Ingeniería de Aplicaciones para la Web Semántica

Clase 03

WSDL y más XML

Mg. A. G. Stankevicius

Segundo Cuatrimestre

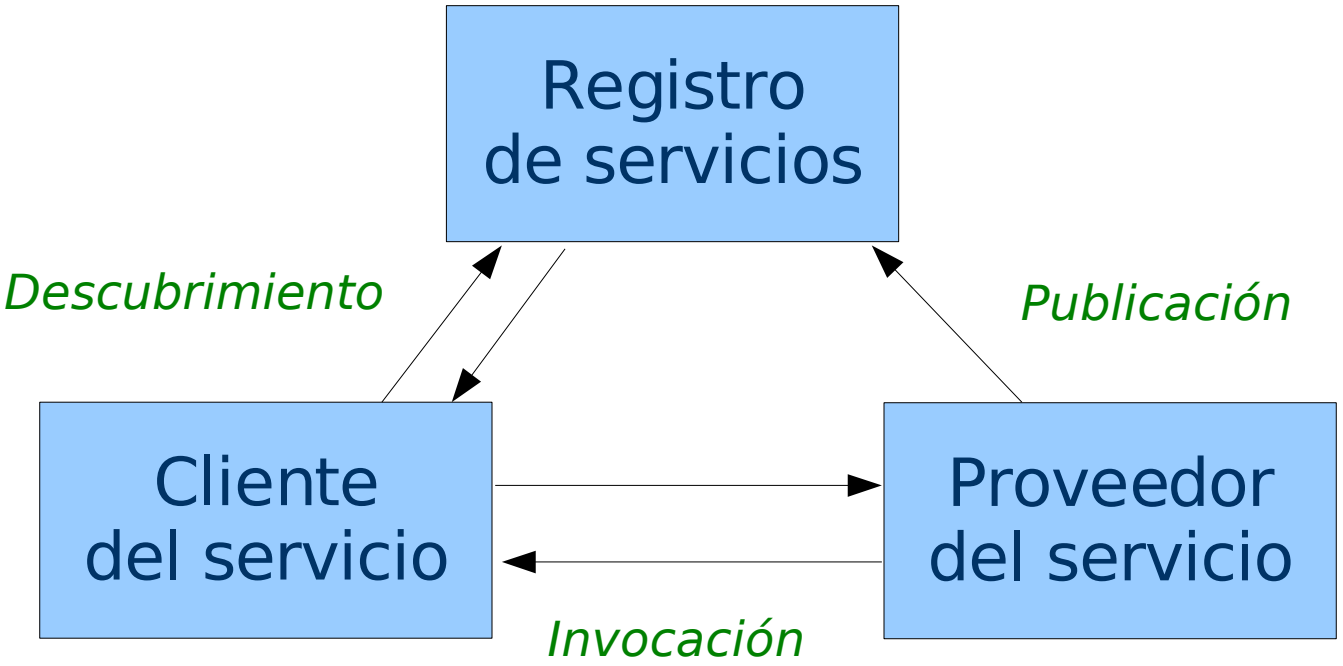
2005

Copyright

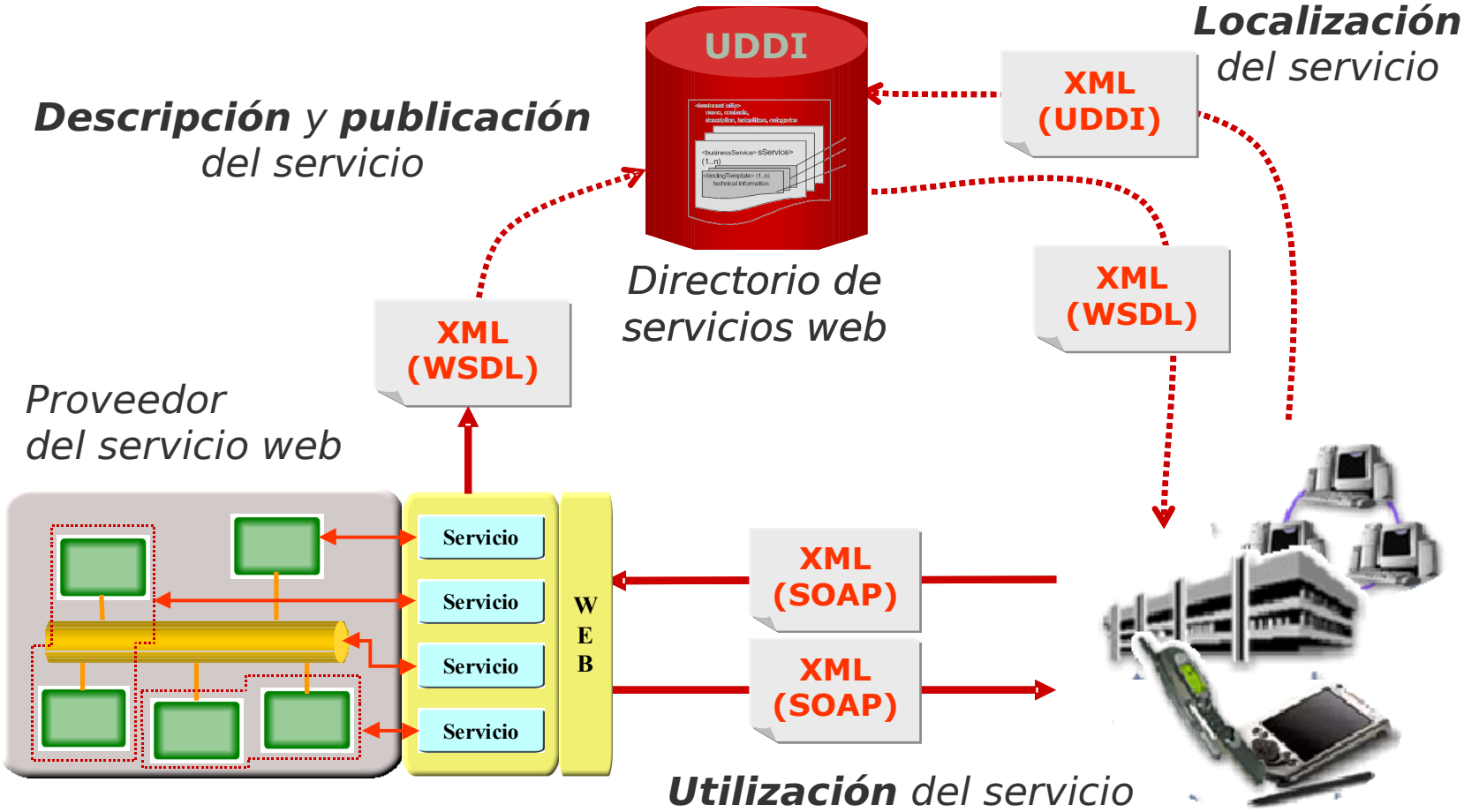
Contenidos

- El rol de WSDL dentro de los SW.
- Estructura de un documento WSDL.
- Introducción a los esquemas XML.
- Especificación de tipos simples.
- Especificación de tipos compuestos.
- Restricciones de forma y de contenido.

Anatomía de un servicio web



El rol de WSDL dentro de los servicios web



¿Qué es WSDL?

- WSDL es un lenguaje basado en XML que permite:
  - describir servicios web, y además
  - especificar cómo accederlos.
- Todo documento WSDL determina la ubicación de un cierto servicio y de los métodos que ese servicio hace visible.
- Todavía no es una recomendación aceptada por la W3C.
  - La versión 2.0 (a punto de salir) lo será.

¿Qué describe WSDL?

- WSDL describe dos aspectos de los servicios web:
  - La información que debe estar presente en la invocación a un determinado método.
  - La información que es retornada de toda invocación válida a un cierto método.
- En cierto sentido se está especificando las condiciones bajo las cuales el cliente del servicio contratará las prestaciones.

¿Qué más describe WSDL?

- WSDL también hace referencia a la ubicación física de cada uno de los métodos exportados por un cierto servicio.
- Para cada método se hace mención explícita de qué protocolo de comunicación se debe adoptar para poder comunicarse con el mismo.

Estructura de un documento WSDL

