

# Construcción de Aplicaciones Empresariales con el Lenguaje de Programación Java

---

## Unidad 11

### Web Services

- 11. Web Services
  - ¿Qué es un Web Service?
  - Tecnologías involucradas
    - HTTP, XML
    - SOAP
    - WSDL
    - UDDI
  - Uso de XML en Java: JAXP, JDOM, JAXB
  - Uso de Web Services en Java: JAX-RPC
  - Construcción de un Cliente de un Web Service
  - Construcción de un Web Service en la plataforma J2EE
    - En la capa Web
    - En un stateless session bean



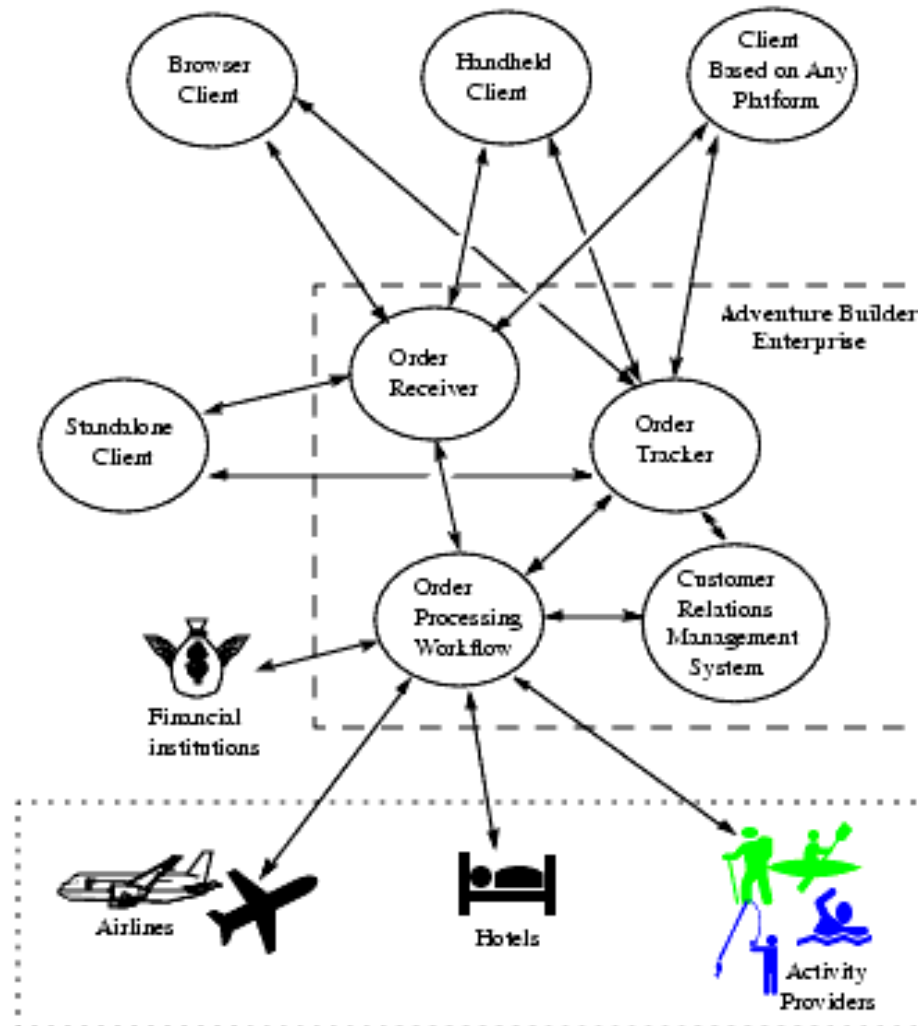
# Ejemplo: “Adventure Builder”

---

- La aplicación “Adventure Builder” (ejemplo tomado del libro “Designing Web Services with the J2EE Platform”) pertenece a una empresa que vende paquetes de aventuras para vacaciones.
- La aplicación incluye un sitio Web para la interacción con sus clientes, y se integra con sus socios comerciales:
  - Instituciones financieras
  - Aerolíneas
  - Hoteles
  - Proveedores de aventuras

# Ejemplo: "Adventure Builder"

## ■ Principales módulos de la aplicación



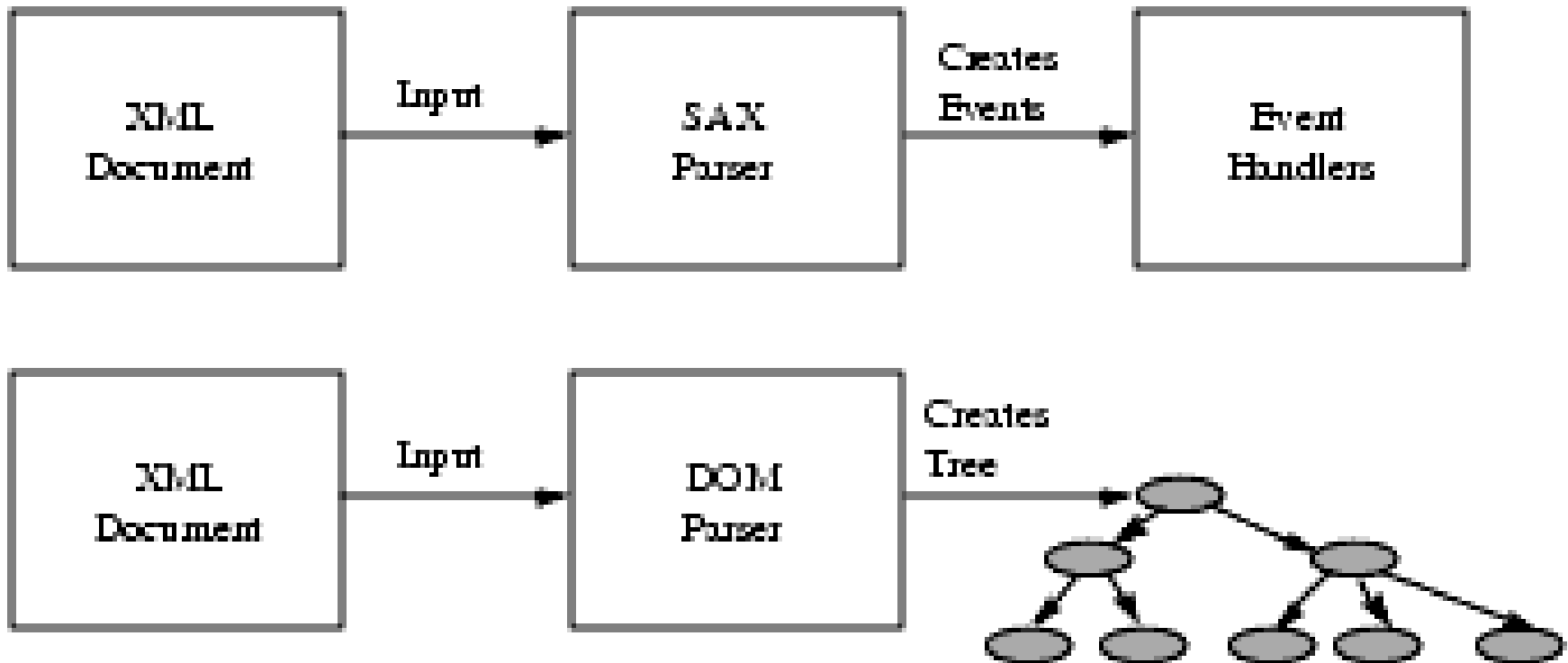
# XML

- XML (<http://www.w3.org/TR/REC-xml>) es un lenguaje orientado a expresar información
- Ejemplo

```
<?xml version="1.0"?>
<authors>
  <name>
    <firstname>Larry</firstname>
    <lastname>Brown</lastname>
  </name>
  <name>
    <firstname>Marty</firstname>
    <lastname>Hall</lastname>
  </name>
  ...
</authors>
```

