Resumen XSD

**2.4 Introduccion**

Debido a la gran variedad de DTD el consorcio W3C creó los XSD (XML Schema) como sustitución de estos para definición de documentos XML

El XSD tiene grandes ventajas como:

* Tiene un propio sistema de datos
* Soporta espacios de nombres
* Reutilizable, compatible con POO
* Sencillo al ser XML

**2.4.1 Asociar un esquema a un archivo XML**

Con los XSD no es necesario especificar en el XML ninguna asociación. Únicamente hay que escribir en el espacio de nombre el atributo xmlns y definir el archivo del esquema

Ejemplo:

<liga xmlns: XSI = "http://www.w3.org/2001/XMLSchema-instance"

XSI: noNamespaceSchemaLocation = "lliga.xsd" >

**2.4.2 Definir un archivo de esquema**

Un XSD:

* Opcionalmente tener una declaración XML
* Sólo tiene un elemento raíz <schema>

<? xml version = "1.0" ?>

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

...

</ xs: schema >

Xs es el alias para hacer referencia a las demás etiquetas

Manera de definir un elemento:

<xs: element name=”nombre” type=”xsi:string”/>

No podremos validar ninguna etiqueta que no tenga un tipo igual al del xsd

**minOccurs:** Atributo que permite definir cuantas veces tiene que salir un elemento como mínimo

**maxOccurs:** Atributo que permite definir cuantas veces tiene que salir un elemento como máximo

Ejemplo:

<xs: element name=”nombre” />

<xs: element name=” apellido” maxOccurs =”2” />

Quiere decir que solo puede haber 2 apellidos

**Tipos simples personales**

Se permite definir tipos de datos personales con la etiqueta **<simpleType>** y **list** permite definir que un elemento puede contener listas de valores.

<xs: element name= “partidos” >

<xs: simpleType>

<xs: list itemType = “xs: date” />

</xs: simpleType>

</xs : elemento >

Siendo un ejemplo de xml valido

<partidos > 2011-01-07 2011-01-15 2011-01-21 </partidos>

El atributo **unión** sirven para hacer que se puedan modificar diferentes tipos de contenido

El atributo **restricción** se pueden crear tipos de datos que solo acepten una determinada condición

**Extensión** sirve para añadir características extra a los tipos

* **Elementos con contenido de tipo complejo**

**Atributos**: Solo los elementos de tipo complejo pueden incluirlos, su etiqueta es atribute, los atributos siempre son opcionales

**SimpletContent,** será el contenido de complexType si el elemento solo contiene texto

**ComplexContent** permite definir extensiones o restricciones a un tipo complejo

**Elementos sin contenido:**

<xs: complexType />

**Contenido mezclado**

Se define poniendo el atributo mixed =”true” en la definición de <complexType>