Análisis y Diseño de Sistemas de Información para Internet

3. Conceptos básicos de Schemas y documentos XML

Luís Rodríguez Baena (luis.rodriguez@upsam.net)

Universidad Pontificia de Salamanca (campus Madrid)
Facultad de Informática

Documentos validados

- Un documento validado es un documento bien formado en el que sus elementos cumplen una serie de reglas.
  - Esas normas pueden regular el número, nombre u orden de los elementos o atributos o el contenido de los mismos.
  - XML presenta dos mecanismos para asegurar esas reglas:
    - ✓ Definición del tipo de documento (DTD).
    - ✓ Esquemas XML.
- Las aplicaciones pueden necesitar utilizar documentos validos para:
  - Describir la estructura del documento de manera estricta.
  - Comunicar esa estructura a otras aplicaciones
  - Comprobar que están presentes los elementos requeridos.
  - Comprobar el cumplimiento de la estructura de un documento, los valores de los atributos y sus contenidos.
  - Suministrar valores predeterminados en los atributos no especificados.
- Estas operaciones se podrían realizar mediante programación de las aplicaciones:
  - Si un documento es utilizado por muchas aplicaciones este sistema puede ser engorroso.
  - Es más eficiente definir la estructura del documento y que los parser XML se encarguen de esta tarea.

Luis Rodríguez Baena, Facultad de Informática, 2008

Introducción a las DTD

- Una DTD hace una descripción formal de un vocabulario XML.
  - Puede ser compartida por varios documentos XML.
  - Permiten validar si un documento utiliza esa descripción.
- Pueden formar parte de un documento XML o ser un documento externo.
- Para utilizarla, el documento XML a validad debe tener una declaración del tipo de documento DOCTYPE.
  - Para DTD internas.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE elementoraíz[
   … declaración de la DTD …
]>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Introducción a las DTD (II)

- DOCTYPE para DTD externas:

  `<!DOCTYPE` *`elementoDelDocumento origen ubicación1 ubicación2`*`>`

  - *`elementoDelDocumento`* es el elemento raíz del documento XML.
  - *`origen`* puede contener alguna de las palabras claves `SYSTEM` o `PUBLIC`.
    - ✓ `SYSTEM` obliga a especificar la ubicación del documento utilizando un URI
    - ✓ `PUBLIC` permite validad el documento con un nombre sin necesidad de tener un acceso real a la DTD.
      - Se utiliza para DTD utilizadas ampliamente y permiten que determinadas aplicaciones la utilicen ya que las tienen implementadas de antemano.
  - *`ubicación1`* y *`ubicación2`* indican dónde está almacenada la DTD.
    - ✓ Si se trata de un origen de tipo `SYSTEM` será un URI.
    - ✓ Si se trata de un origen de tipo `PUBLIC` será un nombre ya aceptado.
      - En este caso la *`ubicación2`* será siempre un URI.
- Ejemplos:
  - Declaración DOCTYPE de tipo SYSTEM: Temperaturas (del tema 1).

```
<!DOCTYPE Temperaturas SYSTEM "Temperaturas.dtd">
```

  - Declaración DOCTYPE de tipo PUBLIC de un documento HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Definición de elementos

- Permite especificar el nombre de una etiqueta y su contenido.

  `<!ELEMENT` *`nombre contenido`*`>`

  - Contenido:

Contenido	Significado	Ejemplo
`#PCDATA`	Indica que se trata de un elemento de tipo texto.	`<!ELEMENT nombre (#PCDATA)>`
`(elem1, elem2...)`	Indica que el elemento contiene subelementos, es decir una lista de elementos hijos.	`<!ELEMENT empleado (nombre,dni)>`
`EMPTY`	Indica que se trata de un elemento vacío (ver apartado anterior)	`<!ELEMENT vacio #EMPTY>`
`ANY`	Puede contener cualquier tipo de contenido.	
Mixto	Pueden contener distintos tipos de elementos	`<!ELEMENT p (#PCDATA |b|i|div)*>`

Definición de elementos (II)

- Elementos descendientes.
  - Los elementos pueden contener otros elementos hijos.
  - Los subelementos que pueden contener se indican en la declaración de elemento mediante paréntesis.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE alumno SYSTEM "alumnos.dtd">
<alumno>
     <expediente>041056</expediente>
     <apellidos>Ramírez González</apellidos>
     <nombre>Juan María</nombre>
</alumno>
```

```
<!ELEMENT alumno (expediente,apellidos,nombre)>
<!ELEMENT expediente (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE alumno SYSTEM "alumnos.dtd">
<alumno>
  <expediente>041056</expediente>
  <nombre_completo>
    <apellidos>Ramírez González</apellidos>
    <nombre>Juan María</nombre>
  </nombre_completo>
</alumno>
```

```
<!ELEMENT alumno (expediente,nombre_completo)>
<!ELEMENT expediente (#PCDATA)>
<!ELEMENT nombre_completo (apellidos,nombre)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Definición de elementos (III)

- Número de hijos.
  - La lista puede llevar un sufijo que indica el número de hijos o la presencia o no de los mismos.

Sufijo	Significado
+	Uso obligatorio y múltiple. Permite la existencia de varios elementos de este tipo dentro del elemento padre, pero es necesario que al menos exista uno.
*	Uso opcional y múltiple. Pueden aparecer una o varias ocurrencias del elemento hijo.
?	Opcional y único. Puede existir una o ninguna ocurrencia de este elemento dentro del elemento hijo.
|	Equivalente a OR. Da la opción de utilizar un único elemento entre dos opciones que aparecerán separadas por la barra.