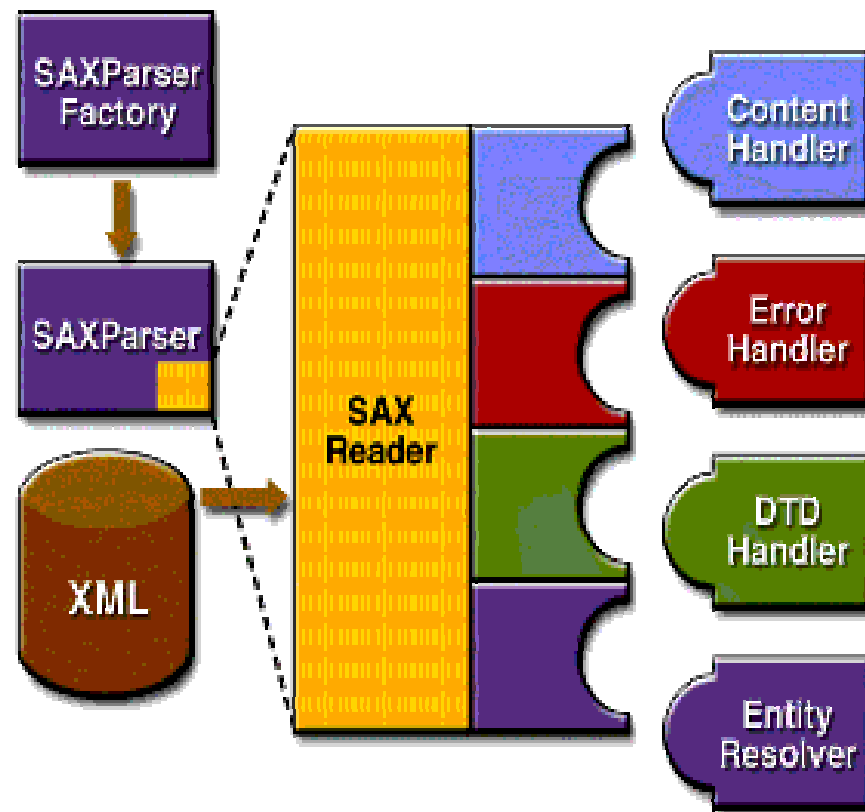
- Aplicaciones:
  - Intercambio de información de manera portable
  - B2B

- JAXP: Java APIs for XML Processing
  - SAX: Simple API for XML
  - DOM: Document Object Model



# SAX – Simple API for XML

- Para parsear un documento XML utilizando SAX, se debe instanciar un **SAXParser**, y luego invocar el método **parse()** proveyendo un **ContentHandler**, **ErrorHandler**, **DTDHandler**, y **EntityResolver**



# SAX - Ejemplo

 SAXdemo.java

```
public class SAXdemo extends DefaultHandler
{
    public static void main(String[] args) throws Throwable
    {
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser saxParser = factory.newSAXParser();
        saxParser.parse(new File(args[0]), new SAXdemo());
    }

    public void startElement(String uri, String localName, String qName, Attributes attributes)
    throws SAXException
    {
        System.out.println("start -> " + localName);
        for (int i=0; i < attributes.getLength(); i++)
        {
            System.out.println("  " + attributes.getLocalName(i) + " = " + attributes.getValue(i));
        }
    }

    public void endElement(String uri, String localName, String qName) throws SAXException
    {
        System.out.println("end -> " + localName);
    }
}
```





# DOM

---

- Para parsear un documento XML utilizando DOM, se debe instanciar un `DocumentBuilder`, luego invocar el método `parse()` para crear el árbol de nodos en memoria, y luego recorrer el árbol según se desee, con la funcionalidad provista por DOM

# DOM - Ejemplo

SAXdemo.java

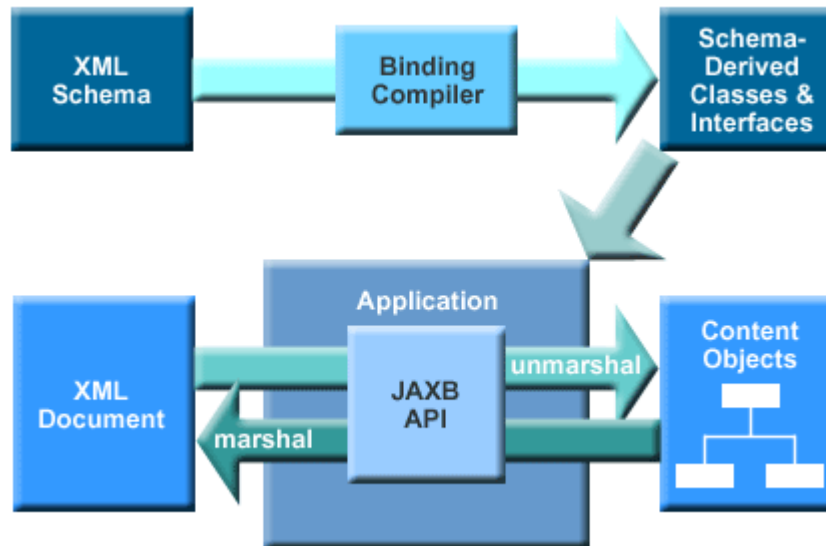
DOMdemo.java

```
public class DOMdemo
{
    public static void main(String[] args) throws Throwable {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse(new File(args[0]));

        NodeList nodes = document.getElementsByTagName("slideshow");
        if (nodes != null && nodes.getLength() > 0)
        {
            Node node = nodes.item(0);
            System.out.println(node.getLocalName());
            NamedNodeMap attributes = node.getAttributes();
            for (int i=0; i < attributes.getLength(); i++)
            {
                Node attribute = attributes.item(i);
                System.out.println("  " + attribute.getLocalName() + " = " +
                                   attribute.getNodeValue());
            }
        }
    }
}
```

# Otras Alternativas

- JAXB: <http://java.sun.com/xml/jaxb>
  - Java Architecture for XML Binding



- JDOM: <http://www.jdom.org>

# Servicios

- SOA: Service-Oriented Architecture
- Objetivo: facilitar la integración entre diferentes agentes de software





# Web Services

---

- SOAP: Simple Object Access Protocol
  - <http://www.w3.org/TR/SOAP/>
- WSDL: Web Services Description Language
  - <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- UDDI: Universal Description, Discovery and Integration of Web Services
  - <http://www.uddi.org/>



# SOAP

---

- Simple Object Access Protocol
- *"SOAP Version 1.2 (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics."*

