



Introducción a los Servicios Web

Jose Emilio Labra Gayo

Octubre 2006



Contenidos

Introducción

Estándares

SOAP

WSDL

UDDI

Arquitecturas

Retos



Servicios Web

- ▶ Aplicaciones auto-contenidas, auto-descritas que pueden ser publicadas, localizadas e invocadas a través de la Web
- ▶ Una vez desarrolladas, otras aplicaciones (y otros servicios Web) pueden descubrirlas e invocar el servicio dado



Servicios Web

- ▶ Posible definición:
 - ▶ *Aplicaciones auto-contenidas, auto-descritas que pueden ser publicadas, localizadas e invocadas a través de la Web*
 - ▶ Una vez desarrolladas, otras aplicaciones (y otros servicios Web) pueden descubrirlas e invocar el servicio dado
- ▶ *Nota:* No todos los servicios Web están publicados para ser descubiertos automáticamente



Objetivos

- ▶ *Independencia* del lenguaje y de la plataforma
 - ▶ Separación de especificación de la implementación
- ▶ *Interoperabilidad*
 - ▶ Utilización de estándares: XML, SOAP, WSDL, UDDI...
- ▶ *Acoplamiento débil*: Sistemas basados en mensajes
 - ▶ Interacciones síncronas y asíncronas
- ▶ A través de *Internet*
 - ▶ Sin control centralizado
 - ▶ Utilización de Protocolos establecidos
 - ▶ Consideraciones de seguridad
- ▶ *Modularidad y Reusabilidad* de servicios
- ▶ *Escalabilidad*: Uno-a-uno frente a uno-a-muchos



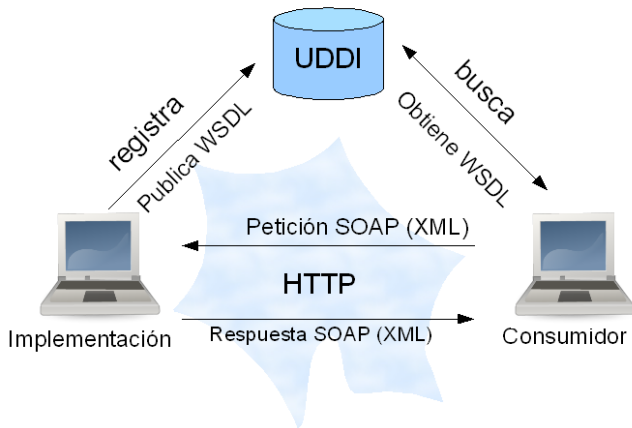
Principales acciones

- ▶ *Representación de mensajes*
 - ▶ Los mensajes suelen ser documentos XML
 - ▶ SOAP es un vocabulario XML que permite incluir mensajes XML
 - ▶ Además, en SOAP se representa otra información:
 - ▶ Cabecera (meta-información) y cuerpo del mensaje
 - ▶ Codificación de errores
- ▶ *Transporte de mensajes*: Habitualmente HTTP, aunque pueden utilizarse otros protocolos
- ▶ *Descripción del servicio*: Representación del tipo de operaciones y su funcionalidad (*Interfaz*)
 - ▶ WSDL es el vocabulario más utilizado
 - ▶ Define las operaciones y el tipo que tienen (no define funcionalidad)
- ▶ *Registro*: Es necesario disponer de un sistema que permita





Estándares de servicios Web



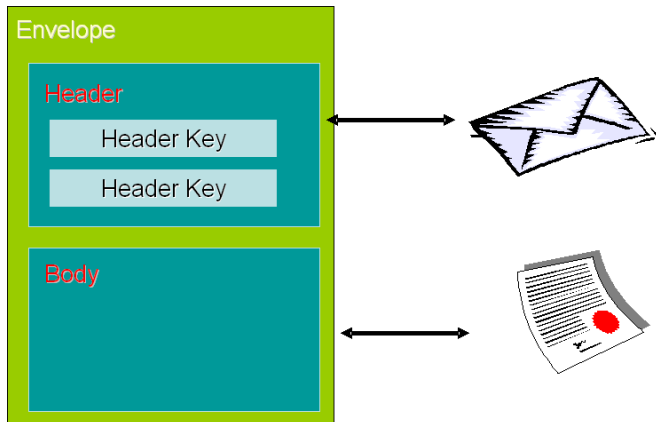


SOAP

- ▶ SOAP define el formato de los mensajes
- ▶ Evolución:
 - ▶ Desarrollado a partir de XML-RPC
 - ▶ SOAP 1.0 (1999), SOAP 1.1 (2000), SOAP 1.2 (2003)
 - ▶ Participación inicial de Microsoft
 - ▶ Adopción posterior de IBM, Sun, etc.
 - ▶ Aceptación industrial



