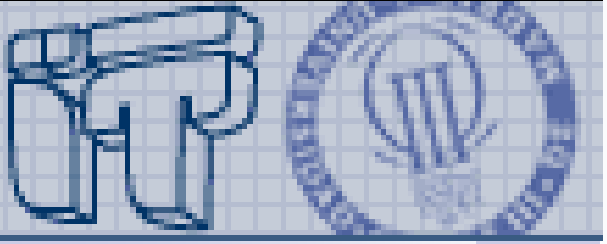


Sistemas de Información

Servicios Web II (Ejemplos)

Agradecimientos: Mario Muñoz Organero <munozm>, Simon Pickin de IT/UCIIIM <spickin>

Web Services *(Repaso)*

- **¿Qué son?**
  - un mecanismo de comunicación distribuida
  - que permite que las aplicaciones:
    - compartan información
    - invoquen funciones de otras aplicaciones
  - independientemente de
    - Cómo hayan sido creadas (lenguaje de programación)
    - Cómo se ejecutan (sistema operativo y plataforma)
    - Dispositivos utilizados para acceder a ellas
- **¿Para qué sirven?**
  - Incluido en la propia definición (puntos 2 y 3)
  - Crean una especie de WWW paralela de carácter cibernético
    - WWW humana (personas accediendo a pags web)
    - WWW cibernética (aplicaciones accediendo a servicios Web)

Web Services *(Repaso)*

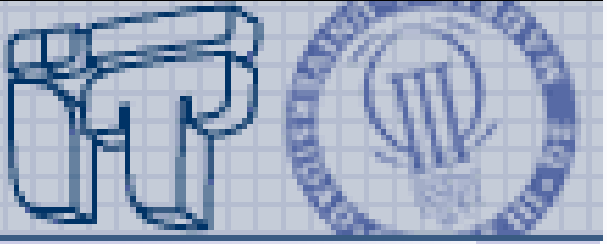
¿Cómo se usan?

- **Crear la lógica del negocio del Servicio Web.** Diseñar e implementar la aplicación que contenga la lógica de negocio del servicio Web (cualquier lenguaje, plataforma y sistema operativo)
- **Desplegar el servicio en un servidor**
  - Instalar y configurar el servidor
  - Ubicar la aplicación en el lugar adecuado
  - Convertir aplicación en un WS proporcionando descripción **WSDL**.
  - Registrar el servicio web en un directorio (opcional) usando **UDDI**
- **Desarrollar la aplicación cliente que accede al servicio Web**.
  - El cliente usa protocolo de mensajería XML para acceder al WS
  - 2 estilos (**rpc** vs **Document**)

Web Services *(Repaso)*

¿Tecnologías utilizadas?

