XML Schema

Departamento de Informática

Universidad de Oviedo

Lenguajes de Esquemas

Esquema = definición de estructura de un conjunto de documentos XML

Validar = Chequear que un documento sigue un esquema

- Principal Ventaja: Protección de errores
- Otras aplicaciones: Edición, compresión, etc.

DTDs = un ejemplo de esquemas (con varias limitaciones)

XML Schema = desarrollo posterior del W3c

Existen Otros:

- RELAX-NG, Schematron, etc.

Objetivos de Diseño

- Sintaxis XML
- Soporte para Espacios de Nombres
- Mayor expresividad
  - Restricciones numéricas
  - Integridad dependientes del contexto
- Tipos de datos
  - Gran cantidad de tipos de datos predefinidos
  - Creación de tipos de datos por el usuario
- Extensibilidad
  - Inclusión/Redefinición de esquemas
  - Herencia de tipos de datos
- Soporte a Documentación

Ejemplo

alumnos.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.uniovi.es/alumnos"
           xmlns="http://www.uniovi.es/alumnos">
<xs:element name="alumnos">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="alumno" minOccurs="1" maxOccurs="200"
               type="TipoAlumno"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
    <xs:element name="nacim" type="xs:gYear"/>
  </xs:sequence>
  <xs:attribute name="dni" type="xs:integer"/>
 </xs:complexType>
</xs:schema>
```

Elemento raíz schema y espacio de nombres determinado

Permite especificar rangos de inclusión

Permite especificar tipos

Estructura del Schema

El esquema está formado por:

- Elemento raíz: **schema** del espacio de nombres
  - **http://www.w3.org/2001/XMLSchema**
- Atributo: **targetNamespace** indica el espacio de nombres que se está definiendo
- Subelementos:
  - Declaraciones globales de elementos y atributos
  - Definiciones de tipos de elementos y atributos
  - Anotaciones

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.uniovi.es/alumnos"
           xmlns="http://www.uniovi.es/alumnos">
   <xs:element name="alumnos">
     . . .
  </xs:element>
   . . .
</xs:schema>
```

Tipos Complejos vs Simples

Pueden declararse 2 tipos:

Complejos: Pueden contener sub-elementos y atributos

Ejemplo de Tipo Complejo

```
<alumno dni="9873435">
  <nombre>Jose</nombre>
  <apellidos>Bueno</apellidos>
</alumno>
```

Simples

Simples: No contienen sub-elementos ni atributos

Pueden aparecer dentro de elementos o en valores de atributos

Validación: esquemas e Instancias

Un documento XML Schema define un conjunto de documentos con una determinada estructura

Un documento XML puede validarse contra varios esquemas

- Puede asociaciarse explícitamente mediante el atributo **schemaLocation**
  - Utiliza 2 cadenas, el espacio de nombres y la URL del documento
- Si no se utiliza espacio de nombres, puede usarse **noNamespaceSchemaLocation**

alumnos.xml

```
<alumnos
  xmlns="http://www.uniovi.es/alumnos"
  xsi:schemaLocation="http://www.uniovi.es/alumnos
                      alumnos.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
. . .
</alumnos>
```

Validación: esquemas e instancias

alumnos.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.uniovi.es/alumnos"
           xmlns="http://www.uniovi.es/alumnos">
<xs:element name="alumnos">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="alumno" minOccurs="1" maxOccurs="200"
               type="TipoAlumno"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:complexTy
  <xs:sequenc
    <xs:eleme
    <xs:eleme
    <xs:eleme
  </xs:sequen
  <xs:attribu
 </xs:complexType>
</xs:schema>
```

alumnos.xml

```
<alumnos
  xmlns="http://www.uniovi.es/alumnos"
  xsi:schemaLocation="http://www.uniovi.es/alumnos
                      alumnos.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
. . .
</alumnos>
```

Los espacios de nombres deben coincidir.
También puede usarse:
xsi:noNameSpaceLocation

Tipos Anónimos vs. con nombre

```
<xs:element name="alumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

+ legible

```
…
<xs:element name="alumno" type="TipoPersona"/>
…
<xs:ComplexType name="TipoPersona">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:ComplexType>
```

+ Reutilizable

Otra posibilidad: Referencias

```
<xs:element name="alumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

```
<xs:element name="alumnos">
<xs:sequence>
  <xs:element ref="alumno" />
</xs:sequence>
</xs:element>
```

Tipos complejos: Creación a partir de tipos simples

```
<xs:element name="precio">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:decimal">
            <xs:attribute name="moneda" type="xs:string" />
          </xs:extension>
       </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

```
<precio moneda="euros">23.45</precio>
```

