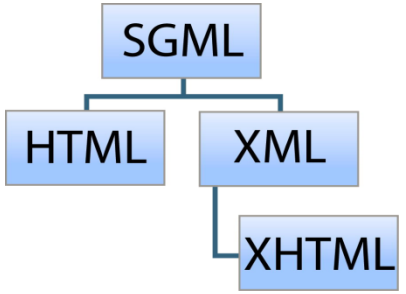
Lenguaje XML

conceptos básicos

definición de XML

- **XML** (eXtensible Markup Language)
- Metalenguaje extensible de etiquetas desarrollado por la W3C (World Wide Web Consortium).
- Es una aplicación de SGML (Standard Generalized Markup Language)
- XML se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.
- Es un lenguaje para definir estructuras de datos, abierto a cualquier estructura (independiente de la plataforma).
- Es comprensible por un ser humano, lo que permite su edición en forma directa, a diferencia, por ejemplo, de una cadena binaria.



conceptos básicos

documento XML

- XML expresa la información estructurada, de una manera abstracta y reutilizable.
- Estructurada porque se compone de partes bien definidas, y esas partes se componen a su vez de otras. Estas partes se llaman elementos, y se las señala mediante etiquetas.
- Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Es un pedazo de información con un sentido claro y definido.
- Las etiquetas tienen la forma <nombre>, donde "nombre" es el nombre del elemento que se está señalando y se encierra entre signos "<" y ">".

Un mensaje en XML:

```
<message>Hola, mundo</message>
```

estructura

ejemplo de documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mensaje SYSTEM "mensaje.dtd">
<mensaje tipo="email">
  <remitente>
    <nombre>John Doe</nombre>
    <email>jdoe@eldu.es</email>
  </remitente>
  <destinatario>
    <nombre>Juan Pérez</nombre>
    <email>juan.perez@gmail.com</email>
  </destinatario>
  <asunto>Saludo</asunto>
  <texto>
    <parrafo>Hola, ¿qué tal?</parrafo>
  </texto>
</mensaje>
```

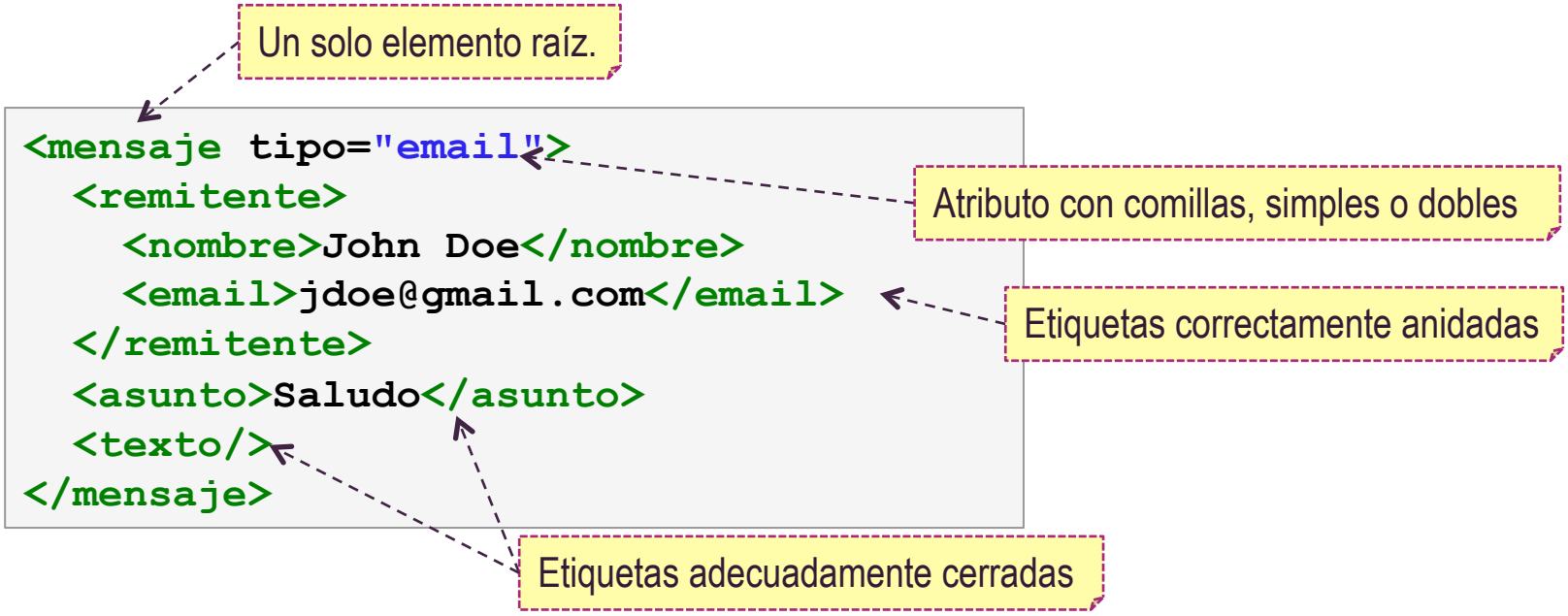
estructura

mismo ejemplo como árbol

formato

documento bien formado

- Documento XML bien formado: cumple con definiciones básicas de formato.
- Puede validarse con cualquier analizador sintáctico (*parser*).
- Estructura jerárquica. Los documentos XML deben seguir una estructura estrictamente jerárquica en sus etiquetas:



formato

documento bien formado

- Etiquetas vacías. XML permite elementos sin contenido:
- **<elemento-sin-contenido/>**
- Mayúsculas y minúsculas. XML es sensible a mayúsculas y minúsculas (*case-sensitive*). Se las considera caracteres distintos.

- Caracteres equivalentes a un "espacio en blanco" dentro de una etiqueta:
  - Espacio en blanco (Unicode / ASCII 32).
  - Tabulación (Unicode / ASCII 9).
  - Retorno de carro (Unicode / ASCII 13).
  - Salto de línea (Unicode / ASCII 10).