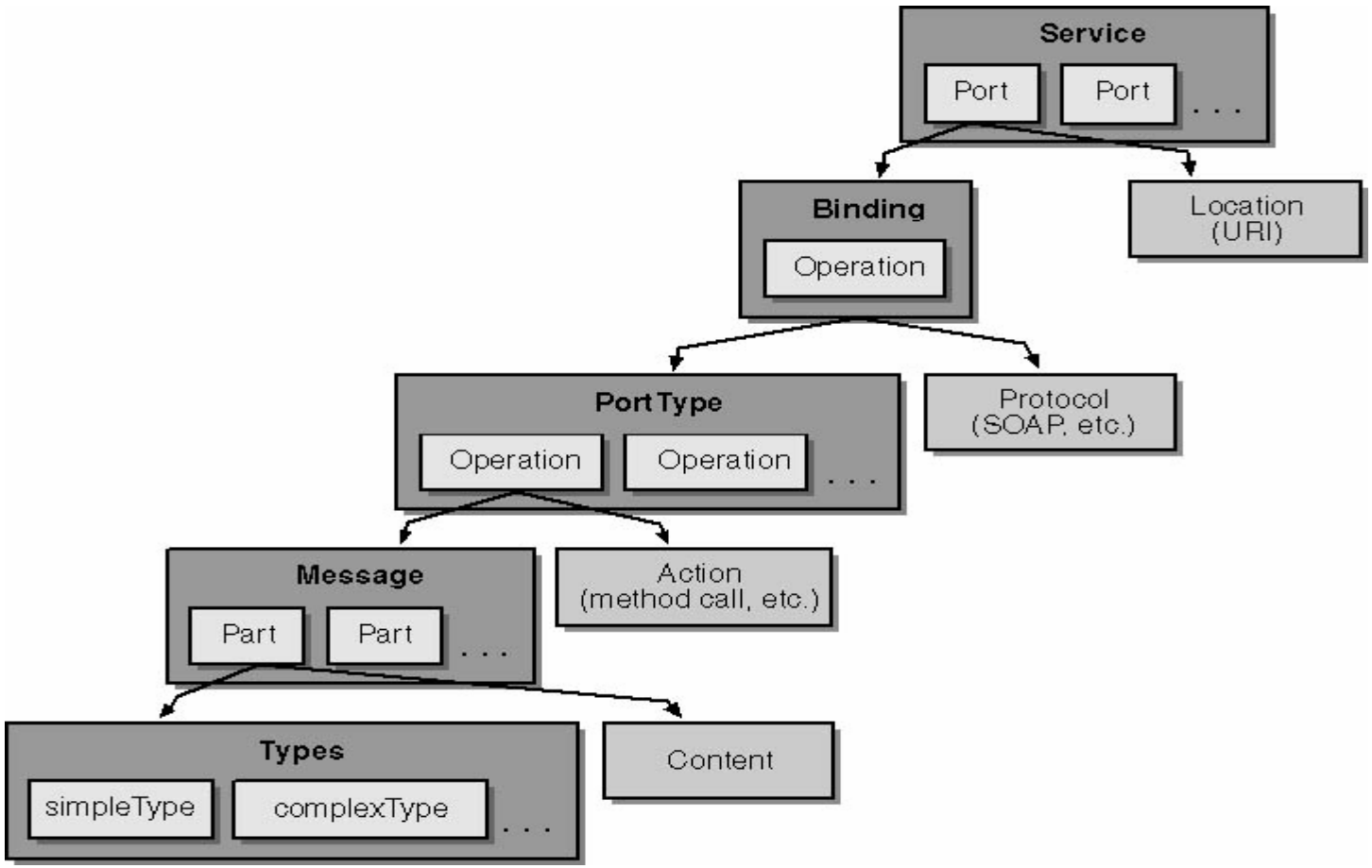
- **WSDL** *(**W**eb **S**ervice **D**escription **L**anguage)* :
  - Sirve para describir el servicio
  - Suele utilizarse
- **SOAP** *(**S**imple **O**bject **A**ccess **P**rotocol)* :
  - Sirve para intercambio de mensajes protocolo subyacente
  - Ampliamente utilizado
- **UDDI** *(**U**niversal **D**escription, **D**iscovery and **I**ntegration)*:
  - Sirve para descubrimiento de servicios.
  - Empieza a utilizarse



WSDL - Ejemplos

Web Service Description Language

Componentes de un servicio



Componentes de un servicio

<?xml version="1.0" encoding="UTF-8"?>

<definition> Es el elemento raíz de un documento WSDL

<types> Indica qué tipo de datos serán transmitidos **</types>**

<message> Indica qué mensaje será transmitido **</message >**

<porttype> Indica qué operaciones (funciones) se soportan **</porttype >**

<binding> Indica:

- Cómo se transmitirán los mensajes por la red
- Qué detalles hay sobre SOAP

</binding >

<service> Indica donde está localizado el servicio **</service>**

</definition>

Otros componentes

- documentation
  - Se usa para añadir comentarios en lenguaje humano a las definiciones.
- import
  - Importa otros documentos dentro del actual incorporando otro espacio de nombres.
- include
  - Igual que import pero con el mismo espacio de nombres.