Tipos Complejos: Secuencia

Construcción básica mediante secuencia de elementos

```
<xs:element name="alumno">
  <xs:complexType>
  <xs:sequence>
    <xs:element name="nombre"    type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
    <xs:element name="nacim" type="xs:gYear"
                minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="dni" type="xs:integer"/>
 </xs:complexType>
</xs:element>
```

```
<alumno dni="9399390">
 <nombre>Juan</nombre>
 <apellidos>García García</apellidos>
 <nacim>1985</nacim>
</alumno>
```

Tipos Complejos: Alternativa

choice: Representa alternativas

OJO: Es una o-exclusiva

```
<xs:complexType name="Transporte">
   <xs:choice>
    <xs:element name="coche" type="xs:string"/>
     <xs:element name="tren" type="xs:string"/>
     <xs:element name="avión" type="xs:string"/>
   </xs:choice>
 </xs:complexType>
```

```
<transporte>
<coche>Renault R23</coche>
</transporte>
```

Tipos Complejos: Contenido Mixto

El contenido Mixto permite mezclar texto con elementos

```
 <xs:complexType name="TCom" mixed="true">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="emph" type="xs:string"/>
  </xs:choice>
</xs:complexType>

<xs:element name="comentarios" type="TCom" />
```

```
<comentarios>
 Es un poco <emph>listillo</emph>
</comentarios>
```

Secuencias no ordenadas

all = Todos los elementos en cualquier orden

En DTDs requería enumerar las combinaciones:

(A,B,C)|(A,C,B)|...|(C,B,A)

```
<xs:complexType name="TipoLibro">
  <xs:all>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="título" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:element name="libro" type="TipoLibro" />
```

```
<libro>
 <autor>Juanita la Loca</autor>
 <título>No estoy loca</título>
</libro>
```

```
 <libro>
   <título>El kigote</título>
   <autor>Cerbantes</autor>
  </libro>
```

Agrupaciones

Es posible nombrar agrupaciones de elementos y de atributos para hacer referencias a ellas

```
<xs:group name="nombApell">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:group>
```

```
<xs:complexType name="TipoAlumno">
 <xs:group ref="nombApell" />
 <xs:element name="carrera" type="xs:string"/>
</xs:complexType>
```

Tipos Simples

- Los tipos simples no pueden contener elementos o atributos
- Pueden ser:
  - Predefinidos o *built-in* (Definidos en la especificación)
    - Primitivos
    - Derivados
  - Definidos por el usuario
    - Restringiendo facetas de tipos predefinidos