Definición de elementos (IV)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE alumnos SYSTEM "alumnos.dtd">
<alumnos>
  <alumno>
    <expediente>041056</expediente>
    <nombre_completo>
      <primer_apellido>Ramírez</primer_apellido>
      <segundo_apellido>González</segundo_apellido>
      <nombre>Juan María</nombre>
    </nombre_completo>
  </alumno>
  <alumno>
    <expediente>041034</expediente>
    <nombre_completo>
      <primer_apellido>Smith</primer_apellido>
      <nombre>John</nombre>
    </nombre_completo>
  </alumno>
</alumnos>
```

```
<!ELEMENT alumnos (alumno+)>
<!ELEMENT alumno (expediente,nombre_completo)>
<!ELEMENT expediente (#PCDATA)>
<!ELEMENT nombre_completo (primer_apellido,segundo_apellido?,nombre)>
<!ELEMENT primer_apellido (#PCDATA)>
<!ELEMENT segundo_apellido (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos

- Permiten incluir información adicional sobre un elemento, es decir, *metainformación*.
  - Representan las propiedades de un objeto.
  - No pueden contener subatributos ni cualquier otro elemento.
  - Se utilizan para incluir texto significativo pero con poca información, muy específica y desestructurada.
- La DTD deben declarar todos los atributos presentes en los elementos.
- Una única declaración de atributos puede declarar varios atributos para un elemento dado.
- Declaración de atributos:

  `<!ATTLIST` *`elemento nombreAtt tipoAtt`* `[`*`comportamiento`*`][`*`valorDefecto`*`]>`

  - *`elemento`*: elemento al que pertenece.
  - *`nombreAtt`*: nombre del atributo.

Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos (II)

- Tipos de atributos.
  - Determina el tipo de información que puede contener.

Tipo	Significado
`(valor1|valor2|...)`	Puede tomar el valor de alguna de las opciones separadas por barras
`CDATA`	Cadena
`ID`	Nombre no compartido por ningún otro atributo
`IDREF`	Referencia a un ID definido en algún otro lugar del documento
`IDREFS`	Lista separada por espacios que contiene una o más referencias a ID
`ENTITY`	Nombre de una entidad definida en el DTD
`ENTITIES`	Lista de entidades separadas por espacios
`NMTOKEN`	Un nombre XML compuesto por letras, números, guión bajo y puntos
`NMTOKENS`	Lista de NMTOKEN separados por espacios
`NOTATION`	Nombre de una notación definida en la DTD.

Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos (III)

- Tipo `CDATA`.
  - Cualquier valor de texto Unicode.
  - No se admitirían entidades de texto (por ejemplo `<`).
- Tipo enumerado: `(elem1|elem2|…)`.
  - Admite cualquiera de los valores que aparecen entre paréntesis.
  - Los valores deben ser texto XML válido.
- Identificador: `ID`.
  - Sólo puede haber un atributo de tipo `ID` por elemento.
  - El contenido debe ser un nombre XML válido (debe comenzar por un carácter).
  - Se utiliza para referenciar unívocamente un elemento de la DTD.

Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos (IV)

- Tipos `IDREF`, `IDREFS`.
  - El atributo hace referencia al valor de un atributo `ID` de algún otro elemento.
  - Si la referencia es a un único objeto, se utiliza `IDREF`, si son varios `IDREFS`.

```
<catalogo>
   <persona idPersona="MCS" rol = "Autor">
        Miguel de Cervantes Saavedra
   </persona>
   <persona idPersona="WS" rol = "Autor">
        William Shakespeare
   </persona>
   <libro autor ="MCS WS">
        La obra imposible de Cervantes y
    Shakespeare
   </libro>
</catalogo>
```

```
<!ELEMENT catalogo (persona+,libro+)>
<!ELEMENT persona (#PCDATA)>
<!ATTLIST persona idPersona ID rol (autor|ilustrador)>
<!ATTLIST libro autor IDREFS ilustrador IDREFS>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos (V)

- Tipo `NMTOKEN`, `NMTOKENS`.
  - Intermedio entre un CDATA y un ID.
  - Limita el valor del contenido permitiendo únicamente el uso de los caracteres válidos en un nombre XML.
    - ✓ Caracteres alfanuméricos, ideogramas, el punto, los dos puntos, el guión y el guión bajo.
    - ✓ A diferencia de un nombre XML pueden comenzar por cualquier carácter.
  - Herencia de SGML.
  - `NMTOKENS` sería una lista de `NMTOKEN` separada por comas.
- `ENTITY`, `ENTITIES`, `NOTATION`.
  - Hacen referencia a una entidad, una lista de entidades o una notación definida en la DTD.

Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos (VI)

- Predeterminados de atributos.
  - Permiten indicar si un atributo es o no opcional y el proceso que se debe hacer en caso de que no se encuentre.

Predeterminado	Descripción
`#REQUIRED`	El atributo debe aparecer en cada instancia del elemento.
`#IMPLIED`	El atributo es opcional
`#FIXED valorPredeterminado`	El atributo puede aparecer o no. Si aparece debe coincidir con el valor predeterminado.
`valorPredeterminado`	El atributo puede aparecer o no. Si aparece puede tomar cualquier valor válido, si no tomará el valor predeterminado.

Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos (VII)

- #IMPLIED

```
<!ATTLIST unElemento suAtributo CDATA #IMPLIED>

<unElemento suAtributo="cualquiera">Un valor</unElemento>
<unElemento>Otro valor</unElemento>
<unElemento suAtributo= "   .otros caracteres">Un valor
   mas</unElemento>
```

- #REQUIRED