```
<mensaje tipo="email">

</mensaje>
```

```
<mensaje
   tipo="email">

</mensaje>
```

formato

partes de un documento XML

- Prólogo:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

- Cuerpo


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE listado SYSTEM "validata.dtd">
<!-- Descripcion de un listado de empleados -->
<listado>
  <empleado>
    <dni>12345678A</dni>
    <nombre>Juan Pérez</nombre>
    <departam>Viajes</departam>
    <telefono tipo="movil">612345678</telefono>
  </empleado>
  <empleado>
  ...
```

formato

partes de un documento XML

- Elemento. Pueden tener:
  - Contenido en forma de más elementos.
  - Texto.
- Comentario.
  - No es procesado.
  - Sirve para aclaraciones.
  - No se pueden poner dentro de un elemento ni CDATA.
- Atributo. Característica adicional de un elemento. Se encierran entre comillas.

```
<empleado>
   <dni>12345678A</dni>
</empleado>
```

```
<!-- Descripcion de un listado
     de empleados -->
```

```
<telefono tipo="movil">
   612345678</telefono>
```

formato

caracteres de escape

Existen caracteres conflictivos si aparecen en documentos XML, como "<" o ">". Para evitar problemas, tienen una representación de escape. Ejemplos:

Entidad	Caracter
&amp;	&
&lt;	<
&gt;	>

Entidad	Caracter
&apos;	'
&quot;	"

- Secciones CDATA: cuando hay muchos caracteres especiales, se puede utilizar una sección CDATA para facilitar lectura.

```
<comparacion>6 es &lt; 7 &amp; 7 &gt; 6</comparacion>

<comparacion><![CDATA[6 es < 7 & 7 > 6]]></comparacion>
```

validación y namespaces

vocabulario

Un vocabulario sirve para definir las etiquetas que se deben utilizar en la creación de determinados tipos de documentos, así como sus normas de utilización.

W3C ha definido dos estándares para la creación de vocabularios XML:

- Document Type Definition (DTD)
- XML Schema Definition (XSD)

Junto con la validación estructural, se comprueban las reglas definidas en el vocabulario.

Cuando un documento cumple ambos requisitos se dice que es un documento válido.