import

```
<definitions .... >
    <import namespace="uri" location="uri"/> *
</definitions>
```

```
<?xml version="1.0"?>
<schema targetNamespace="http://example.com/stockquote/schemas"
        xmlns="http://www.w3.org/2000/10/XMLSchema">

    <element name="TradePriceRequest">
        <complexType>
            <all>
                <element name="tickerSymbol" type="string"/>
            </all>
        </complexType>
    </element>
```

```
<?xml version="1.0"?>
<definitions name="StockQuote"

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

   <import namespace="http://example.com/stockquote/schemas"
           location="http://example.com/stockquote/stockquote.xsd"/>

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

Espacios de nombres

Prefijo	***URI del espacio de nombres***	***definición***
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL namespace for WSDL framework.
soap	http://schemas.xmlsoap.org/wsdl/soap/	WSDL namespace for WSDL SOAP binding.
http	http://schemas.xmlsoap.org/wsdl/http/	WSDL namespace for WSDL HTTP GET & POST binding.
Mime	http://schemas.xmlsoap.org/wsdl/mime/	WSDL namespace for WSDL MIME binding.
soapenc	http://schemas.xmlsoap.org/soap/encoding/	Encoding namespace as defined by SOAP 1.1
Soapenv	http://schemas.xmlsoap.org/soap/envelope/	Envelope namespace as defined by SOAP 1.1
Xsi	http://www.w3.org/2000/10/XMLSchema-instance	Instance namespace as defined by XSD
Xsd	http://www.w3.org/2000/10/XMLSchema	Schema namespace as defined by XSD
Tns	(various)	El “this namespase” (tns) se usa como convención para referirse al documento corriente.

Ejemplo WSDL

http://www.oreilly.com/catalog/webservess/chapter/ch06.html

<?xml version="1.0" encoding="UTF-8"?>

<definition> El servicio HelloService

<message>
1) sayHelloRequest: El nombre del primer parámetro
2) sayHelloResponse: el saludo que corresponde al valor de retorno
</message >

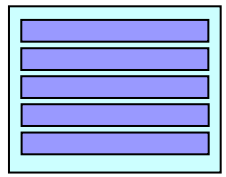
<porttype> operación sayHello que consiste en un servicio request/response **</porttype >**

<binding> Indica:
Dirección para usar el protocolo de transporte SOAP HTTP
</binding >

<service> Dirección donde está localizado el servicio http://localhost:8080/soap/servlet/rpcrouter **</service>**

</definition>

definitions



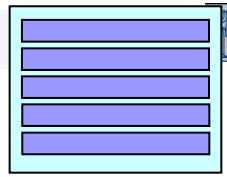
- Atributos:
  - Un nombre local para las definiciones
  - El nombre para el espacio de nombres "http://www.w3.org/2004/08/wsdl".
  - Los siguientes atributos:
    - targetNamespace – requerido
    - Otros espacios de nombres distintos a "http://www.w3.org/2004/08/wsdl".
- Elementos hijos:
  - **import**
  - **types**
  - **message**
  - **portType**
  - **binding**
  - **port**
  - **service**

```
<?xml version="1.0"?>
<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>
```

Ejemplo - WSDL



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService“
    targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

<type> Indica qué operaciones (funciones) se soportan </porttype >

```
<message name="SayHelloRequest">
        <part name="firstName" type="xsd:string"/>
</message>
<message name="SayHelloResponse">
        <part name="greeting" type="xsd:string"/>
</message>
```

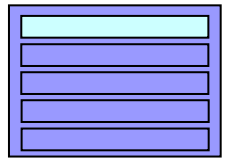
<porttype> Indica qué operaciones (funciones) se soportan </porttype >

<binding> Cómo se transmitirán los mensajes </binding >

<service> Indica donde está localizado el servicio </service>

```
</definitions>
```

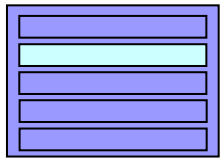
types



- Definiciones de tipos de datos que son relevantes para el intercambio de mensajes.
- Se usa XSD (interoperabilidad).

```
<definitions .... >
    <types>
        <xsd:schema .... />*
    </types>
</definitions>
```

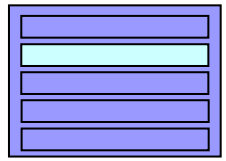
messages



- Cada mensaje tiene partes.
- Cada parte tiene nombre y tipo.

```
<definitions .... >
    <message name="nmtoken"> *
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </message>
</definitions>
```

message



```
<definitions .... >
    <types>
        <schema .... >
           <element name="PO" type="tns:POType"/>
           <complexType name="POType">
               <all>
                   <element name="id" type="string/>
                   <element name="name" type="string"/>
                   <element name="items">
                       <complexType>
                           <all>
                               <element name="item" type="tns:Item"
                           </all>
                       </complexType>
                   </element>
               </all>
           </complexType>
                   ...
    <message name="PO">
        <part name="po" element="tns:PO"/>
        <part name="invoice" element="tns:Invoice"/>
    </message>
</definitions>
```

Ejemplo - WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService“
    targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

<type> Indica qué operaciones (funciones) se soportan </porttype >

```
<message name="SayHelloRequest">
        <part name="firstName" type="xsd:string"/>
</message>
<message name="SayHelloResponse">
        <part name="greeting" type="xsd:string"/>
</message>
```

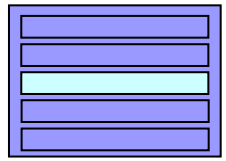
<porttype> Indica qué operaciones (funciones) se soportan </porttype >

<binding> Cómo se transmitirán los mensajes </binding >

<service> Indica donde está localizado el servicio </service>