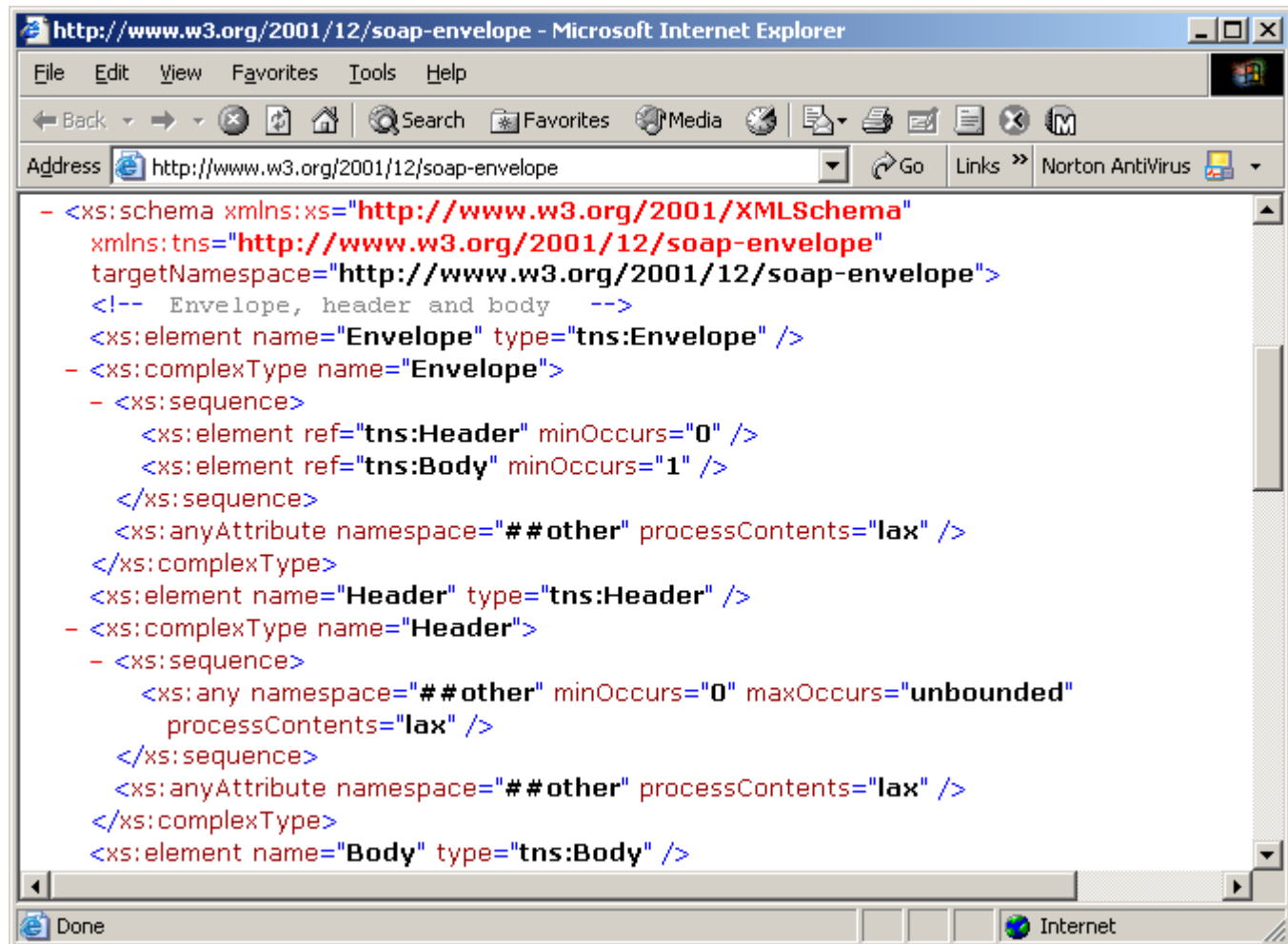
- Protocolo de comunicaciones para aplicaciones
- Formato para el envío de mensajes sobre Internet (HTTP)
- Independiente de plataformas y lenguajes
- Basado en XML

- Un mensaje SOAP es un documento XML con los siguientes elementos:
  - Un sobre (envelope) obligatorio que identifica el documento XML como un mensaje SOAP
  - Un encabezado (header) opcional
  - Un cuerpo (body) obligatorio, que contiene información de la llamada y la respuesta
  - Un elemento de falla (fault) opcional, que contiene información acerca de errores de procesamiento