Actualmente, DTD se considera *legacy*, privilegiando el uso de XSD. En este curso sólo se describe XSD.

validación y namespaces

XML Schema Definition

Permite definir el tipo del contenido de los elementos o atributos de un XML, en formato XML, a diferencia de un DTD:

Prólogo XML

Elemento raíz (schema)

Namespace

Nombre elemento raíz

Definición tipo complejo

Secuencia con tres elementos

Atributo de elemento "emple"

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.example.com/emple"
        elementFormDefault="qualified">
    <element name="emple">
        <complexType>
            <sequence>
                <element name="nombre" type="string"/>
                <element name="apell1" type="string"/>
                <element name="fecnac" type="date"/>
            </sequence>
            <attribute name="titulo" type="string"/>
        </complexType>
    </element>
</schema>
```

validación y namespaces

XML Schema Definition

```
<element name="emple">
 <complexType>
  <sequence>
   <element name="nombre" type="string"/>
   <element name="apel1" type="string"/>
   <element name="fecnac" type="date"/>
  </sequence>
  <attribute name="titulo" type="string"/>
 </complexType>
</element>
```

XML que cumple el XSD anterior:

```
<?xml version="1.0" encoding="UTF-8"?>
<emple xmlns="http://www.example.com/nombre"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.example.com/nombre personaSchema.xsd"
    titulo="Sr.">
    <nombre>Juan</nombre>
    <apel1>Pérez</apel1>
    <fecnac>1977-08-25</fecnac>
</emple>
```

validación y namespaces

referencia a un XSD

Para referenciar al schema:

- Se declara el namespace del XML.
- Se asocia el namespace al XSD.
- El XSD puede estar en una ruta física o una URL.

```
<?xml version="1.0" encoding="UTF-8"?>
<emple xmlns="http://www.example.com/nombre"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
       http://www.example.com/nombre personaSchema.xsd"
    titulo="Sr.">
    <nombre>Juan</nombre>
    <apel1>Pérez</apel1>
    <fecnac>1977-08-25</fecnac>
</emple>
```

Definición de namespace default

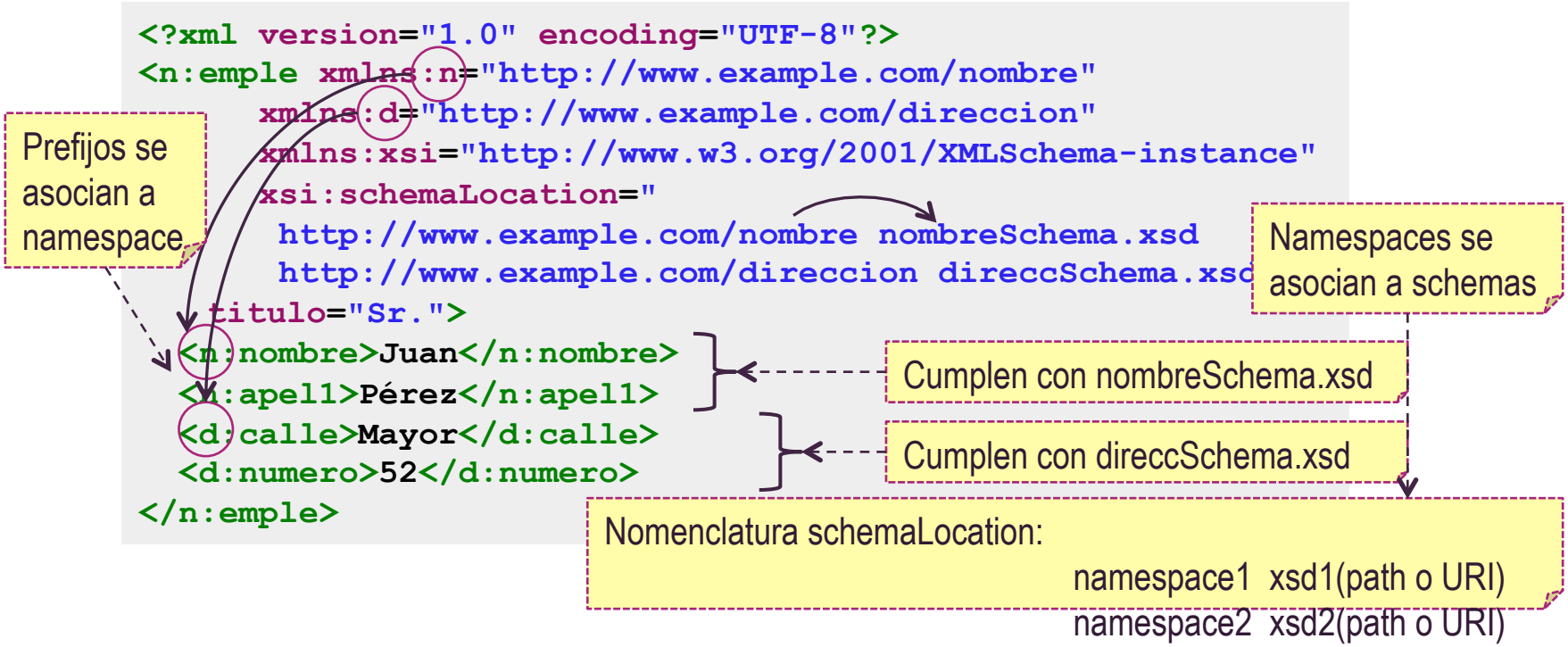
Asociación namespace con XSD

validación y namespace

namespaces

Podría suceder que en un XML, hayan etiquetas que se validan con distintos esquemas. En ese caso se distinguen a través del uso de namespaces.

- Namespace: un identificador (URI), un prefijo y un XSD asociado.