```
<!ATTLIST unElemento suAtributo (valor1|valor2) #REQUIRED>

<unElemento suAtributo="valor1">Un valor</unElemento>
<unElemento>Otro valor</unElemento> <!-- esto no es válido -->
<unElemento suAtributo= "valor2">Un valor mas</unElemento>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Atributos (VIII)

- #FIXED *valorPredeterminado*

```
<!ATTLIST unElemento suAtributo CDATA #FIXED "valorFijo">

<unElemento suAtributo="valorFijo">Un valor</unElemento>
<unElemento>Otro valor</unElemento> <!-- esto no es válido -->
<!-- esto no es válido -->
<unElemento suAtributo= "Otro valor">Un valor mas</unElemento>
```

- valorPredeterminado

```
<!ATTLIST unElemento suAtributo (valor1|valor2) "valor1">

<unElemento suAtributo="valor2">Un valor</unElemento>
<unElemento>Otro valor</unElemento> <!- suAtributo toma "valor1" -->
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Ejemplo de DTD y documento XML

```
<!ELEMENT alumnos (alumno+)>
<!ELEMENT alumno (nombre_completo, notas?)>
<!ELEMENT nombre_completo (primer_apellido,segundo_apellido?,nombre)>
<!ELEMENT primer_apellido (#PCDATA)>
<!ELEMENT segundo_apellido (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT notas (asignatura)>
<!ELEMENT asignatura (examen, calificación)>
<!ELEMENT examen EMPTY>
<!ELEMENT calificación (#PCDATA)>
<!ATTLIST alumno expediente CDATA #REQUIRED>
<!ATTLIST asignatura
          código IDREF #REQUIRED
          denominación CDATA #REQUIRED>
<!ATTLIST examen
          fecha NMTOKEN #REQUIRED
          veces_matriculada NMTOKEN #REQUIRED
          numero_convocatoria NMTOKEN #REQUIRED>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Ejemplo de DTD y documento XML (II)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE alumnos SYSTEM "alumnos.dtd">
<alumnos>
  <alumno expediente="041056">
    <nombre_completo>
      <primer_apellido>Ramírez</primer_apellido>
      <segundo_apellido>González</segundo_apellido>
      <nombre>Juan María</nombre>
    </nombre_completo>
    <notas>
      <asignatura código="I315"
                  denominación="Análisis y Diseño de Sistemas de información para Internet">
        <examen fecha="02-02-2008" numero_convocatoria="1" veces_matriculada ="1" />
        <calificación></calificación>
      </asignatura>
    </notas>
  </alumno>
  <alumno expediente="041034">
    <nombre_completo>
      <primer_apellido>Smith</primer_apellido>
      <nombre>John</nombre>
    </nombre_completo>
  </alumno>
</alumnos>
```

Ejemplos

- Realizar una DTD para el almacenamiento de CD musicales con las especificaciones detalladas en el ejemplo del tema 2.
- Diseñar una DTD que sirva de soporte para las especificaciones de la práctica 1.
- Comprobar los documentos de instancia utilizando un validador que admita DTD como XML Validation (www.xmlvalidation.com).

Luis Rodríguez Baena, Facultad de Informática, 2008

Esquemas XML

- Un Esquema XML es un documento XML que contiene un descripción formal de la estructura de un documento XML valido.
  - Un Esquema de Lenguaje de Esquemas del W3C es un esquema XML escrito en una determinada sintaxis recomendada por el W3C.
- Diferencias entre DTD y esquemas.

DTD	Esquemas
Estándar propio basado en las DTD de SGML	Sintaxis basada en XML
No permite el uso de los espacios de nombres (Namespaces)	Permite el uso de namespaces.
No permite la herencia.	Permite la herencia.
Sólo puede especificar 0, 1 o varios ejemplares de un elemento.	Define exactamente la cantidad de elementos.
Cada DTD valida un solo documento XML.	Un esquema valida varios documentos XML basados en los espacios de nombres.
No tiene soporte DOM	Se puede visualizar y manipular con DOM, ya que es XML
Utiliza comentarios simples.	Además de los comentarios habituales, define anotaciones con dos tipos de comentarios: los de usuario y los de programa.

Esquemas XML (II)

- Diferencias entre DTD y esquemas.

DTD	Esquemas
Todos los datos se tratan como textos.	Define más de 40 tipos base.
No permite definir tipos de usuario.	Permite definir nuevos tipos mediante la restricción y de la extensión de los tipos base e incluso de otros tipos de usuario..
Sólo los atributos pueden definir listas de valores.	Define listas de valores tanto con elementos como con atributos..
Sólo los atributos permiten definir valores fijos y por omisión.	Define valores fijos y por omisión tanto con elementos como con atributos.
No puede agrupar atributos.	Agrupa atributos para elementos diferentes.
Define elementos vacíos.	Define elementos vacíos y con contenido nulo.

Luis Rodríguez Baena, Facultad de Informática, 2008

Esquemas XML
Ejemplo de DTD y Schema