- WSDL caracteriza al servicio web que se está definiendo mediante los siguientes elementos XML:
  - `<portType>`: Define las operaciones soportadas por el servicio web.
  - `<message>`: Describe la estructura de los mensajes que utiliza cada operación.
  - `<type>`: Indica los tipos de datos empleados por las operaciones del servicio web.
  - `<binding>`: Especifica qué protocolo de comunicación es adoptado.

Estructura de un documento WSDL

```
<definitions>
    <types>
        ...definición de los tipos de datos...
    </types>
    <message>
        ...de los mensajes entendidos por las operaciones...
    </message>
    <portType>
        ...de las operaciones publicadas...
    </portType>
    <binding>
        ...de las vinculaciones a protocolos...
    </binding>
</definitions>
```

Fragmento de un documento WSDL

```
<message name="GetStockPriceRequest">
    <part name="stock" type="xs:string"/>
</message>
<message name="GetStockPriceResponse">
    <part name="value" type="xs:string"/>
</message>

<portType name="StocksRates">
    <operation name="GetStockPrice">
        <input message="GetStockPriceRequest"/>
        <output message="GetStockPriceResponse"/>
    </operation>
</portType>
```

Modelos de comunicación de las operaciones

- Cada puerto puede brindar múltiples operaciones.

```
<portType name="...">
   <operation name="...">
   </operation>
   ...
</portType>
```

- A su vez, cada operación puede adoptar un modelo de comunicación diferente:
  - RCP (existen `<input>` y `<output>`)
  - De sentido único (sólo hay `<output>`).