```
</definitions>
```

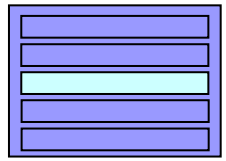
portType



- Conjunto de operaciones.
- Cada tipo de puerto tiene su nombre.

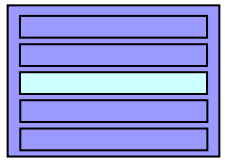
```
<wsdl:definitions .... >
    <wsdl:portType name="nmtoken">
        <wsdl:operation name="nmtoken" .... /> *
    </wsdl:portType>
</wsdl:definitions>
```

operations



- Cada una de las funciones que se pueden invocar en un puerto.
- Cuatro tipos:
  - **One-way**. Se recibe un mensaje.
  - **Request-response**. Se recibe un mensaje y se contesta.
  - **Solicit-response**. Se manda un mensaje y se espera la respuesta.
  - **Notification**. Se envía un mensaje para el que no se espera respuesta.

Ejemplo - WSDL



<?xml version="1.0" encoding="UTF-8"?>
<definitions … >

<type> Indica qué operaciones (funciones) se soportan </porttype >

<message> Indica qué mensaje será transmitido </message >

```
<portType name="Hello_PortType">
        <operation name="sayHello">
                <input message="tns:SayHelloRequest"/>
                <output message="tns:SayHelloResponse"/>
        </operation>
</portType>
```

<binding> Cómo se transmitirán los mensajes </binding >

<service> Indica donde está localizado el servicio </service>

</definitions>

bindings

- Se les pone un nombre.
- El tipo hace referencia al portType.
- Los campos de extensibilidad recogen elementos que especifican la gramática concreta para los mensajes de entrada, salida y error.
- Se puede especificar opcionalmente información de binding para toda una operación así como para todo el elemento de binding.

Otros binding

- A parte de SOAP tenemos:
  - MIME
  - HTTP GET/POST

```
<definitions .... >
    <binding .... >
        <http:binding verb="nmtoken"/>
        <operation .... >
            <http:operation location="uri"/>
            <input .... >
                <-- mime elements -->
            </input>
            <output .... >
                <-- mime elements -->
            </output>
        </operation>
    </binding>

    <port .... >
        <http:address location="uri"/>
    </port>
</definitions>
```

Ejemplo - WSDL

<?xml version="1.0" encoding="UTF-8"?>

<definitions … >

<type> Indica qué operaciones (funciones) se soportan </porttype >

<message> Indica qué mensaje será transmitido </message >

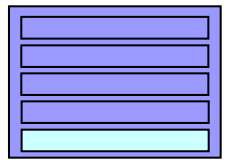
<porttype> Indica qué operaciones (funciones) se soportan </porttype >

```
<binding name="Hello_Binding" type="tns:Hello_PortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
        <soap:operation soapAction="sayHello"/>
        <input>
            <soap:body  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:examples:helloservice" use="encoded"/>
        </input>
        <output>
            <soap:body  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:examples:helloservice" use="encoded"/>
        </output>
    </operation>
</binding>
```

<service> Indica donde está localizado el servicio </service>

</definitions>

Services



- Tienen nombre
- Agrupan un serie de puertos

```
<wsdl:definitions .... >
    <wsdl:service name="nmtoken"> *
        <wsdl:port .... />*
    </wsdl:service>
</wsdl:definitions>
```

ports

- Se definen dentro de los servicios
- Tienen nombres y se asocian a un binding
- Indican la dirección dónde se invocan

```
<wsdl:definitions .... >
    <wsdl:service .... > *
        <wsdl:port name="nmtoken" binding="qname"> *
           <-- extensibility element (1) -->
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Ejemplo - WSDL

<?xml version="1.0" encoding="UTF-8"?>
<definitions … >

<type> Indica qué operaciones (funciones) se soportan </porttype >

<message> Indica qué mensaje será transmitido </message >

<porttype> Indica qué operaciones (funciones) se soportan </porttype >

<binding> Cómo se transmitirán los mensajes </binding >

```
<service name="Hello_Service">
        <documentation>
                WSDL File for HelloService
        </documentation>
        <port binding="tns:Hello_Binding" name="Hello_Port">
            <soap:address
        location="http://localhost:8080/soap/servlet/rpcrouter"/>
        </port>
</service>
```