# Schema SOAP

- Definición elementos en <http://www.w3.org/2001/12/soap-envelope>

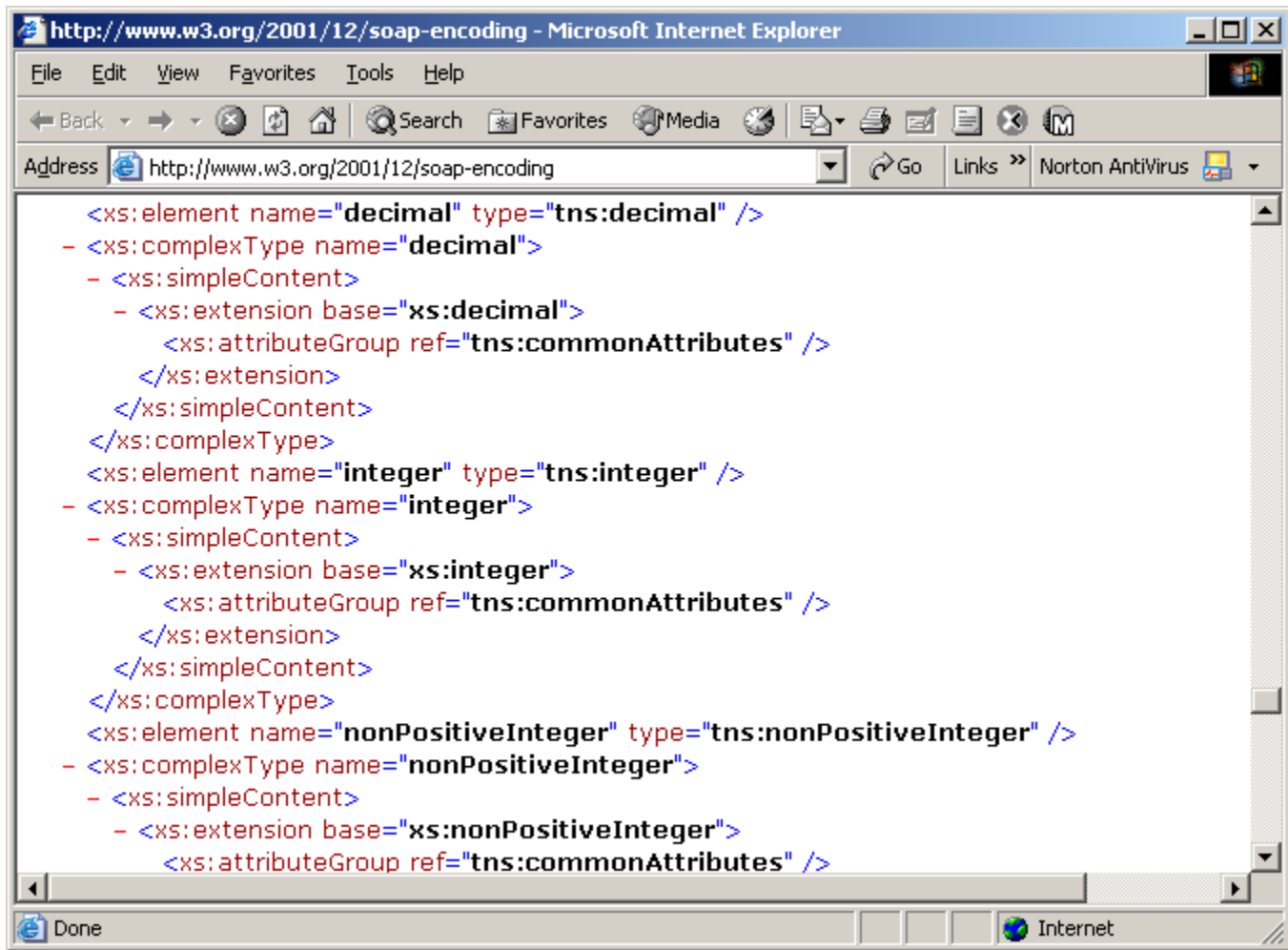


The screenshot shows a Microsoft Internet Explorer window with the address bar displaying <http://www.w3.org/2001/12/soap-envelope>. The main content area displays the XML Schema for the SOAP Envelope, which defines the structure of a SOAP message. The schema includes elements for the Envelope, Header, and Body, along with their respective complex types and sequence constraints.

```
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.w3.org/2001/12/soap-envelope"
  targetNamespace="http://www.w3.org/2001/12/soap-envelope">
  <!-- Envelope, header and body -->
  <xs:element name="Envelope" type="tns:Envelope" />
  - <xs:complexType name="Envelope">
    - <xs:sequence>
      <xs:element ref="tns:Header" minOccurs="0" />
      <xs:element ref="tns:Body" minOccurs="1" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
  <xs:element name="Header" type="tns:Header" />
  - <xs:complexType name="Header">
    - <xs:sequence>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
        processContents="lax" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
  <xs:element name="Body" type="tns:Body" />
```

# Tipos de Datos SOAP

- Definición de tipos de datos en <http://www.w3.org/2001/12/soap-encoding>



The screenshot shows a Microsoft Internet Explorer browser window with the address bar displaying <http://www.w3.org/2001/12/soap-encoding>. The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar contains buttons for Back, Forward, Stop, Home, Search, Favorites, Media, and a Go button. The address bar also includes a Go button, a Links button, and a Norton AntiVirus icon. The main content area displays the XML Schema Definition for SOAP data types, which is a fragment of the SOAP 1.2 specification. The schema defines three data types: decimal, integer, and nonPositiveInteger. Each type is defined as a complex type that extends a base type (xs:decimal, xs:integer, or xs:nonPositiveInteger) and includes a reference to the tns:commonAttributes attribute group. The schema is displayed in a monospaced font with syntax highlighting.

```
<xs:element name="decimal" type="tns:decimal" />
- <xs:complexType name="decimal">
- <xs:simpleContent>
- <xs:extension base="xs:decimal">
  <xs:attributeGroup ref="tns:commonAttributes" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:element name="integer" type="tns:integer" />
- <xs:complexType name="integer">
- <xs:simpleContent>
- <xs:extension base="xs:integer">
  <xs:attributeGroup ref="tns:commonAttributes" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:element name="nonPositiveInteger" type="tns:nonPositiveInteger" />
- <xs:complexType name="nonPositiveInteger">
- <xs:simpleContent>
- <xs:extension base="xs:nonPositiveInteger">
  <xs:attributeGroup ref="tns:commonAttributes" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
```

# Ejemplo SOAP

- El siguiente ejemplo muestra un mensaje de notificación expresado en SOAP
- El mensaje contiene:
  - un header con el nombre local **alertcontrol**
  - Un body con el nombre local **alert**

Example 1: SOAP message containing a SOAP header block and a SOAP body

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

# Un Requerimiento SOAP

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Áke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2001-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2001-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference/>
      </p:return>
    </p:itinerary>
    <q:lodging
      xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>
```

Sample SOAP message for a travel reservation containing header blocks and a body

# Una Respuesta SOAP

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Áke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itineraryClarification
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>
          <p:airportChoices>
            JFK LGA EWR
          </p:airportChoices>
        </p:departing>
      </p:departure>
      <p:return>
        <p:arriving>
          <p:airportChoices>
            JFK LGA EWR
          </p:airportChoices>
        </p:arriving>
      </p:return>
    </p:itineraryClarification>
  </env:Body>
</env:Envelope>
```

SOAP message sent in response to the message in [Example 1](#)

- Web Services Description Language
- *"WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME."*

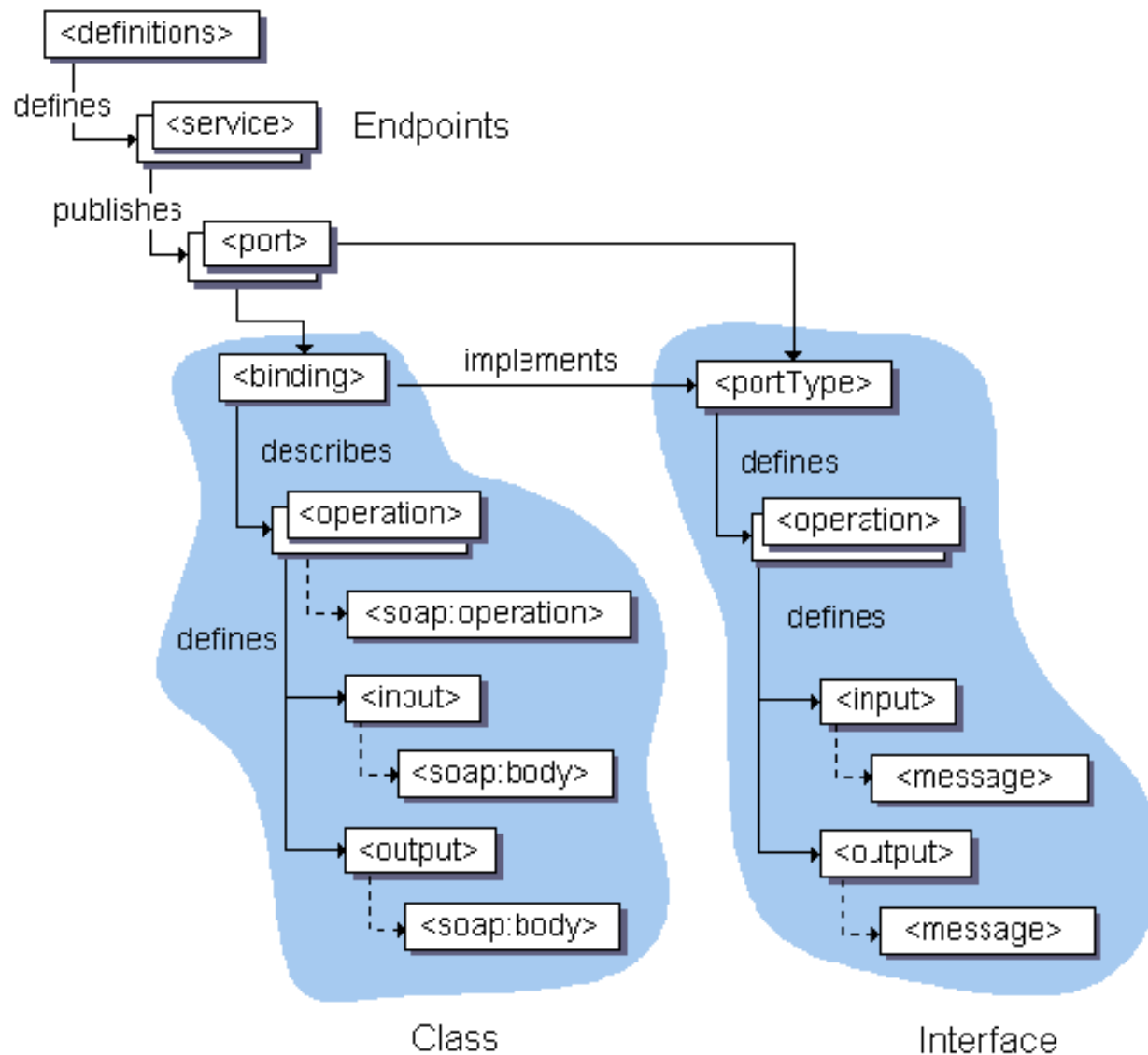


# Estructura de WSDL

---

- Un documento WSDL es un documento XML utilizado para describir un Web Service
- Un documento WSDL define un Web Service utilizando los siguientes elementos:
  - `<types>`: información sobre tipos de datos complejos usados por el Web Service
  - `<message>`: definición abstracta de la información transferida
  - `<operation>`: descripción abstracta de la acción soportada por el servicio
  - `<portType>`: conjunto abstracto de operaciones soportadas por uno o más endpoints
  - `<binding>`: descripción de protocolos de comunicaciones y formatos de datos utilizados
  - `<port>`: especifica un endpoint como una dirección para el binding
  - `<service>`: colección de endpoints o ports
  - `<definitions>`: contiene la definición de uno o más servicios