SOAP





Ejemplo SOAP

Ejemplo

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  xmlns:p="http://www.mafia.it/pizzas">
  <soap:Header>
    <p:priority> urgente </p:priority>
    <p:origin>pepe@oviedo.es</p:origin>
  </soap:Header>
  <soap:Body>
    <p:order>
      <p:pizza nombre="Margarita">
        <p:size>familiar</p:size>
        <p:comment>con mucho queso
        </p:comment>
      </p:pizza>
    </p:order>
  </soap:Body>
</soap:Envelope>
```





Formato SOAP

- ▶ SOAP especifica el formato de mensajes
- ▶ Es independiente del protocolo de transporte
- ▶ Aunque se define un enlace (binding) con HTTP
- ▶ envelope contiene: header (opcional) y body (obligatorio)
 - ▶ body contiene datos en formato XML
 - ▶ header contiene meta-información
- ▶ También se pueden indicar errores mediante fault



WSDL

- ▶ WSDL (*Web Services Description Language*) permite describir servicios web
 - ▶ ¿Qué puede hacer el servicio?
 - ▶ ¿Dónde reside?
 - ▶ ¿Cómo invocarlo?
- ▶ Vocabulario basado en capas
 - ▶ Es posible concentrarse en una capa cada vez
- ▶ Evolución
 - ▶ Iniciativa conjunta de Ariba, IBM y Microsoft
 - ▶ (2001) Propuesto a W3C como recomendación (WSDL 1.1)
 - ▶ (2003) En desarrollo WSDL 2.0



Estructura de WSDL

- ▶ **definitions** incluye las siguientes entradas:
 - ▶ **types**: Tipos de datos usados en los mensajes (XML Schema)
 - ▶ **message**: Definición abstracta de los datos transmitidos.
 - ▶ **portType**: Conjunto de operaciones abstractas
 - ▶ **binding**: Protocolo concreto y especificaciones de las operaciones del mensaje
 - ▶ **port**: Especifica una dirección para el enlace definiendo un único punto de destino
 - ▶ **service**: Colección de puntos de destino



Ejemplo WSDL (1/3)

Ejemplo

```
<?xml version="1.0"?>
<definitions name="Pizzas"
  targetNamespace="http://mafia.it/pizzas.wsdl"
  xmlns:tns="http://mafia.it/pizzas.wsdl"
  xmlns:xsd1="http://mafia.it/pizzas.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://mafia.it/pizzas.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="PrecioPizzaRequest">
        <complexType>
          <all>
            <element name="nombrePizza" type="string"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>
</definitions>
```





Ejemplo WSDL (2/3)

Ejemplo

```
<message name="precioPizzaInput">
  <part name="body" element="xsd1:PrecioPizzaRequest" />
</message>

<message name="precioPizzaOutput">
  <part name="body" element="xsd1:precioPizza" />
</message>

<portType name="PizzasPortType">
  <operation name="verPrecio">
    <input message="tns:precioPizzaInput" />
    <output message="tns:precioPizzaOutput" />
  </operation>
</portType>
```





Ejemplo WSDL (3/3)

Ejemplo

```
<binding name=" PizzasSoapBinding"
  type="tns:PizzasPortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="precioPizza">
    <soap:operation soapAction="http://mafia.it/Pizzas"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