```
<!ELEMENT Temperaturas (Ciudades+)>
<!ELEMENT Ciudades (Ciudad+)>
<!ELEMENT Ciudad (Nombre,Dia,Hora,TemperaturaActual,TemperaturaMaxima,TemperaturaMinima)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Dia (#PCDATA)>
<!ELEMENT Hora (#PCDATA)>
<!ELEMENT TemperaturaActual (#PCDATA)>
<!ELEMENT TemperaturaMaxima (#PCDATA)>
<!ELEMENT TemperaturaMinima (#PCDATA)>
<!ATTLIST TemperaturaActual unidad (Farenheit|Celsius) #REQUIRED>
<!ATTLIST TemperaturaMaxima unidad (Farenheit|Celsius) #REQUIRED>
<!ATTLIST TemperaturaMinima unidad (Farenheit|Celsius) #REQUIRED>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Esquemas XML
Ejemplo de DTD y Schema (II)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Temperaturas">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" ref="Ciudades" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Ciudades">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="Ciudad" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Ciudad">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Nombre" />
        <xs:element ref="Dia" />
        <xs:element ref="Hora" />
        <xs:element ref="TemperaturaActual" />
        <xs:element ref="TemperaturaMaxima" />
        <xs:element ref="TemperaturaMinima" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Esquemas XML
Ejemplo de DTD y Schema (III)

```
  <xs:element name="Nombre" type="xs:string" />
  <xs:element name="Dia" type="xs:date" />
  <xs:element name="Hora" type="xs:time" />
  <xs:element name="TemperaturaActual" type="Temperatura" />
  <xs:element name="TemperaturaMaxima" type="Temperatura" />
  <xs:element name="TemperaturaMinima" type="Temperatura" />
  <xs:complexType name="Temperatura">
      <xs:simpleContent>
        <xs:extension base="xs:decimal">
          <xs:attribute name="unidad" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="Farenheit" />
                <xs:enumeration value="Celsius" />
                <xs:enumeration value="Kelvin" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Estructura básica de un esquema

- Un esquema tiene la misma estructura básica de un archivo XML.
  - Prologo del esquema.
  - Elementos del esquema.
- El prólogo, además está formado por dos elementos:
  - Declaración XML.
    `<?xml version="1.0" encoding="iso-8859-1"?>`
  - Declaración del elemento `schema` con el espacio de nombres al que hace referencia.
    `<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">`

Luis Rodríguez Baena, Facultad de Informática, 2008

Espacios de nombre

- Permiten indicar a un analizador XML que vocabulario utilizar cuando analice un documento.
  - Sólo puede tener utilidad si se mezcla más de un vocabulario XML en un único documento.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<agenda>
  <persona dni="123456C">
    <primerApellido>Pérez</primerApellido>
    <segundoApellido>Gómez</segundoApellido>
    <nombre>Ana María</nombre>
    <telefonos>
      <telefono tipo="casa">+34912343456</telefono>
      <telefono tipo="móvil">+34678675678</telefono>
    </telefonos>
  </persona>
</agenda>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<compañia>
    <nombre>Una empresa cualquiera</nombre>
    <telefono>3432343234</telefono>
  </compañia>
```

  - Si queremos utilizar los dos vocabularios, un analizador podría encontrar ambigüedades entre el elemento `nombre` de la agenda y el elemento `nombre` de la compañía.
- Un espacio de nombre (namespace) tiene dos propósitos en XML.
  - Distinguir elementos y atributos de distintos vocabularios y distintos significados pero con el mismo nombre.
  - Agrupar todos los elementos y atributos relacionados de una aplicación XML para que el software pueda reconocerlos fácilmente.

Luis Rodríguez Baena, Facultad de Informática, 2008

Espacios de nombre (II)

- Un espacio de nombre XML asocia elementos y atributos a una referencia de una URI en la que se ha definido un vocabulario.
- Precisa de dos elementos:
  - Un prefijo que permita asociar cada elemento a un vocabulario.
  - Una declaración que permita asociar ese prefijo a una URI con la definición del vocabulario.
- Prefijos.
  - Para utilizar un elemento de un espacio de nombre concreto hay que utilizar el **nombre cualificado** del elemento.
  - Ese nombre está formado por:
    - ✓ Un prefijo que identifica el espacio de nombre.
    - ✓ El carácter dos puntos (:).
    - ✓ El nombre de la etiqueta (parte local del nombre cualificado).
  - El prefijo puede estar formado por cualquier carácter XML.
    - ✓ No se puede utilizar el carácter dos puntos ni comenzar por la secuencia xml.
  - El nombre local no puede utilizar el carácter dos puntos.

Luis Rodríguez Baena, Facultad de Informática, 2008

Espacios de nombre (III)

- Asociar el prefijo a una URI.
  - El prefijo debe asociarse a una URI que indique de dónde se tomará el vocabulario relacionado con ese prefijo.
  - Para asociarlo se utiliza el atributo `xmlns:prefijo`.
    - ✓ prefijo es el prefijo que vamos a utilizar en el documento XML.
  - El formato de la instrucción será:

```
<prefijo:espacioDeNombre xmlns:prefijo="URI">
```

  - El ámbito de la URI es el elemento en el que se declara y en todos los elementos descendientes.
  - El prefijo se declara en el elemento de la parte superior o en cualquier ancestro del mismo.
    - ✓ Puede ser en cualquier elemento raíz del documento o alguno de nivel inferior.

Luis Rodríguez Baena, Facultad de Informática, 2008

Espacios de nombre (IV)

- El siguiente ejemplo utiliza los vocabularios de agenda y empresa combinados.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<agenda>
  <persona dni="123456C">
    <primerApellido>Pérez</primerApellido>
    <segundoApellido>Gómez</segundoApellido>
    <nombre>Ana María</nombre>
    <telefonos>
      <telefono tipo="casa">+34912343456</telefono>
      <telefono tipo="móvil">+34678675678</telefono>
    </telefonos>
  </persona>
  <cia:compañia xmlns:cia="http://www.micompania.com/compania-sintax">
    <cia:nombre>Una empresa cualquiera</cia:nombre>
    <cia:telefono>3432343234</cia:telefono>
  </cia:compañia>
</agenda>
```

- Cómo al declararse un espacio de nombre se declara para todos los descendientes sería posible hacer también esto:

```
<cia:compañia xmlns:cia="http://www.micompania.com/compania-sintax">
    <nombre>Una empresa cualquiera</nombre>
    <telefono>3432343234</telefono>
 </cia:compañía>
```

Espacios de nombre (V)

- Lo importante de un espacio de nombre no es el prefijo, sino la URI.
  - Distintos documentos XML pueden tener distintos prefijos, pero si comparten la URI también comparten el espacio de nombres.
- Es posible establecer el espacio de nombres predeterminado de un documento (o de los elementos descendientes).
  - Se utiliza la sintáxis `<etiqueta xmlns="URI">`
- También es posible indicar varios espacios de nombre en un documento.
  - Un elemento puede tener distintas parejas `xmlns:prefijo="URI"`que indicarán distintos prefijos y su asociación a espacios de nombre.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<agenda xmlns="http://www.compania.com/agenda-sintax"
        xmlns:cia="http://www.micompania.com/compania-sintax">
  <persona dni="123456C">
    <primerApellido>Pérez</primerApellido>
    <segundoApellido>Gómez</segundoApellido>
    <nombre>Ana María</nombre>
    <telefonos>
      <telefono tipo="casa">+34912343456</telefono>
      <telefono tipo="móvil">+34678675678</telefono>
    </telefonos>
  </persona>
  <cia:compañia>
    <cia:nombre>Una empresa cualquiera</cia:nombre>
    <cia:telefono>3432343234</cia:telefono>
  </cia:compañia>
</agenda>
```

Declaraciones del esquema

- Espacios de nombre para los esquemas.
  - El espacio de nombre principal para los esquemas se encuentra en `http://www.w3.org/2001/XMLSchema`.

    ```
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    ```

    ✓ Cada elemento del esquema que tenga el prefijo `xs` se asociará con ese espacio de nombres.
  - Algunos elementos se encuentran en otro espacio de nombre, por lo que el preámbulo del esquema también suele incluir el espacio de nombres situado en `http://www.w3.org/XMLSchema-instance`.

    ✓ Es común añadir también este espacio de nombres.

    ```
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    ```


Luis Rodríguez Baena, Facultad de Informática, 2008

Declaraciones del esquema (II)

- Atributos de la declaración de esquema.
  - targetNamespace.
    - ✓ Indica mediante una URI un espacio de nombre de destino al esquema.
  - elementFormDefault.
    - ✓ Define si vamos a utilizar nombres de elementos cualificados (con el prefijo) o no cualificados (sin el prefijo) para asociarlos al namespace.
      - Admite los valores qualifed o unqualifed.
  - attributeFormDefault.
    - ✓ Define si vamos a utilizar nombres de atributos cualificados o no cualificados.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
         targetNamespace="http//www.meteorologia.org”
         elementFormDefault ="unqualifed"
         attributeFormDefault="unqualifed">
  <element name="Temperaturas">
    <complexType>
      <sequence>
        <element minOccurs="1" maxOccurs="1" ref="Ciudades" />
      </sequence>
    </complexType>
  </element>
```

Asociación de un documento a un esquema

- Asociación de los elementos no incluidos en ningún espacio de nombres.
  - El atributo `noNamespaceSchemaLocation` permite asociar los elementos de un documento XML a un esquema.
    - ✓ El atributo está dentro del espacio de nombres de documentos de instancia http://www.w3.org/XMLSchema-instance.
      - Es necesario hacer referencia a el en la declaración principal del elemento.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<Temperaturas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:noNamespaceSchemaLocation="Temperaturas.xsd">
  <Ciudades>
    <Ciudad>
      <Nombre>Madrid</Nombre>
      <Dia>2008-09-05</Dia>
      <Hora>11:52:10</Hora>
      <TemperaturaActual unidad="Celsius">22</TemperaturaActual>
      <TemperaturaMaxima unidad="Celsius">25</TemperaturaMaxima>
      <TemperaturaMinima unidad="Celsius">19</TemperaturaMinima>
    </Ciudad>
</Temperaturas>
```

Asociación de un documento a un esquema (II)

- Asociación de los elementos asociados a un determinado espacio de nombre.
  - El atributo `schemaLocation` permite asociar los elementos de un documento XML a un esquema.
    - ✓ El atributo está dentro del espacio de nombres de documentos de instancia http://www.w3.org/XMLSchema-instance.
      - Es necesario hacer referencia a el en la declaración principal del elemento.
    - ✓ Hay que incluir un espacio de nombre seguido de un URL de la ubicación del esquema a partir del espacio de nombre.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<Temperaturas xmlns="http://www.colimbo.net/Temperaturas"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://www.colimbo.net/Temperaturas
              http://www.colimbo.net/Pruebas/schema/Temperaturas.xsd">
  <Ciudades>
    <Ciudad>
      <Nombre>Madrid</Nombre>
      <Dia>2008-09-05</Dia>
      <Hora>11:52:10</Hora>
      <TemperaturaActual unidad="Celsius">22</TemperaturaActual>
      <TemperaturaMaxima unidad="Celsius">25</TemperaturaMaxima>
      <TemperaturaMinima unidad="Celsius">19</TemperaturaMinima>
    </Ciudad>
</Temperaturas>
```

  - Se dispone de dos vocabularios distintos. xsi para el espacio de nombres de XML para instancias (XML Schema namespace for instances) y otro para las temperaturas.

Declaración de elementos

- Un esquema XML está compuesto básicamente por elementos anidados y `xs:element`.
- Declaración de elementos.
  - El elemento `xs:element` declara un elemento del esquema.
    - ✓ Cómo mínimo deberá tener el atributo `name`.
    - ✓ Para los elementos hoja, habrá que especificar el tipo de dato mediante el atributo `type`.
      - El atributo `type` permite indicar el tipo de dato del elemento.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="nombre_completo" type="xs:string" />
</xs:schema>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<nombre_completo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:noNamespaceSchemaLocation="Universidad.xsd">
  Juan María López Herranz
</nombre_completo>
```