```
<?xml version="1.0" encoding="UTF-8"?>
<n:emple xmlns:n="http://www.example.com/nombre"
    xmlns:d="http://www.example.com/direccion"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      http://www.example.com/nombre nombreSchema.xsd
      http://www.example.com/direccion direccSchema.xsd
  titulo="Sr.">
  <n:nombre>Juan</n:nombre>
  <n:apel1>Pérez</n:apel1>
  <d:calle>Mayor</d:calle>
  <d:numero>52</d:numero>
</n:emple>
```

parsing y binding

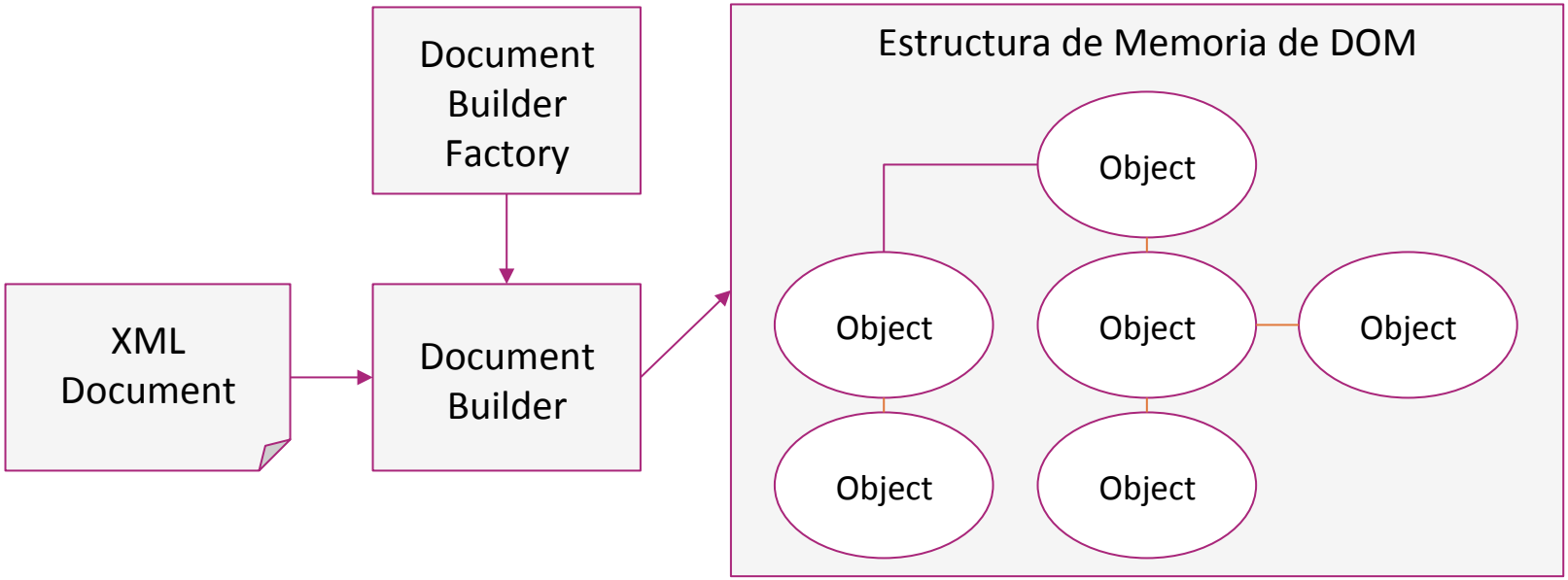
concepto de XML parsing

- Se refiere a interpretar los datos de un documento XML.
- Se extraen los datos y se convierten a un objeto para que la información pueda ser procesada según las necesidades.
- Tipos de parser:
  - SAX (Simple API for XML)
  - DOM (Document Object Model)
  - StAX (Streaming API for XML)

parsing y binding

parsing con DOM

- Carga la estructura de árbol del documento en memoria, antes de manipular la información contenida en el XML.
- Bidireccional: puede ser usada para la generación de documentos XML, ya que el documento entero se encuentra disponible en la memoria.



validación y namespace

Ejercicio práctico: Validación de un XML con un XSD

Resumen del ejercicio:

- Crear un XSD con una definición básica de elementos y atributos.
- Crear un XML que es válido según el XSD.
- Agregar características al XSD, y ajustar el XML para que las cumpla.
- Crear otro XSD, e importarlo en el primero.
- Crear otro XML, que utiliza los elementos del XSD importado.

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

- Importar en Eclipse el proyecto Java llamado cp-xml-xsd.
- En la carpeta src, crear el siguiente schema con el nombre basic-employee.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="id" type="xsd:int"/>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="address" type="Address"/>
        <xsd:element name="email" type="xsd:string"
                     minOccurs="0" maxOccurs="1"/>
        <xsd:element name="hire-date" type="xsd:date"/>
        <xsd:element name="salary" type="xsd:double"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="Address">
    <xsd:sequence>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="number" type="xsd:int"/>
      <xsd:element name="city" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" />
  </xsd:complexType>
</xsd:schema>
```