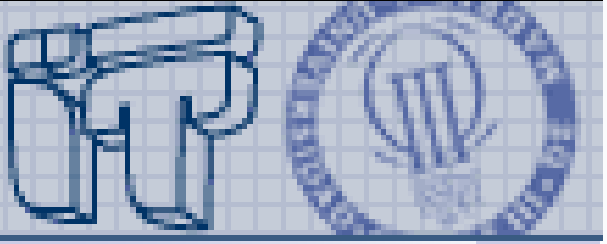
</definitions>

Mensaje de invocación

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:soap='http://schemas.xmlsoap.org/soap/
   envelope/' xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
   soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
   <soap:Body>
      <n:sayHello xmlns:n='urn:examples:helloservice'>
         <firstName xsi:type='xsd:string'>World</firstName>
      </n:sayHello>
   </soap:Body>
</soap:Envelope>
```

Mensaje de respuesta

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
   xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
   xmlns:xsd='http://www.w3.org/1999/XMLSchema'>
   <SOAP-ENV:Body>
      <ns1:sayHelloResponse
         xmlns:ns1='urn:examples:helloservice'
         SOAP-ENV:encodingStyle=
            'http://schemas.xmlsoap.org/soap/encoding/'>
         <return xsi:type='xsd:string'>Hello, World!</return>
      </ns1:sayHelloResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



SOAP: Ejemplos

Simple Object Access Protocol

¿Qué es SOAP?

- SOAP es una aplicación de la especificación XML.
- SOAP = XML messaging.
- El transporte de SOAP puede ser HTTP, FTP, TCP, SMTP, POP3, MQSeries,etc

SOAP conlleva.

- Tipos de información que vamos a intercambiar.
- Como será expresada la información en XML.
- Como se transmitirá esa información.

Todo eso lo provee SOAP

Estructura de un mensaje SOAP

- El SOAP Header es opcional.
- El SOAP Body contiene el mensaje en cuestión a ser procesado.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <!--Optional header information goes here. -->
    <To>Scott</To>
    <From>Suzanne</From>
  </soap:Header>

  <soap:Body>
    <!--Message goes here. -->
    Please pick up some milk on your way home from work.
  </soap:Body>
</soap:Envelope>
```

Usos de SOAP

- RPC: envío de parámetros y recepción de resultados.
- EDI: envío de y recepción de facturas, ejemplo: info financiera

Ejemplo:
http://cscgrad.cc.edu/jtowell/AAATeaching/csc560/WebServices/web_services_SOAP.htm

RPC – ejemplo petición

- En general viene de a pares (solicitud y respuesta), pero no siempre tiene que haber una respuesta.

```
<s: Envelope xmlns:s=http://www.w3.org/2001/06/soap-envelope>
   <s:Header>
      <m:transaction xmlns:m="soap-transaction"
                                          s:mustUnderstand="true"/>
   </s:Header>
   <s:Body>
      <n:getQuote xmlns:n=urn:QuoteService">
                  <symbol xsi:type="xsd:string">
                              IBM
                   </symbol>
      </n:getQuote>
   </s:Body>
</s:Envelope>
```

RPC – ejemplo respuesta

```
<s:Envelope xmlns:s=http://www.w3.org/2001/06/soap-envelope>
    <s:Body>
        <n:getQuoteResponse xmlns:n="urn:QuoteService">
            <value xsi:type="xsd:float">
                98.06
            </value>
        </n:getQuoteResponse>
    </s:Body>
</s:Envelope>
```

Estilos de codificación

- Es un conjunto de reglas que definen exactamente como los data types de aplicaciones y de la plataforma serán codificados en una sintaxis XML común .
- SOAP define más un método de codificación para convertir los datos de un objeto software a formato XML y viceversa.
- Para especificar el tipo de codificación se utiliza el atributo encodingStyle en el elemento getQuote del ejemplo:

```
<s:Envelope xmlns:s=http://www.w3.org/2001/06/soap-envelope>
    <s:Body>
      <n:getQuote xmlns:n:"urn:QuoteService"
          s:encodingStyle=http://www.w3.org/2001/06/soap-encoding>
                <symbol xsi:type="xsd:string">IBM</symbol>
      </n:getQuote>
    </s:Body>
</s:Envelope>
```

Estilos de codificación

- 3 de los “encoding styles” de SOAP se han popularizado:
  - *SOAP Remote Procedure Call (RPC) encoding,*
  - *SOAP Remote Procedure Call Literal encoding (SOAP RPC-literal),*
  - *SOAP document-style encoding*

Invocaciones estilo RPC

- La invocación es representada en una sola estructura con sus parámetros in o in-out.
- Los nombres y el orden físico de los parámetros debe coincidir con el de los parámetros del método que invocó.