# Estructura de WSDL





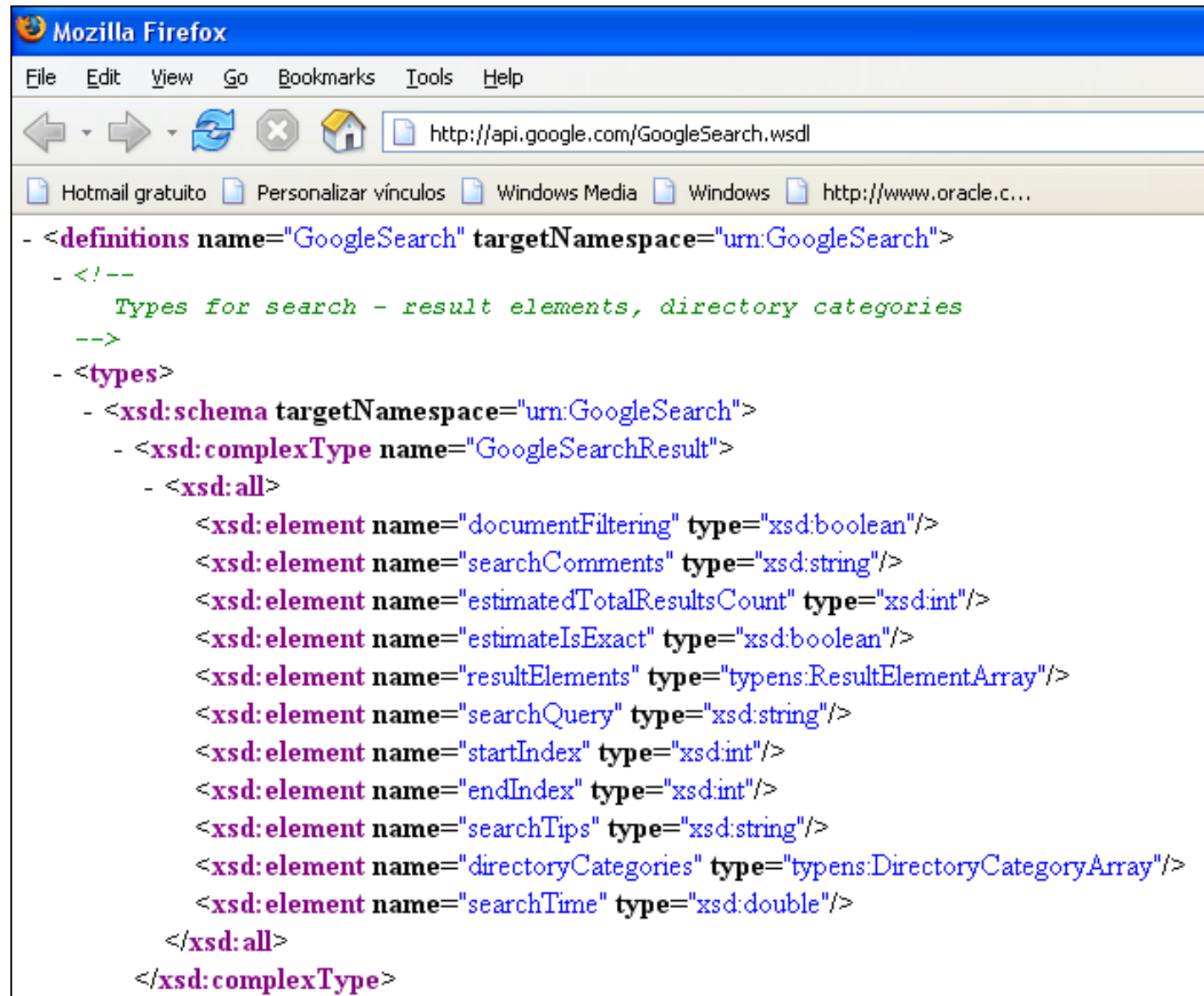


# Ejemplo WSDL

---

- Página WSDL para servicios de búsqueda de Google:  
<http://api.google.com/GoogleSearch.wsdl>
- Permite realizar búsquedas usando el motor de Google
- Retorna respuestas estructuradas
  - Información sobre la respuesta: número estimado de hits, etc.
  - Un arreglo con los hits (el subconjunto indicado en la búsqueda)
  - Por cada hit, retorna información estructurada: resumen, URL, snippet, título, etc.

# GoogleSearch.wsdl - Tipos



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://api.google.com/GoogleSearch.wsdl`. The browser's menu bar includes File, Edit, View, Go, Bookmarks, Tools, and Help. Below the menu bar, there are navigation buttons (back, forward, reload, stop, home) and a search bar. The main content area displays the XML Schema Definition (XSD) for GoogleSearch.wsdl. The schema is titled `<definitions name="GoogleSearch" targetNamespace="urn:GoogleSearch">`. It includes a comment `<!-- Types for search - result elements, directory categories -->` and a `<types>` section. The `<types>` section contains a `<xsd:schema targetNamespace="urn:GoogleSearch">` element, which in turn contains a `<xsd:complexType name="GoogleSearchResult">` element. This complex type is defined with an `<xsd:all>` group containing several elements: `<xsd:element name="documentFiltering" type="xsd:boolean"/>`, `<xsd:element name="searchComments" type="xsd:string"/>`, `<xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>`, `<xsd:element name="estimateIsExact" type="xsd:boolean"/>`, `<xsd:element name="resultElements" type="typens:ResultElementArray"/>`, `<xsd:element name="searchQuery" type="xsd:string"/>`, `<xsd:element name="startIndex" type="xsd:int"/>`, `<xsd:element name="endIndex" type="xsd:int"/>`, `<xsd:element name="searchTips" type="xsd:string"/>`, `<xsd:element name="directoryCategories" type="typens:DirectoryCategoryArray"/>`, and `<xsd:element name="searchTime" type="xsd:double"/>`. The `<xsd:all>` group is closed with `</xsd:all>`, and the complex type is closed with `</xsd:complexType>`.

```
- <definitions name="GoogleSearch" targetNamespace="urn:GoogleSearch">
- <!--
-   Types for search - result elements, directory categories
- -->
- <types>
-   <xsd:schema targetNamespace="urn:GoogleSearch">
-     <xsd:complexType name="GoogleSearchResult">
-       <xsd:all>
-         <xsd:element name="documentFiltering" type="xsd:boolean"/>
-         <xsd:element name="searchComments" type="xsd:string"/>
-         <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
-         <xsd:element name="estimateIsExact" type="xsd:boolean"/>
-         <xsd:element name="resultElements" type="typens:ResultElementArray"/>
-         <xsd:element name="searchQuery" type="xsd:string"/>
-         <xsd:element name="startIndex" type="xsd:int"/>
-         <xsd:element name="endIndex" type="xsd:int"/>
-         <xsd:element name="searchTips" type="xsd:string"/>
-         <xsd:element name="directoryCategories" type="typens:DirectoryCategoryArray"/>
-         <xsd:element name="searchTime" type="xsd:double"/>
-       </xsd:all>
-     </xsd:complexType>
```