all: En el XML pueden seguir cualquier orden.

Datos del empleado, de distintos tipos.

sequence: En el XML deben seguir orden establecido.

Datos de la dirección

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

- En la carpeta src, crear un archivo con el nombre basic-employee123.xml, que sea válido según basic-employee.xsd, y que tenga los siguientes datos:

José García
id:123

jose.garcia@gmail.com

Dirección trabajo: Gran Vía, 52
Madrid
Salario: 45123,45

Contratación: 20/05/2001

Nota: En el XML, la fecha va en formato yyyy-MM-dd, y el separador decimal es punto. En el tipo de dirección, colocar "Trabajo"

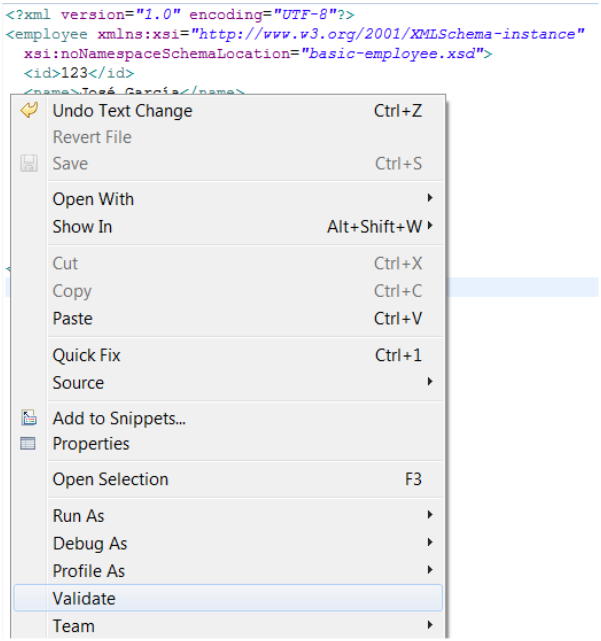
- Para declarar el XSD asociado al XML, utilizar la nomenclatura sin namespace:

```
<employee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="basic-employee.xsd">
  <id>123</id>
  ...
</employee>
```

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

- Validar basic-employee123.xml, utilizando Eclipse:



Si Eclipse indica "*No grammar constraints (DTD or XML schema) detected for the document.*", es porque no encuentra el XSD, ya sea porque está mal escrito o mal declarado. Si en este caso se utiliza xsi:schemaLocation en vez de xsi:noNamespaceSchemaLocation, no lo encontraría.