Ejemplo - invocación

Declaración: String checkStatus(String orderCode, String customerID);

Invocación: result = checkStatus(“abc123”, “Bob’s store”);

```
<soap:Envelope xmlns:soap=“…”>
   <soap:Body>
      <checkStatus xmlns=“….”
         soap:encodingStyle=http://www.w3.org/2001/06/soap-encoding>
         <orderCode xsi:type=“string”>abc123</orderCode>
         <customerID xsi:type=“string”>Bob’s store</customerID>
      </checkStatus>
   </soap:Body>
</soap:Envelope>
```

Ejemplo - Respuesta

```
<soap:Envelope xmlns:soap="…">
   <soap:Body>
      <checkStatusResponse soap:encodingStyle="http:// ….">
         <return xsi:type="xsd:string">new</return>
      </checkStatusResponse>
   </soap:Body>
</SOAP:Envelope>
```

El “sobre”

- *SOAP Envelopes*: están hechos para llevar documentos XML arbitrarios, sin importar si es un objeto como una factura o una invocación RPC que codifica sus parámetros con “encoding rules” o no. El uso de “encoding styles” es totalmente opcional.
- Codificación de datos que no sean string de texto como estructurados, arrays y otros tipos compuestos se hace en base64

Tipos de datos

- SOAP define 3 formas distintas de expresar los tipos de datos de un tag:
  - Utilizar el atributo xsi:type en cada tag, explicitamente referenciando el tipo de datos de acuerdo con la especificación del XML Schema.

```
<person>
    <name xsi:type="xsd:string">John Doe</name>
</person>
```

  - Referenciar un XML Schema que define particularmente ese tipo de datos exacto.

```
<person xmlns="personschema.xsd">
    <name>John Doe</name>
</person>
<!-- en personschema.xsd se define el elemento como type=xsd:string -->
```

  - Referenciar otro tipo de documento schema que defina el tipo de datos de un tipo de elemento dentro del cual se declara.

```
<person xmlns="urn:some_namespace">
    <name>John Doe</name>
</person>
<!-- urn:namespace indica en el cual los valores de los elementos son strings -->
```

SOAP Faults

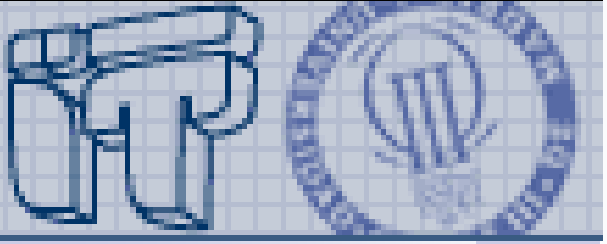
- Siempre en el Body
- Fault codes extensibles
  - No correspondencias de versiones
  - Problemas reconociendo headers o data encodings
  - Problemas genéricos en emisor o receptor
- Causas de fallo en texto legible

SOAP Faults

- Emplazarlos en el cuerpo del mensaje SOAP (env:Body elements)
- Dentro del Body en elemento env:Fault
- Subelementos:
  - env:Node identifica al nodo que genera el fallo
    - La ausencia implica “ultimo receptor”
  - env:Code
    - env:Value
    - env:Subcode
  - env:Reason
    - env:Text
  - env:Detail
    - Específico de la aplicación