# GoogleSearch.wsdl - Tipos

```
- <xsd:complexType name="ResultElement">
  - <xsd:all>
    <xsd:element name="summary" type="xsd:string"/>
    <xsd:element name="URL" type="xsd:string"/>
    <xsd:element name="snippet" type="xsd:string"/>
    <xsd:element name="title" type="xsd:string"/>
    <xsd:element name="cachedSize" type="xsd:string"/>
    <xsd:element name="relatedInformationPresent" type="xsd:boolean"/>
    <xsd:element name="hostName" type="xsd:string"/>
    <xsd:element name="directoryCategory" type="typens:DirectoryCategory"/>
    <xsd:element name="directoryTitle" type="xsd:string"/>
  </xsd:all>
</xsd:complexType>
- <xsd:complexType name="ResultElementArray">
  - <xsd:complexContent>
    - <xsd:restriction base="soapenc:Array">
      <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="typens:ResultElement[]"/>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

# GoogleSearch – Mensajes y Puertos

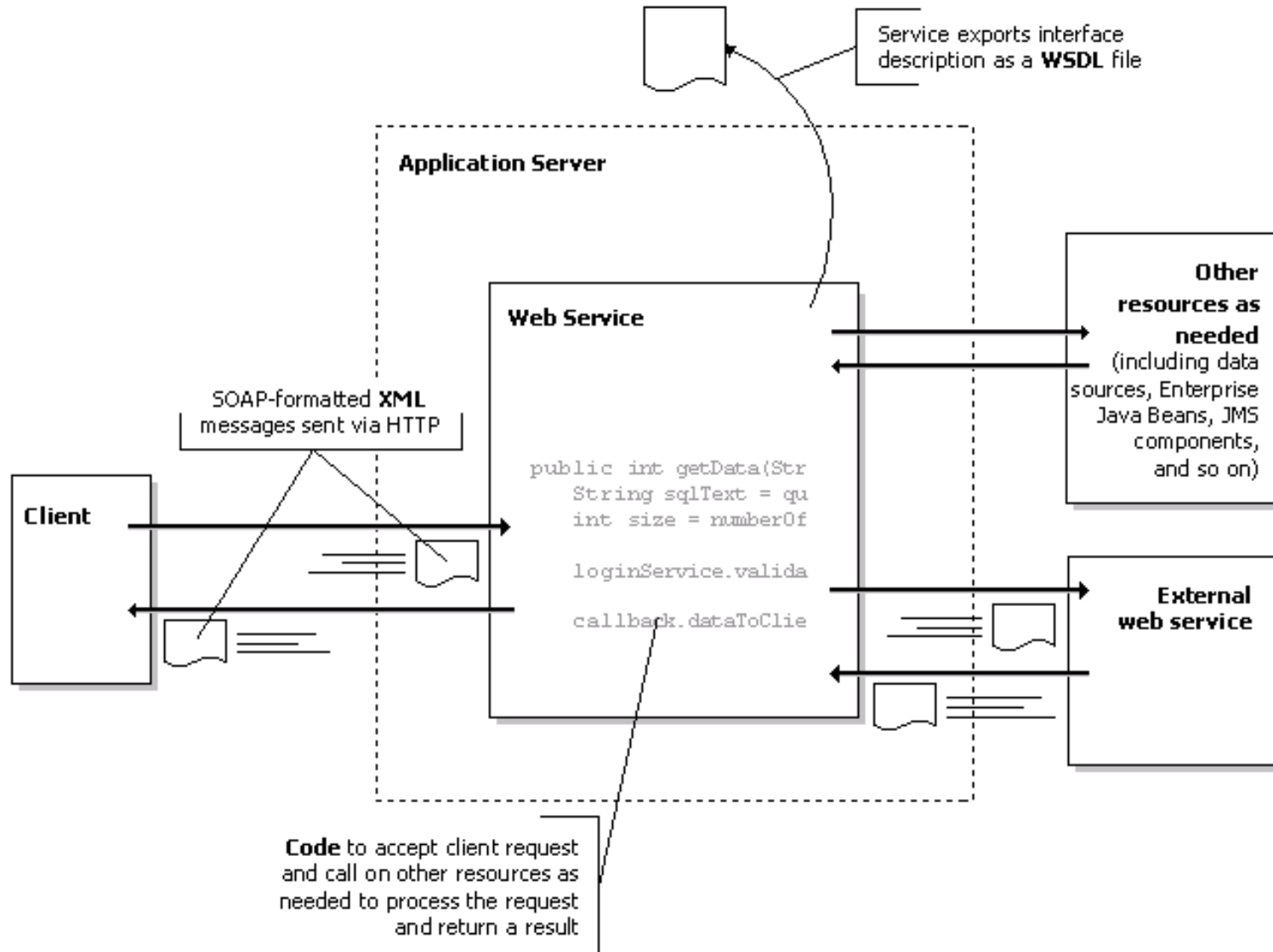
```
- <!--  
    Messages for Google Web APIs - cached page, search, spelling.  
-->  
- <message name="doGoogleSearch">  
    <part name="key" type="xsd:string"/>  
    <part name="q" type="xsd:string"/>  
    <part name="start" type="xsd:int"/>  
    <part name="maxResults" type="xsd:int"/>  
    <part name="filter" type="xsd:boolean"/>  
    <part name="restrict" type="xsd:string"/>  
    <part name="safeSearch" type="xsd:boolean"/>  
    <part name="lr" type="xsd:string"/>  
    <part name="ie" type="xsd:string"/>  
    <part name="oe" type="xsd:string"/>  
    </message>  
- <message name="doGoogleSearchResponse">  
    <part name="return" type="typens:GoogleSearchResult"/>  
    </message>  
    <!-- Port for Google Web APIs, "GoogleSearch" -->  
- <portType name="GoogleSearchPort">  
    - <operation name="doGoogleSearch">  
        <input message="typens:doGoogleSearch"/>  
        <output message="typens:doGoogleSearchResponse"/>  
    </operation>  
    </portType>
```

# GoogleSearch – Binding y Service