Tipos simples Primitivos

string

boolean

number, float, double

duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth

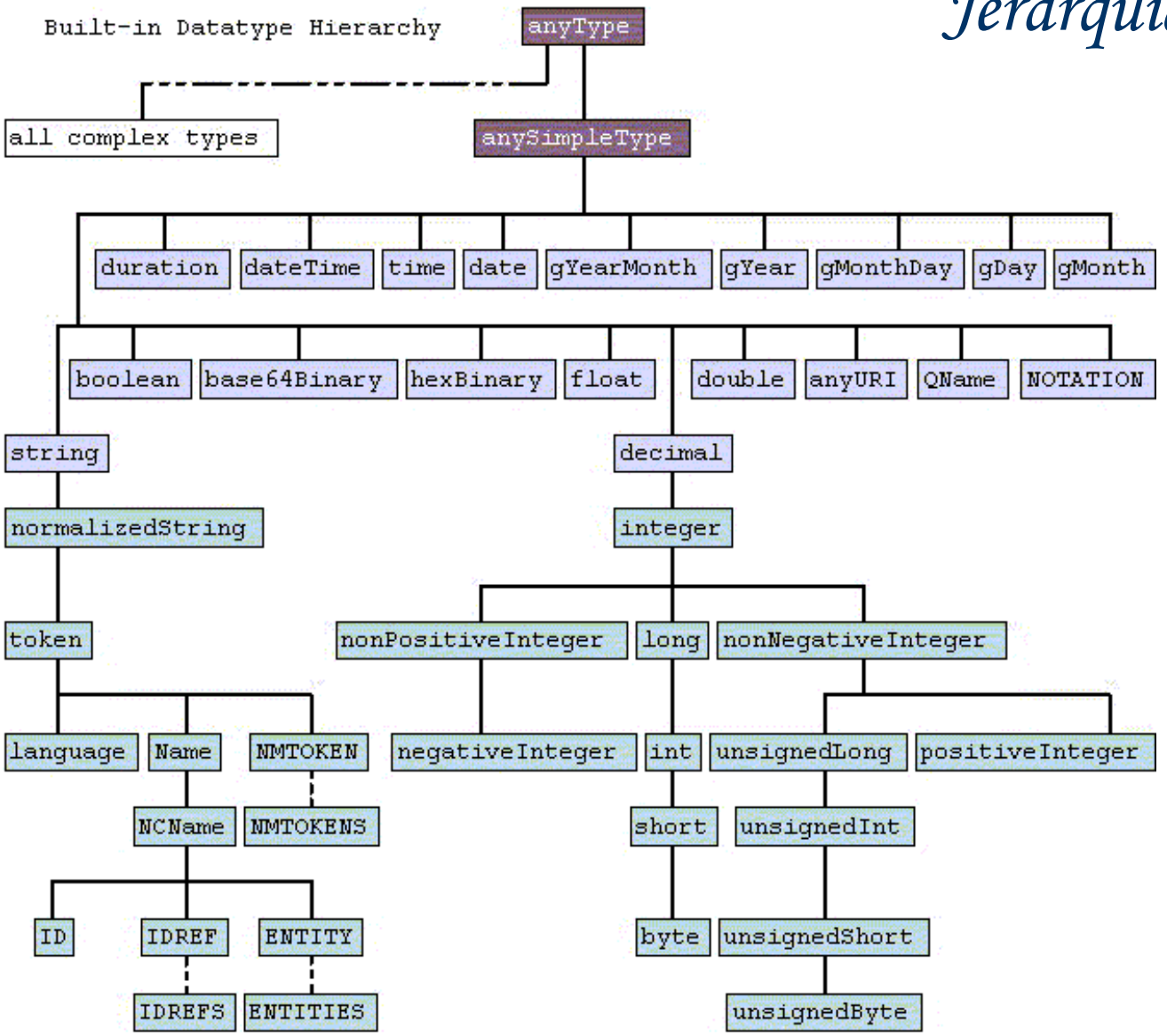
hexBinary, base64Binary

anyURI

QName = Nombre cualificado con espacio de nombres

NOTATION = Notación binaria (similar a DTD)

Jerarquía de tipos



Creación de nuevos tipos simples

Facetas

Los nuevos tipos se construyen mediante restricción de facetas:

- *length, minlength, maxlength:* Longitud del tipo de datos
- *pattern:* Restricciones sobre valores mediante expresiones regulares
- *enumeration:* Restringe a una determinada enumeración de valores
- *whitespace:* Define política de tratamiento de espacios (preserve/replace, collapse)
- *(max/min)(in/ex)clusive:* Límites superiores/inferiores del tipo de datos
- *totaldigits, fractionDigits:* número de dígitos totales y decimales

Enumeración

```
<xs:simpleType name="TipoCarrera">
<xs:restriction base="xs:token">
 <xs:enumeration value="Gestión"/>
 <xs:enumeration value="Sistemas"/>
</xs:restriction>
</xs:simpleType>
```

Restricciones sobre valores

```
<xs:simpleType name="mes">
<xs:restriction base="xs:integer">
  <xs:minInclusive value="1" />
  <xs:maxInclusive value="31" />
</xs:restriction>
</xs:simpleType>
```

Listas

```
<xs:simpleType name="ListaComponentes">
 <xs:list itemType="TipoComponente"/>
</xs:simpleType>

<xs:simpleType name="Tipoomponente">
<xs:restriction base="xs:nonNegativeInteger">
   <xs:maxInclusive value="255" />
</xs:restriction>
</xs:simpleType>
```

Se pueden aplicar las facetas: length, maxLength, minLength, enumeration

```
<xs:simpleType name="ColorRGB">
   <xs:restriction base="ListaComponentes">
     <xs:length value="3" />
   </xs:restriction>
 </xs:simpleType>
```

```
<color>255 255 0</color>
```

Uniones

```
<xs:simpleType name="TipoNota">
<xs:union>
  <xs:simpleType>
    <xs:restriction base="xs:float">
      <xs:maxInclusive value="10" />
      <xs:minInclusive value="0" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="No presentado" />
    </xs:restriction>
  </xs:simpleType>
</xs:union>
</xs:simpleType>

<xs:element name="nota" type="TipoNota" />
```

```
<nota> 5.75 </nota>
```

```
<nota> No presentado </nota>
```

Ejemplos de expresiones regulares

```
<xs:simpleType name="NIF">
 <xs:restriction base="xs:token">
   <xs:pattern value="\d{7,8}[A-Z]" />
 </xs:restriction>
</xs:simpleType>

<xs:element name="nif" type="NIF" />
```

```
<nif>9394173J</nif>
```

```
<nif>11079845M</nif>
```

Expresión	Posibles valores
Elemento \d	Elemento 2
a*b	b, ab, aab, aaab, …
[xyz]b	xb, yb, zb
a?b	b, ab
a+b	ab, aab, aaab, …
[a-c]x	ax, bx, cx

Expresiones Regulares

`[a-c]x`	ax, bx, cx
`[^0-9]x`	Carácter ≠ dígito seguido de x
`\Dx`	Carácter ≠ dígito seguido de x
`(pa){2}rucha`	paparucha
`.abc`	Cualquier carácter (1) seguido de abc
`(a|b)+x`	ax, bx, aax, bbx, abx, bax,...
`a{1,3}x`	ax, aax, aaax
`\n`	Salto de línea
`\p{Lu}`	Letra mayúscula
`\p{Sc}`	Símbolo de moneda

Fin