Ejemplo - SOAP Faults

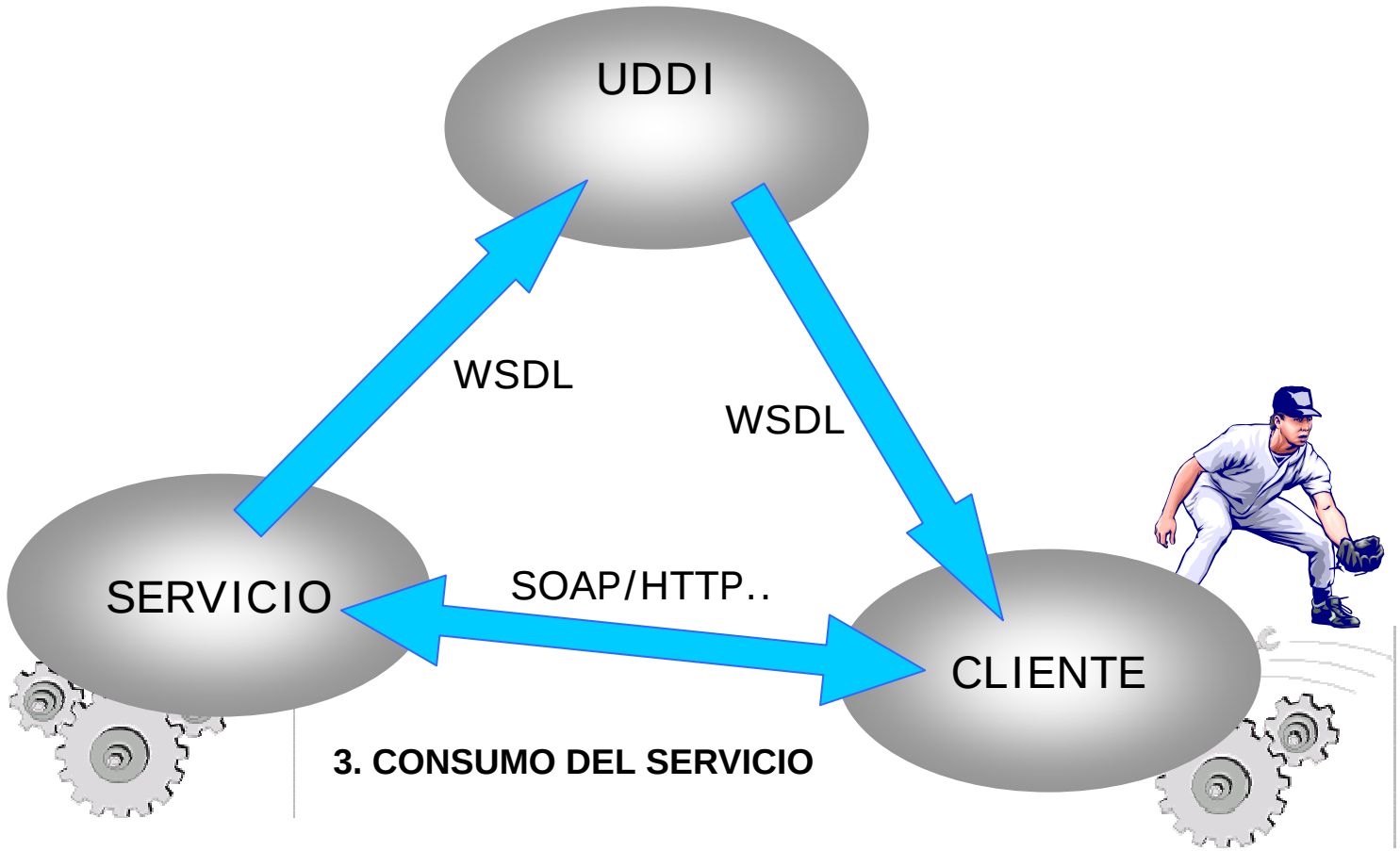
```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                    xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <env:Body xmlns:m="http://www.example.org/timeouts">
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>m:MessageTimeout</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">Sender Timeout</env:Text>
      </env:Reason>
      <env:Detail><m:MaxTime>P5M</m:MaxTime></env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```



UDDI: Ejemplos

Universal Description, Discovery and Integration

Arquitectura



UDDI

- Implementa la funcionalidad de “discovery” necesaria para poder encontrar la descripción WSDL del WebService que se necesita.
- UDDI tiene 2 partes:
  - Un directorio con los metadatos de todos los WebServices, incluyendo un puntero a la descripción WSDL de cada uno.
  - Las definiciones de “port types” WSDL para manipular y buscar en ese directorio.

UDDI

- Define estándares para un registro distribuido de servicios Web:
  - White pages (información general)
  - Yellow pages (categorías de servicios)
  - Green pages (reglas de negocio)

Ejemplo UDDI

<?xml version="1.0" encoding="UTF-8"?>

<businessEntity> Es el elemento raíz

<businessService>

…

< bindingTemplates >

…

< TModelInstanceDetails >

…

</ TModelInstanceDetails >

…

</ bindingTemplates >

…

</ businessService >

</ businessEntity >

Business Entity

- Provee información de quien desarrolló el WebService: la compañía, información de contacto en la misma, categorías de la industria, identificador de negocios y la lista de los servicios provistos
- A continuación presentamos un ejemplo ...