```
- <!--  
    Binding for Google Web APIs - RPC, SOAP over HTTP  
-->  
- <binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">  
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>  
    - <operation name="doGoogleSearch">  
        <soap:operation soapAction="urn:GoogleSearchAction"/>  
        - <input>  
            <soap:body use="encoded" namespace="urn:GoogleSearch"  
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>  
        </input>  
        - <output>  
            <soap:body use="encoded" namespace="urn:GoogleSearch"  
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>  
        </output>  
    </operation>  
    </binding>  
    <!-- Endpoint for Google Web APIs -->  
- <service name="GoogleSearchService">  
    - <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">  
        <soap:address location="http://api.google.com/search/beta2"/>  
    </port>  
    </service>  
</definitions>
```

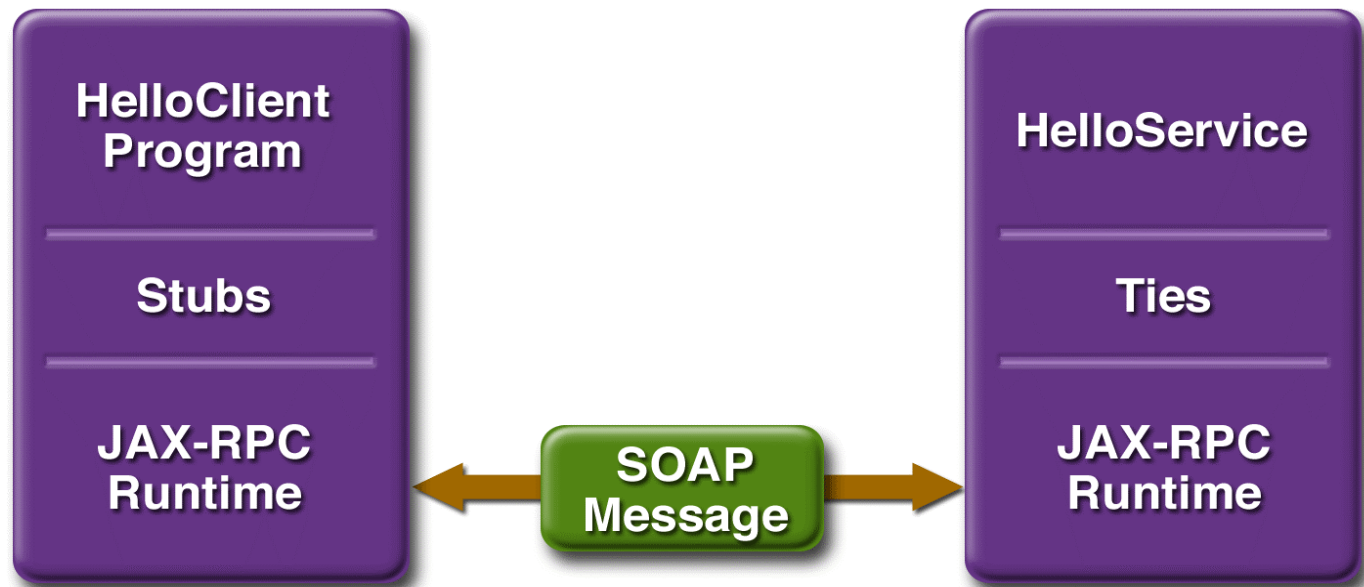
- Universal Description Discovery & Integration
- *"Web services are meaningful only if potential users may find information sufficient to permit their execution. The focus of Universal Description Discovery & Integration (UDDI) is the definition of a set of services supporting the description and discovery of (1) businesses, organizations, and other Web services providers, (2) the Web services they make available, and (3) the technical interfaces which may be used to access those services. Based on a common set of industry standards, including HTTP, XML, XML Schema, and SOAP, UDDI provides an interoperable, foundational infrastructure for a Web services-based software environment for both publicly available services and services only exposed internally within an organization."*

# Arquitectura Web Services



# Web Services y J2EE

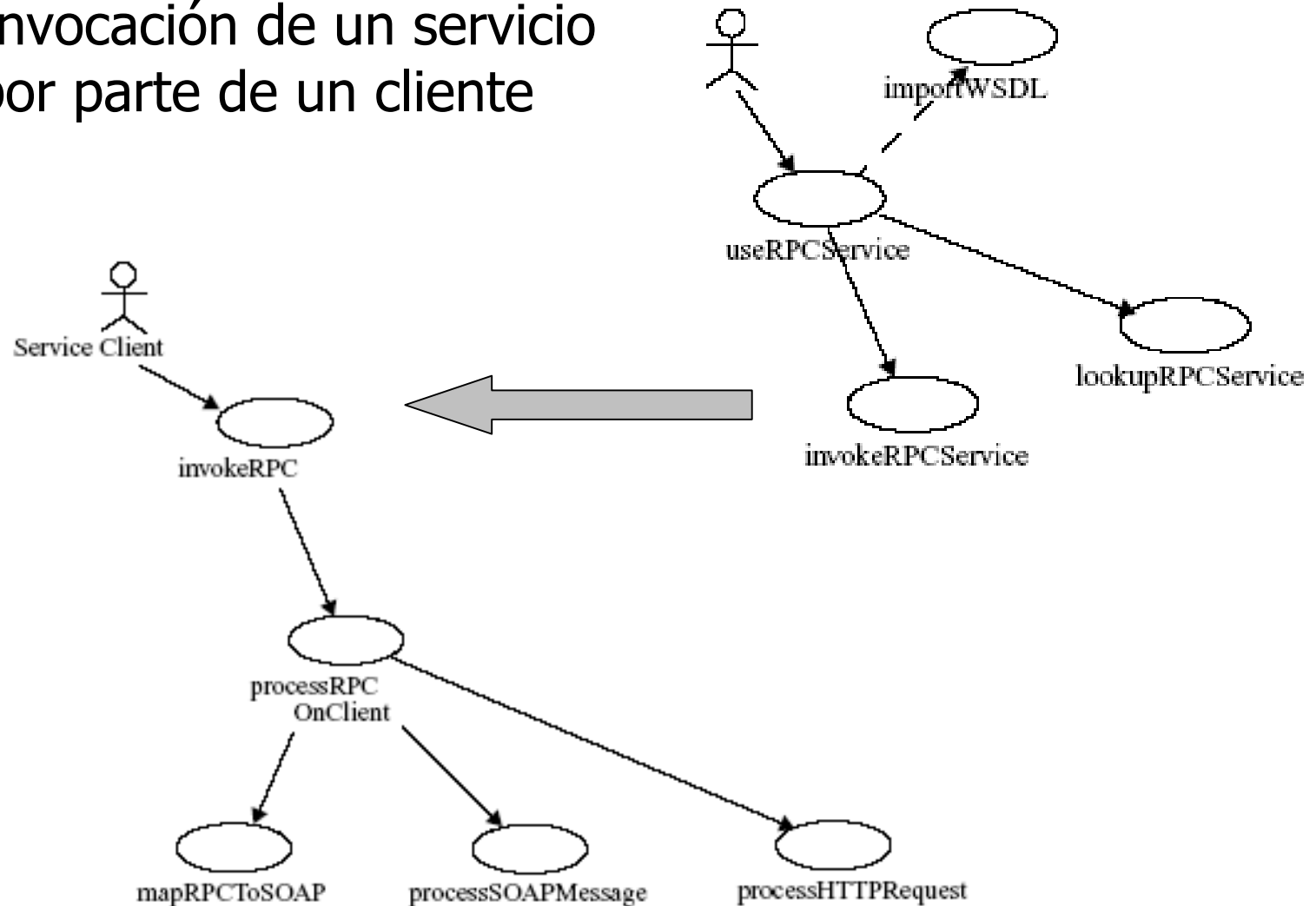
- JAX-RPC: Java API for XML-Based RPC
  - Permite construir Web Services y clientes de Web Services que utilizan SOAP sobre HTTP





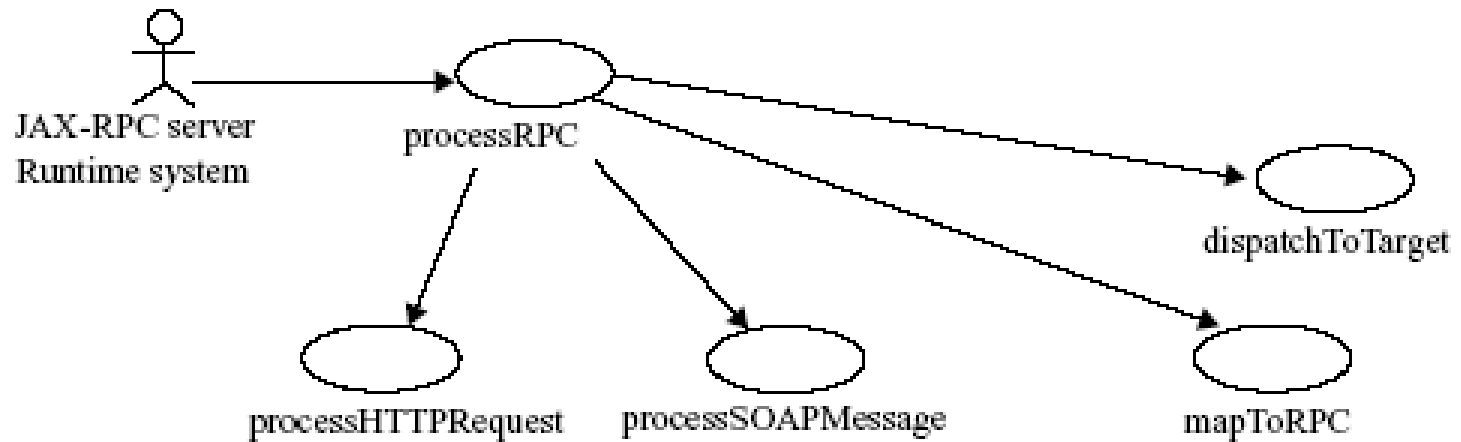
# JAX-RPC

- Invocación de un servicio por parte de un cliente



# JAX-RPC

- Invocación de un servicio en el servidor





# Tipos de Datos Soportados

---

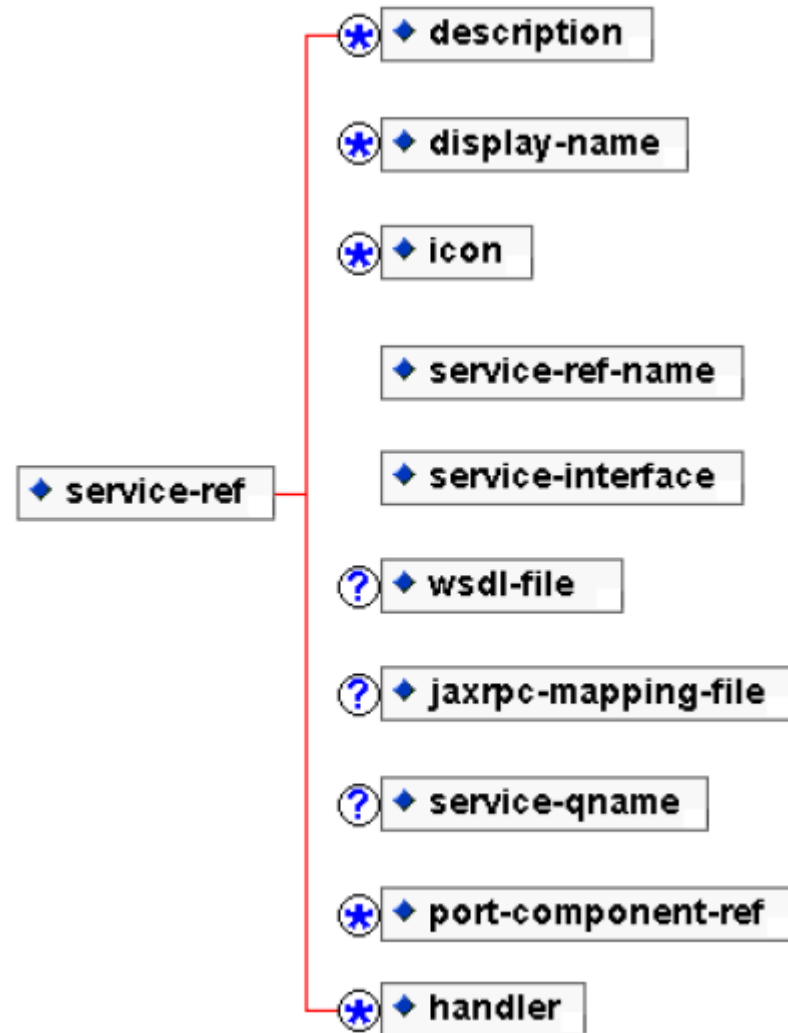
- JAX-RPC provee un mapping para los siguientes tipos de datos
  - Tipos primitivos:
    - boolean, byte, double, float, int, long, short
  - Clases estándar:
    - Boolean, Byte, Double, Float, Integer, Long, Short, String
    - java.math.BigDecimal, java.math.BigInteger
    - java.net.URI
    - java.util.Calendar, java.util.Date
  - Arreglos (unidimensionales y multidimensionales)