Definición de los argumentos de las operaciones

- Los argumentos de las operaciones definidas dentro de un mismo servicio se describen mediante sendos mensajes:

```
<message name="opRequest"> ... </message>
<message name="opResponse"> ... </message>

<portType name="WSName"
   <operation name="op">
      <input message="opRequest"/>
      <input message="opResponse"/>
   </operation>
</portType>
```

Definición de los tipos de datos usados

- Todo tipo de dato mencionado en la definición de los argumentos de las operaciones deben estar a si vez debidamente especificados:

```
<message name="consultaAlumno">
   <part name="registro" type="xs:integer"/>
</message>

<message name="consultaAlumno">
   <part name="registro" type="LU"/>
</message>
```

Tipos de datos predefinidos por los esquemas XML

- En la definición de los argumentos es posible hacer uso de los tipos de datos ya contemplados por los esquemas XML:
  - `xs:string`
  - `xs:integer`
  - `xs:decimal`
  - `xs:boolean`
  - `xs:date`
  - `xs:time`

Tipos de datos no predefinidos

- También es posible definir tipos de datos propios apelando a los esquemas XML:

```
<xs:schema ... > ... </xs:schema>
```

- Se asocia un tipo de dato a cada elemento defindo en el esquema XML:

```
<xs:element name="LU" type="xs:integer">
```

- Estos elementos pueden contener a su vez otros atributos:

```
<xs:attribute name="alive" type="xs:boolean">
```

Restricciones en el uso

- Los atributos pueden ser declarados como obligatorios u opcionales:

```
<xs:attribute name="alive"
type="xs:boolean" use="required">
<xs:attribute name="unemployed"
type="xs:boolean" use="optional">
```

- Los atributos pueden tener asociado un valor por defecto:

```
<xs:attribute name="alive"
type="xs:boolean" default="true">
```

Restricciones acerca del contenido

- También es posible especificar ciertas restricciones sobre los posibles valores que un elemento puede tomar:

```
<xs:element name="age">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:minInclusive value="0"/>
         <xs:maxInclusive value="100"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Restricción a sólo un conjunto de valores

- Otra forma de restrigir los valores que se permiten es la siguiente:

```
<xs:element name="carBrand">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="Ford"/>
         <xs:enumeration value="Chevrolet"/>
         <xs:enumeration value="Torino"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Restricción a valores que conforman a un patrón

- La forma mas poderosa de restrigir los posibles valores es a través de patrones:

```
<xs:element name="csmail">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:pattern value=
         "[a-z]{2,3}@cs\.uns\.edu\.ar"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Restricciones en la longitud

- Finalmente, también se puede restringir la cantidad de caracteres requeridos:

```
<xs:element name="password">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:minLength value="5"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

Reutilización de definiciones

- Las definiciones de datos pueden ser usadas de dos formas; *in situ*:

```
<xs:element name="prodId">
   <xs:simpleType> ... </xs:simpleType>
</xs:element>
```

- O bien, introduciendo un nombre para permitir un uso posterior:

```
<xs:simpleType name="Id">
   ...
</xs:simpleType>
```

Definición de tipos estructurados

- El potencial de los esquemas XML brilla recién al contemplar la definición de tipos de datos estructurados.
- Los esquemas XML permiten definir tipos estructurados de manera similar a Lisp o Prolog, es decir, mediante listas.
- Los tipos estructurados se describen como secuencias de otros tipos simples o estructurados.

Ejemplo de definición de tipos de estructurados

- Los tipos de datos estructurados se definen a través del tag <complexType>:

```
<xs:element name="data" type="contact"

<xs:complexType name="contact">
   <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

Extensión de tipos estructurados

- Un tipo de datos estructurados puede ser extendido de la siguiente forma:

```
<xs:complexType name="addressbook">
   <xs:complexContent>
   <xs:extension base="contact">
      <xs:sequence>
         <xs:element name="email" type="xs:string"/>
         <xs:element name="nick" type="xs:string"/>
      </xs:sequence>
   </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

Alternativas dentro de los tipos estructurados

- Dentro de un tipo estructurado es posible expresar la existencia de alternativas:

```
<xs:complexType name="staffDCIC">
   <xs:complexContent>
   <xs:choice>
      <xs:element name="teacher" type="teacher"/>
      <xs:element name="assistant" type="assistant"/>
   </xs:choice>
   </xs:complexContent>
</xs:complexType>
```

Repeticiones dentro de los tipos estructurados

- Dentro de un tipo estructurado es posible expresar la existencia de alternativas:

```
<xs:complexType name="courses">
   <xs:complexContent>
   <xs:sequence>
      <xs:element name="course" type="course"
      minOccurs="0" maxOccus="3"/>
   </xs:sequence>
   </xs:complexContent>
</xs:complexType>
```

Binding de operaciones a protocolos concretos

- La especificación del protocolo en concreto se hace mediante el elemento <binding>:

```
<binding type="stockPrice" name="b1">
   <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
      <soap:operation
      soapAction="http://example.com/getPrice"/>
         <input> <soap:body use="literal"/> </input>
         <output> <soap:body use="literal"/> </output>
   </operation>
</binding>
```

Modelos de comunicación y estilos de codificación SOAP

- La especificación de todo servicio web deja en claro la decisión tomada en torno al modelo de comunicación y a los estilos de codificación del SOAP Body y del mensaje SOAP en sí:
  - RPC/Encoded.
  - Document/Literal.
- Para un discusión más a fondo:
  - Keep up with the Web service styles.