```
<businessEntity businessKey="uuid:11111111-2222-3333-4444-555555555555"
      operator="http://www.ibm.com"
      authorizedName="John Doe">
 <name>Acme Company</name>
 <description>
  We create cool WebServices
 </description>
 <contacts>
  <contact useType="general info">
   <description>General Information</description>
   <personName>John Doe</personName>
   <phone>(123) 123-1234</phone>
   <email>jdoe@acme.com</email>
  </contact>
 </contacts>
 <businessServices>
  ...
 </businessServices>
 <identifierBag>
  <keyedReference ModelKey="UUID:11111111-2222-3333-4444-555555555556"
            name="D-U-N-S"
            value="123456789" />
 </identifierBag>
 <categoryBag>
  <keyedReference ModelKey="UUID:11111111-2222-3333-4444-555555555557"
            name="NAICS"
            value="111336" />
 </categoryBag>
<businessEntity>
```

Business Services

- Representa un único WebService provisto por la “Business Entity”.
- La descripción incluye: tipo de WebService y a que categorías pertenece.
- La forma de identificar todas las “business entities” y los “business services” en UDDI es a través del UUID (“universally unique identifiers”).

Ejemplo de Business Service

```
<businessService serviceKey="uuid:11111111-2222-4444-5555-666666666666"
                 businessKey="uuid:11111111-2222-4444-5555-666666666667">
    <name>Hello World WebServices</name>
    <description>A friendly WebService</description>
    <bindingTemplates>
      ....
    </bindingTemplates>
    <categoryBag />
</businessService>
```

Binding Templates

- Son la descripción técnica de los WebServices representados por la estructura “business service”.
- Representan la implementación del WebService.
- Basicamente equivalen a el elemento “service” descripto en WSDL.
- Como un mismo servicio puede estar implementado de diferentes formas y puede ser asociado a múltiples protocolos o diferentes direcciones, puede haber varios binding templates para un mismo WebService
- Veamos un ejemplo a continuación …

```
<bindingTemplate serviceKey="uuid:11111111-2222-4444-5555-666666666666"
                 businessKey="uuid:11111111-2222-4444-5555-666666666667">
  <description>Hello World SOAP Binding</description>
  <accessPoint URLType="http">
    http://localhost:8080
  <accessPoint>
  <TModelInstanceDetails>
    <TModelInstanceInfo TModelKey="uuid:11111111-2222-4444-5555-666666666668">
      <instanceDetails>
        <overviewDoc>
          <description>
            references the description of the WSDL service definition
          </description>
          <overviewURL>
            http://localhost/helloworld.wsdl
          </overviewURL>
        </overviewDoc>
      </instanceDetails>
    </TModelInstanceInfo>
  </TModelInstanceDetails>
</bindingTemplate>
```

TModels

- Es una forma de describir varias estructuras “business + service + template” dentro del directorio UDDI.
- Sirven para representar los nuevos “port types” dentro de WSDL. Luego se puede especificar que un determinado business service implementa ese “port type”, asociando el TModel con uno de los “binding templates” del “business service”.

Ejemplo de TModel

```
<TModel TModelKey="uuid:xyz987…"
        operator="http://www.ibm.com"
        authorizeName="John Doe">
    <name>HelloWorldInterface Port Type</name>
    <description>
      An interface for a friendly WebService
    </description>
    <overviewDoc>
      <overviewURL>
        http://localhost/helloworld.wsdl
      </overviewURL>
    </overviewDoc>
</TModel>
```

Interfaces UDDI

- Ofrece 2 interfaces:
  - PublishSOAP: Para los proveedores de servicios
  - InquireSOAP: Para los consumidores de servicios
- Estos mismos servicios son descriptos en WSDL también.
- El tipo de datos manejados por las interfaces UDDI (por ej. businessDetails) está en las "UDDI XML Schema definitions".

- Para importar las definiciones WSDL:

```
<import namespace="urn:uddi-org:api
      location="http://www.uddi.org/schema/2001/uddi_v1.xsd" />
```

Interfaz Publisher

- get_authToken
- discard_authToken
- save_business
- save_service
- save_binding
- save_Tmodel
- delete_business
- delete_service
- delete_binding
- delete_Tmodel
- get_registeredInfo

Interfaz Inquiry

- find_binding
- find_business
- find_ltservice (por ej. lista WebServices que correponden a un criterio dado)
- find_TModel
- get_bindingDetail
- get_businessDetail
- get_businessDetailExt
- get_serviceDetail
- get_TModelDetail

Bibliografía

- Libros
  - L/D 004.738.52 WEB, Web services: concepts, architectures and applications. Alonso, Gustavo
  - L/D 004.738.52 CER, Web services essentials. Cerami, Ethan
  - L/S 004.738.5.057.4 SNE, Programming Web services with SOAP. Snell, James
  - L/D 004.438 JAVA BUI, Building Web services with Java : making sense of XML, Soap, WSDL and UDDI. Graham, Steve
- Web
  - Introducción a los servicios Web en java:
    http://www.programacion.com/java/tutorial/servic_web/
  - Web Services: XML-RPC, SOAP, sobre PHP, Perl, y otros conceptos:
    http://www.programacion.com/tutorial/xmlrpcsoap/