  - “Value Types” – clases que cumplan con los siguientes requerimientos:
    - Posee un constructor default
    - No implementar java.rmi.Remote
    - Sus campos deben ser soportados por JAX-RPC
    - Campos públicos no pueden ser final ni transient
    - Campos no públicos deben poseer getters y setters

# Creación de Web Services

- Para generar un Web Service en J2EE, debe definirse el Service Endpoint y la clase de implementación
- Service Endpoint
  - ```
public interface HolaIF extends java.rmi.Remote {  
    public String hola(String s) throws java.rmi.RemoteException;  
}
```
- Implementación (puede ser una clase Java o un EJB session stateless)
  - ```
public class HolaImpl implements HolaIF {  
    public String hola(String s) {  
        return "Hola, " + s + "!";  
    }  
}
```
- Los servidores J2EE y los IDE's proveen herramientas para generar stubs y skeletons, y para hacer el deployment

# Web Services en la Capa Web

- Los Web Services se definen en el deployment descriptor de la aplicación Web (archivo web.xml)



# Web Services en la Capa EJB

- Un stateless session bean puede proveer una interfaz Web Service (denominada Web Service Endpoint Interface), además de las interfaces Home y EJB
- La interfaz Web Service debe extender a `java.rmi.Remote`
- Los tipos de datos de parámetros y retornos de los métodos de la interfaz Web Service deben ser soportados por JAX-RPC
- Para cada método de la interfaz Web Service debe haber un método con la misma firma en la clase de implementación
- Deployment descriptor:

```
<session>
```

```
  <ejb-name>SearchBean</ejb-name>
```

```
  <ejb-class>com.wombat.global.SearchBean</ejb-class>
```

```
  ...
```

```
  <service-ref>
```

```
    <service-ref-name>service/SearchService</service-ref-name>
```

```
    <service-interface>com.example.SearchService</service-interface>
```

```
  </service-ref>
```

```
  ...
```

```
</session>
```



# Resumen

---

- La tecnología de Web Services permite la integración de aplicaciones distribuidas, facilitando la invocación de servicios sobre Internet, con independencia de la plataforma tecnológica del cliente y el servidor
- SOAP es un protocolo basado en XML que facilita la invocación de servicios en un ambiente distribuido, basado en el intercambio de mensajes
- WSDL es un formato XML para describir servicios
- UDDI facilita la búsqueda de servicios en un repositorio de tipo páginas amarillas
- J2EE simplifica la construcción de Web Services y clientes de Web Services mediante JAX-RPC
- Las herramientas de desarrollo facilitan el manejo de JAX-RPC, ofreciendo un modelo de programación que oculta las complejidades de la plataforma