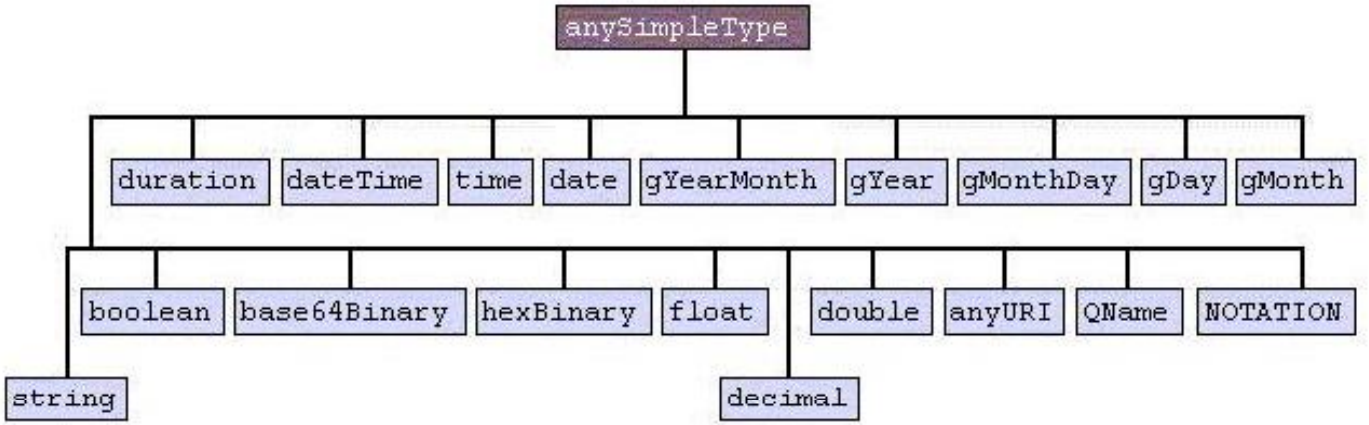

Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos de datos

- Los esquemas XML permiten dos tipos de contenido:
  - Contenido simple.
    - ✓ No puede contener elementos anidados.
  - Contenido complejo.
    - ✓ Pueden contener elementos anidados y atributos.
- Los elementos de contenido simple pueden tener dos clases de tipos de datos.
  - Tipos primitivos.
    - ✓ Se definen en si mismos ya que no se crean a partir de ningún otro tipo de dato.
  - Tipos derivados.
    - ✓ Se definen en base a otros tipos.
    - ✓ Se pueden considerar subtipos de los tipos primitivos.

Luis Rodríguez Baena, Facultad de Informática, 2008

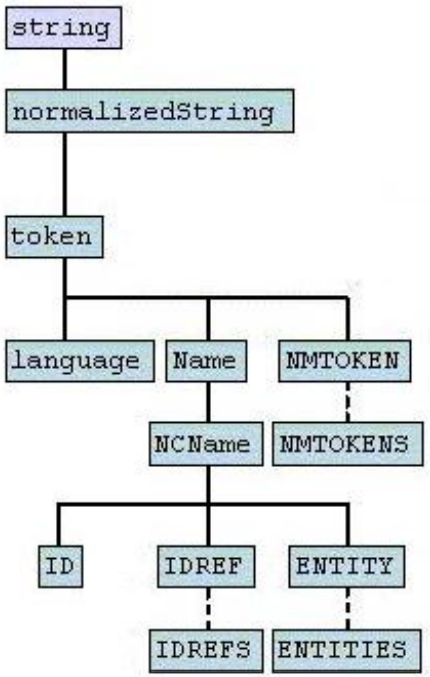
Tipos de datos primitivos



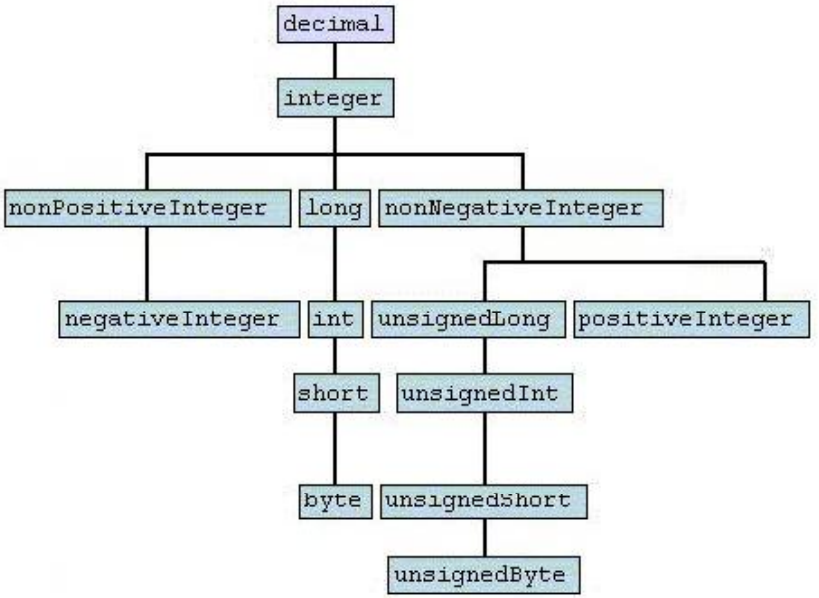
Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos de datos derivados integrados

Tipos numéricos derivados



Tipos de cadena derivados



Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos de datos numéricos integrados

