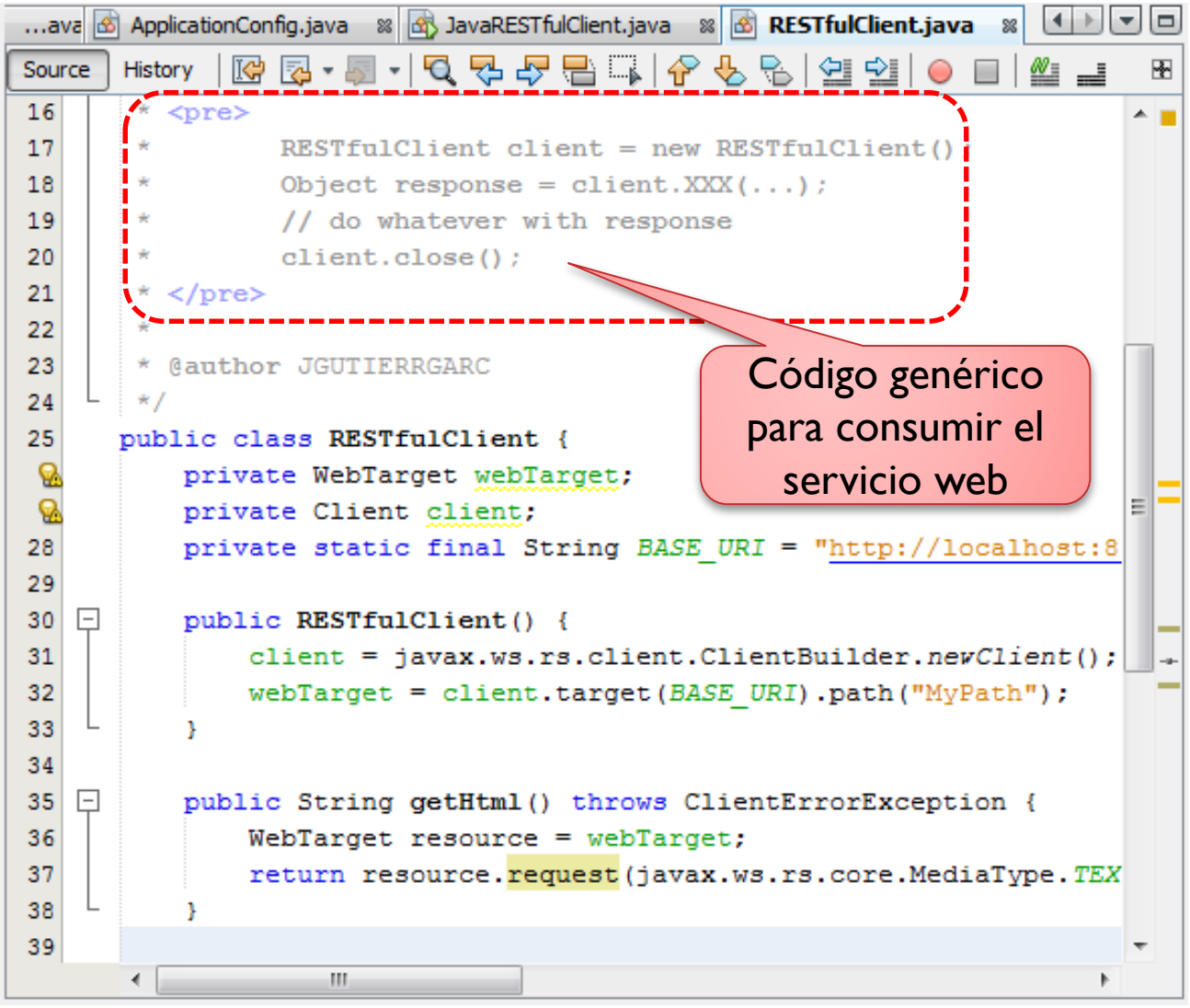
Cliente de Servicios Web RESTful
Aplicación Java

3

Invocar ***Servicio Web RESTful***





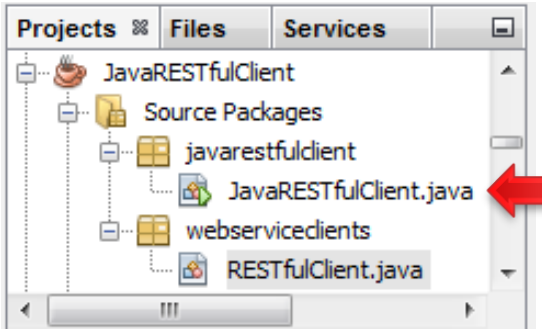


Código genérico para consumir el servicio web

Cliente de Servicios Web RESTful
Aplicación Java

4

Personalizar y usar *código genérico para consumir* ***Servicio Web RESTful***




```
package javarestfulclient;

import webserviceclients.RESTfulClient;

public class JavaRESTfulClient {

    public static void main(String[] args) {
        RESTfulClient client = new RESTfulClient();
        Object response = client.getHtml();
        System.out.println(response);
        client.putHtml("Hi There!");
        client.close();
    }

}
```