<service name=" PizzasService">
  <documentation>Ejemplo de servicio</documentation>
```





UDDI

- ▶ UDDI: (*Universal Discovery, Description and Integration*)
- ▶ Estándar para la publicación y registro de servicios Web
- ▶ Consorcio formado por IBM, Hp, Sun, Microsoft, Oracle, etc.
- ▶ Evolución:
 - ▶ UDDI 1.0 (2000) Fundación del registro
 - ▶ UDDI 2.0 (2001) Alineación con estándares y taxonomía de servicios más flexible
 - ▶ UDDI 3.0 (2002) Interacción de implementaciones públicas y privadas
- ▶ 2 partes
 - ▶ Descripción de negocios
 - ▶ Páginas blancas (información de contacto)
 - ▶ Páginas amarillas (información de la industria)
 - ▶ Páginas verdes (información técnica y especificaciones)
 - ▶ Registro de servicios



Funcionamiento de UDDI

- ▶ Funcionamiento como una base de datos distribuida P2P
- ▶ *Provider*: Información sobre la entidad que ofrece el servicio
- ▶ *Service*: Información sobre una familia particular de ofertas
- ▶ *Binding*: Información técnica sobre un punto de entrada a un servicio
- ▶ `tModel`: Descripción de especificaciones de servicios



Arquitecturas Orientadas a Servicios

- ▶ Importancia de las Interfaces
 - ▶ Descripción rigurosa de las interfaces
 - ▶ Preferiblemente: Tratamiento automático
 - ▶ Recomendación: Desarrollar el sistema a partir de las interfaces
- ▶ Modelos Débilmente acoplados
 - ▶ Sistemas de comunicación asíncrona
 - ▶ Estilo documento vs estilo RPC
 - ▶ Gestión de colas de mensajes
 - ▶ Ejemplo: Solicitud de libro
- ▶ Interoperabilidad
 - ▶ Independencia de Lenguajes y plataformas
 - ▶ Adaptación de arquitecturas ya existentes
 - ▶ Utilización de estándares: REST vs RPC



Estilos de arquitecturas

- ▶ Documentos vs RPC
 - ▶ Estilo *orientado a documentos*: Los mensajes conllevan toda la información necesaria.
 - ▶ Estilo *RPC (Remote procedure call)*: los mensajes incluyen los parámetros de la llamada
- ▶ SOAP vs REST
 - ▶ *SOAP* incluye un nuevo protocolo de mensajería
 - ▶ *REST* propone reutilizar protocolos ya existentes



Modelo REST

- ▶ REST (*REpresentational State Transfer*) fue un modelo propuesto por uno de los desarrolladores de HTTP
- ▶ Dos posibilidades:
 - ▶ Conjunto de principios de arquitectura
 - ▶ Utilización de XML básico sobre HTTP sin añadir ninguna otra capa de mensajería



Principios de diseño REST

- ▶ El estado y la funcionalidad de las aplicaciones se divide en recursos
- ▶ Todo recurso es identificado de forma única global
 - ▶ En HTTP los recursos se identifican mediante URIs
- ▶ Todos los recursos comparten un interfaz uniforme formado por
 - ▶ Conjunto de operaciones limitado para transferencia de estado
 - ▶ En HTTP: GET, PUT, POST, DELETE
 - ▶ Conjunto limitado de tipos de contenidos
 - ▶ En HTTP se identifican mediante tipos MIME
 - ▶ Un protocolo cliente/servidor, sin estado y basado en capas



Ventajas de REST

- ▶ Mejores tiempos de respuesta y disminución de carga en servidor
- ▶ Mayor escalabilidad al no requerir mantenimiento de estado
- ▶ Facilita desarrollo de clientes
- ▶ Mayor estabilidad frente a futuros cambios
 - ▶ Permite evolución independiente de los tipos de documentos
 - ▶ La creación de nuevos tipos de documentos no afecta a los anteriores



Retos para los servicios Web

- ▶ Gestión de servicios Web
 - ▶ WSDM - Web Services Distribution Management
 - ▶ Coordinación de servicios
 - ▶ Ejemplo. Reserva de avión + hotel
 - ▶ Evolución de los servicios
 - ▶ Ejemplo: cambio en la interfaz
 - ▶ Orquestación vs coreografía
- ▶ Modelización de procesos de negocios
 - ▶ BPEL - Business Process Execution Language
 - ▶ Contratos, facturación: ¿Quién gana dinero? ¿Qué pasa cuando algo falla?
- ▶ Seguridad y fiabilidad
 - ▶ XML Security
- ▶ Calidad de servicios
 - ▶ Tiempos de respuesta, soporte, monitorización, etc.