Nota: Existen también formas programáticas de validar el XML, utilizando las librerías de parsing XML, como DOM.

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

Continuando:

- Se pide cambiar en el XSD el tipo de dato del atributo "type" de Address, por uno tipo enumerado, para acotar sus posibles valores a una lista predefinida:

```
<xsd:complexType name="Address">
  <xsd:sequence>
    ...
  </xsd:sequence>
  <xsd:attribute name="type" type="AddressType" />
</xsd:complexType>
<xsd:simpleType name="AddressType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="PARTICULAR" />
    <xsd:enumeration value="BUSINESS" />
  </xsd:restriction>
</xsd:simpleType>
```

Valores predefinidos

- Validar nuevamente el XML y corregir los errores que surjan a partir de este cambio de definición.

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

- A continuación, se analiza cómo trabajar con namespaces en un XSD.
- En la carpeta src, copiar basic-employee.xsd en un nuevo archivo ns-employee.xsd, y declarar en la cabecera de este último un namespace asociado a employee, que incluye el nombre y el targetNamespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:emp="http://example.com/employee"
    targetNamespace="http://example.com/
employee">
  <xsd:element name="employee">
  ...
```

namespace: *http://example.com/employee*
prefijo:
emp

- Al incluir el namespace, los nombres de los tipos complejos deben ser referenciados con el prefijo asociado. Por lo tanto, se cambian las siguientes líneas:

```
<xsd:element name="address" type="emp:Address"/>
...
<xsd:attribute name="type" type="emp:AddressType" />
```

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

En la carpeta src, copiar basic-employee123.xml en un nuevo archivo employee123.xml, y realizar los siguientes ajustes para que sea validado con ns-employee.xsd:

- Declarar el namespace en el tag raíz del XML, utilizando la misma nomenclatura que en el XSD:

```
xmlns:emp="http://example.com/employee"
```

- Declarar el schema como schemaLocation, el que incluye el nombre del namespace separado por un espacio de la ubicación del XSD.

```
xsi:schemaLocation="http://example.com/employee ns-employee.xsd"
```

- Cambiar la declaración del tag raíz del XML, utilizando el prefijo del namespace:

```
<emp:employee ...
```

- Validar el XML con Eclipse, y realizar las correcciones necesarias para que cumpla el XSD.

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

- A continuación, se analiza cómo importar un XSD dentro de otro, para reutilizar sus definiciones en un XML.
- En la carpeta src, incluir otro schema con el nombre company.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://example.com/company">
    <xsd:complexType name="company">
      <xsd:sequence>
        <xsd:element name="cif" type="xsd:string"/>
        <xsd:element name="name" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

company.xsd sólo define un tipo complejo de dato, no un elemento. Define su propio namespace.

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

- En la carpeta src, copiar ns-employee.xsd en un nuevo archivo employee.xsd, y realizar la importación de company.xsd, justo antes de la definición de "employee":

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
  <xsd:import namespace="http://example.com/company"
              schemaLocation="company.xsd" />
  ...
```

- Declarar el namespace de company en el tag raíz del XSD:

```
xmlns:comp="http://example.com/company"
```

- Agregar un elemento "company" opcional al final de los de employee:

```
<xsd:element name="employee">
        ...
        <xsd:element name="company" type="comp:company"
                     minOccurs="0" maxOccurs="1" />
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

En el type, se utiliza el prefijo asociado al namespace.

validación con XSD

Ejercicio práctico: Validación de un XML con un XSD

- En la carpeta src, crear otro archivo employee456.xml que sea válido según employee.xsd, con los siguientes datos, los que incluyen el dato de la compañía.

Jorge Rojas
myCompany, CIF:B-82387770
id:456

Dirección particular: Calle Mayor, 3
Madrid
Salario: 35456,12

Contratación: 01/07/2005

Nota: Incluir un tag <company> con sus elementos definidos en su XSD, dentro del tag <employee>.

- Como company.xsd está importado en employee.xsd, no se utilizan prefijos en los tags de company.
- Validar el XML con Eclipse, y realizar los ajustes necesarios hasta que cumpla el XSD.
- Nótese que no se utiliza el email. Como es opcional, el XML es válido.