TIPO	DESCRIPCIÓN	FORMATO
`decimal`	Número decimal de precisión arbitraria	7,08
`integer`	Número entero	
`negativeInteger`	< 0	
`nonNegativeInteger`	≤ 0	
`positiveInteger`	> 0	
`nonPositiveInteger`	≥ 0	
`long`	De –9223372036854775808 a 9223372036854775807	
`int`	De -2147483648 a 2147483647	
`short`	De -32768 a 32767	
`byte`	De –128 a 127	
`unsignedLong`	De 18446744073709551615	
`unsignedShort`	De 65535	
`unsignedInt`	De 4294967295	
`unsignedByte`	De 255	
`float`	Número de coma flotante de simple precisión y 32 bits	12.56E3, 12, 12.560 Valores especiales: 0, -0, INF, -INF, NaN (infinito, not a number)
`double`	Número de coma flotante de doble precisión y 64 bits	12.56E3, 12, 12560, 0, -0, INF, -INF, NaN

Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos de datos de fecha integrados

TIPO	DESCRIPCIÓN	FORMATO
`date`	Fecha	YYYY-MM-DD
`dateTime`	Fecha y hora	YYYY-MM-DD hh:mm:ss
`duration`	Duración respecto iso 8601	PnYnMnDTnHnMnS
`gDay`	Día	---DD (3 guiones)
`gMonth`	Mes	--MM (2 guiones)
`gMonthDay`	Mes y día	--MM-DD (2 guiones)
`gYear`	Año	YYYY
`gYearMonth`	Año y mes	YYYY-MM
`time`	Hora	hh:mm:ss.sss

Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos de datos de cadena y otros tipos integrados

Tipos de dato de cadena		
TIPO	**DESCRIPCIÓN**	**FORMATO**
`string`	Caracteres	"cadena"
`normalizedString`	Cadena sin tabuladores ni retornos de carro.	
`token`	Cadena normalizada en espacios en los que los espacios al inicio y al final de la cadena han sido suprimidos	
`Qname`	Nombre calificado de un espacio de nombre	prefijo:elemento
`name`	Un nombre XML 1.0 válido.	
`NCName`	Nombre no calificado	
`ID`	Equivalente al atributo **ID** de la DTD	
`IDREF`	Equivalente al atributo **IDREF** de la DTD	
`IDREFS`	Equivalente al atributo **IDREFS** de la DTD	
`ENTITY`	Equivalente al atributo **ENTITY** de la DTD	
`ENTITIES`	Equivalente al atributo **ENTITIES** de la DTD	
`NMTOKEN`	Equivalente al atributo **NMTOKEN** de la DTD	
`NMTOKENS`	Equivalente al atributo **NMTOKENS** de la DTD	
`NOTATION`	Equivalente al atributo **NOTATION** de la DTD	

Tipos de dato de cadena		
TIPO	**DESCRIPCIÓN**	**FORMATO**
`hexBinary`	Cadena de dato hexadecimal	312D322D33 (Cadena ASCII de "1-2-3")
`boolean`	true, false, 1, 0	
`anyURI`	Referencia URI	"http://www.w3.org"
`language`	Identificador de lenguaje como ES, FR, EN, etc.	

Tipos complejos

- XML también permite definir nuevos tipos de datos.
  - La declaración `xs:simpleType` permite definir nuevos tipos de datos no divisibles en unidades más pequeñas a partir de tipos de datos existentes.
  - La declaración `xs:complexType` permite definir tipos de datos complejos.
    - ✓ Se trata de tipos de datos compuestos de otros más simples.
    - ✓ Es un mecanismo similar a las estructuras de otros lenguajes de programación.
- Un tipo complejo puede contener elementos hijos, atributos o una combinación de ambos.
  - Si se trata de un elemento de nivel superior deberá tener el atributo `name`.
    - ✓ De esta forma será posible referenciar el tipo en otro lugar del esquema.
    - ✓ Si aparece dentro de un elemento `xs:element` no tendrá atributo `name` y define un tipo anónimo para ese elemento.
- Los tipos complejos pueden definir un **modelo de contenido**.
  - Un modelo de contenido es una descripción formal de la estructura y del contenido permisible de un elemento.

Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos complejos: xs:sequence

- Indica que los elementos hijos deben aparecer en dicha posición en el documento de instancia.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="alumno">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="expediente" type="xs:string" />
        <xs:element name="nombre_completo">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="primer_apellido" type="xs:string" />
              <xs:element name="segundo_apellido" type="xs:string" />
              <xs:element name="nombre"  type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<alumno xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Universidad.xsd">
  <expediente>041035</expediente>
  <nombre_completo>
    <primer_apellido>López</primer_apellido>
    <segundo_apellido>Herranz</segundo_apellido>
    <nombre>Juan María</nombre>
  </nombre_completo>
</alumno>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos complejos: xs:all