NetBeans

JavaRESTfulClient

Cliente de Servicios Web RESTful

Aplicación Web


```
function loadNewContent(id, target) {
        var ajaxRequest;
        if (window.XMLHttpRequest){
            ajaxRequest=new XMLHttpRequest(); // IE7+, Firefox, Chrome, Opera, Safari
        } else {
            ajaxRequest=new ActiveXObject("Microsoft.XMLHTTP"); // IE6, IE5
        }
        ajaxRequest.onreadystatechange = function(){
            if (ajaxRequest.readyState==4 && ajaxRequest.status==200){
                document.getElementById(id).innerHTML=ajaxRequest.responseText;
            }
        }
        ajaxRequest.open("GET", target, true /*async*/);
        ajaxRequest.send();
 }
```

Cliente de Servicios Web RESTful

Aplicación Web

id: nombre del componente a modificar. Normalmente definido con etiquetas o

method: *GET, PUT, etc.*

target: URL del servicio web RESTful

msg: Mensaje

```
function callRESTfulWebService(id, method, target, msg) {
        var ajaxRequest;
        if (window.XMLHttpRequest){
            ajaxRequest=new XMLHttpRequest(); // IE7+, Firefox, Chrome, Opera, Safari
        } else {
            ajaxRequest=new ActiveXObject("Microsoft.XMLHTTP"); // IE6, IE5
        }
        ajaxRequest.onreadystatechange = function(){
            if (ajaxRequest.readyState==4 &&
               (ajaxRequest.status==200 || ajaxRequest.status==204)){
                document.getElementById(id).innerHTML=ajaxRequest.responseText;
            }
        }
        ajaxRequest.open(method, target, true /*async*/);
        ajaxRequest.setRequestHeader("Content-Type", "text/html");
        ajaxRequest.send(msg);
    }
```

El servidor ha cumplido la petición, pero no necesita regresar un valor.

Cliente de Servicios Web RESTful

Aplicación Web

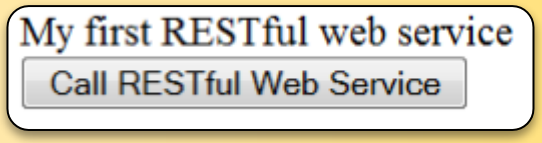
WebRESTfulClient
GlassFish

0 Crear Aplicación Web ***WebRESTfulClient***

1 Crear un <script> con la función
callRESTfulWebService(id, method, target, msg)
dentro de index.html

2 Insertar botón en index.html

```
onclick="callRESTfulWebService(
            'MyDiv',
            'GET',
            'http://localhost:8080/RESTfulWebServices/webresources/MyPath',
            ' ')" />
```



Servicios Web RESTful
Paso de parámetros

@**QueryParam**("name")String name

GET

```
@GET
@Produces("text/html")
public String getHtml(   @QueryParam("name")String name,
                         @QueryParam("age")String age) {
        return "Hi "+ name+ " Age: "+age;
}
```

```
<form action="http://localhost:8080/RESTfulWebServices/webresources/MyPath">
    <input type="text" name="name" value="Juan" />
    <input type="text" name="age" value="44" />
    <input type="submit" value="Send" />
</form>
```

Servicios Web RESTful
Paso de parámetros

@**FormParam**("name")String name

POST

```
@POST
@Produces("text/html")
public String postHtml( @FormParam("name")String name,
                        @FormParam("age")String age) {
        return "Hi "+ name+ " age: "+age;
}
```

```
<form action="http://localhost:8080/RESTfulWebServices/webresources/MyPath"
        method="POST">
    <input type="text" name="name" value="Juan" />
    <input type="text" name="age" value="44" />
    <input type="submit" value="Send" />
</form>
```

Servicios Web RESTful
Paso de parámetros

@**PathParam**("name")String name

```
@Path("MyPath/{username}")
class MyPathResource{

    @GET
    @Produces("text/plain")
     public String getText(@PathParam("username")String username){
         return "UserName:"+username;
      }

}
```

http://localhost:8080/RESTfulWebServices/webresources/MyPath/Juan

Servicios Web RESTful
Práctica de Laboratorio

Servicio web RESTful – Cambio de tipo de monedas

Servicio web SOAP Currency Convertor - ConversionRate

Currency I have	Amount	Currency I want	Amount
US Dollar	10.5	British Pound	6.32835

Mexican Peso
British Pound
Canadian Dollar
US Dollar

NetBeans

CurrencyClient
CurrencyWebServices
GlassFish

Servicios Web RESTful

Práctica de Laboratorio

Tips

AJAX
Asynchronous Javascript And XML

GET

@**QueryParam**("name")String name

document.getElementById("amount").value

onkeyup / onchange

Aplicaciones de los servicios Web

- Arquitecturas orientadas a servicios
- Grid & Cloud computing
  - Amazon Elastic Compute Cloud (EC2)
  - Amazon Simple Storage Service (S3)
  - Amazon Simple Queue Service (SQS)