- Indica que los elementos hijos deben aparecer en el documento aunque en cualquier orden.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="alumno">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="expediente" type="xs:string" />
        <xs:element name="nombre_completo">
          <xs:complexType>
            <xs:all>
              <xs:element name="primer_apellido" type="xs:string" />
              <xs:element name="segundo_apellido" type="xs:string" />
              <xs:element name="nombre"  type="xs:string"/>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<alumno xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Universidad/Universidad.xsd">
  <expediente>041035</expediente>
  <nombre_completo>
    <segundo_apellido>Herranz</segundo_apellido>
    <nombre>Juan María</nombre>
    <primer_apellido>López</primer_apellido>
  </nombre_completo>
</alumno>
```

Tipos complejos: xs:choice

- El documento puede contener alguna de las opciones contenidas.
  - Pueden ser a su vez otros elementos.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="alumno">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="expediente" type="xs:string" />
        <xs:element name="nombre_completo">
          <xs:complexType>
            <xs:choice>
              <xs:element name="nombre"  type="xs:string"/>
              <xs:sequence>
                <xs:element name="primer_apellido" type="xs:string" />
                <xs:element name="segundo_apellido" type ="xs:string" />
                <xs:element name="nombre"  type="xs:string"/>
              </xs:sequence>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<alumno xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Universidad.xsd">
  <expediente>041035</expediente>
  <nombre_completo>
    <nombre>Juan María López Herranz</nombre>
  </nombre_completo>
</alumno>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos complejos: xs:group

- Un grupo de modelo es la parte de la definición de un tipo complejo que describe el contenido de un elemento.
  - El elemento `nombre_completo` se podría considerar un grupo implícito.
  - También es posible crear un grupo de modelo con nombre y referenciarlo más tarde con el atributo `ref` desde otra parte del esquema.

```
<?xml version="1.0" encoding="utf-8" ?>
<ejemplo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="XSDSchema2.xsd">
  <unElemento>
    <elemento1>wwwww</elemento1>
    <elemento2>xxxxx</elemento2>
  </unElemento>
  <otroElemento>
    <elemento1>yyyyy</elemento1>
    <elemento2>zzzzz</elemento2>
  </otroElemento>
</ejemplo >
```

  - El ejemplo, repite los elementos `elemento1` y `elemento2`.
    - ✓ Es posible hacer un grupo que referencie estos 2 elementos.

Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos complejos: xs:group (II)

- El esquema que incluye al grupo...

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:group name="unGrupo">
    <xs:sequence>
      <xs:element name="elemento1" type="xs:string" />
      <xs:element name="elemento2" type="xs:string" />
    </xs:sequence>
  </xs:group>
  <xs:element name="ejemplo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="unElemento">
          <xs:complexType>
            <xs:group ref ="unGrupo" />
          </xs:complexType>
        </xs:element>
        <xs:element name="otroElemento">
          <xs:complexType>
            <xs:group ref ="unGrupo" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Tipos complejos:
xs:any

- El elemento `any` permite incluir elementos con contenidos a los que se puede controlar el nivel de validación.
  - En principio admite cualquier tipo de contenido.
  - Es posible hacer la validación utilizando otro nombre de esquema.
  - Se puede utilizar…
    - ✓ Cuando se pretende validar el contenido de un elemento frente a otro esquema.
    - ✓ Cuando inicialmente se desconocen las marcas del vocabulario que se va a escribir.
  - El atributo `namespace` permite incluir otro espacio de nombres para la validación.
  - El atributo `processContent` permite especificar cómo se realizará la validación.
    - ✓ Puede tomar los valores `skip`, `lax` o `strict`.
  - El siguiente ejemplo permite que el elemento `algunaPáginaWeb` contenga datos XHTML bien formados.
    - ✓ El atributo `skip` indica al analizador que no es necesario validar el código XHTML

```
<xs:element name="algunaPáginaWeb">
   <xs:complexType>
      <xs:any
          namespace=http://www.W3.org/1999/xhtml
          processContent="skip" />
   <xs:complexType>
<xs:element>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos complejos:
Control de la cardinalidad

- Tanto `element` como `sequence` o `choice` admiten atributos que permiten establecer la cardinalidad de los elementos.
  - `minOccurs="enteroNoNegativo"`.
    - ✓ Permite establecer el número mínimo de ocurrencias del elemento.
  - `maxOccurs="enteroNoNegativo | unbounded"`.
    - ✓ Permite establecer el número máximo de ocurrencias del elemento.
  - Por omisión, tanto `minOccurs` como `maxOccurs` toman el valor 1.
  - Su efecto es similar a los sufijos ?, + o * de las DTD.
- Ejemplos:

```
<xs:element name="alumno" minOccurs="1" maxOccurs="unbounded">
```

✓ Establece un número entre 1 y n de elementos `alumno`.

```
<xs:element name="expediente" type="xs:string" minOccurs="1"
  maxOccurs="1" />
```

✓ Siempre debe existir un elemento `expediente`.

```
<xs:element name="segundo_apellido" type="xs:string" minOccurs="0"
  maxOccurs="1"/>
```

✓ Pueden existir 0 o 1 elemento `segundo_apellido`.

Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos complejos:
Control de la cardinalidad (II)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="alumnos">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="alumno" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="expediente" type="xs:string" minOccurs="1" maxOccurs="1" />
              <xs:element name="nombre_completo" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="primer_apellido" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="segundo_apellido" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="nombre"  type="xs:string" minOccurs="1" maxOccurs="1"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```


Luis Rodríguez Baena, Facultad de Informática, 2008

Tipos complejos:
Control de la cardinalidad (III)

- El esquema anterior…
  - Obliga a la existencia de, al menos, un elemento `alumno`.
  - Por cada alumno siempre debe haber al menos un elemento `expediente` y un elemento `nombre_completo`.
  - El elemento `segundo_apellido` puede o no aparecer.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Universidad.xsd">
  <alumno>
  <expediente>041035</expediente>
  <nombre_completo>
    <primer_apellido>López</primer_apellido>
    <segundo_apellido>Herranz</segundo_apellido>
    <nombre>Juan María</nombre>
  </nombre_completo>
  </alumno>
  <alumno>
    <expediente>041067</expediente>
    <nombre_completo>
      <primer_apellido>Smith</primer_apellido>
      <nombre>John</nombre>
    </nombre_completo>
  </alumno>
</alumnos>
```


Luis Rodríguez Baena, Facultad de Informática, 2008