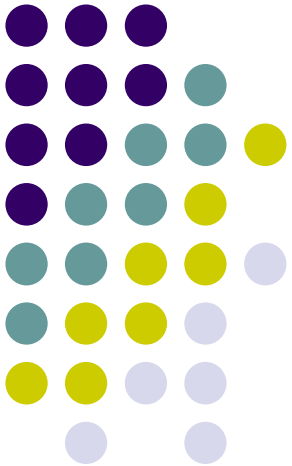
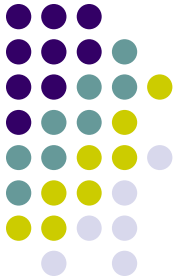
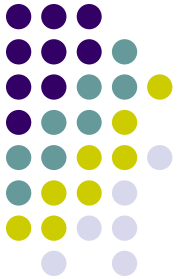
XML



Agenda



- XML
- Ventajas y Desventajas

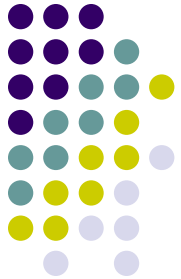


XML

XML

XML, (eXtensible Markup Language), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

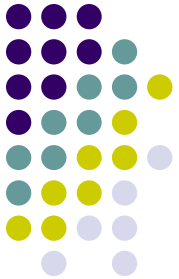
http://www.w3schools.com/xml/default.asp



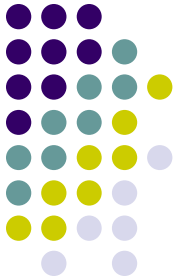
XML

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, etc.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan. También tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.



Estructura



Estructura

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable.

Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de partes con información. 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 este como un elemento, 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.

XML

-Texto plano bien formateado.
-Extensible
-Portable
-Independiente de la plataforma y del lenguaje
-Posibilidad de validar el contenido (DTD – Esquema)

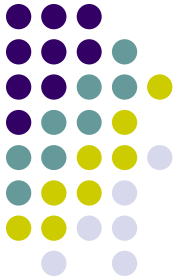
<Etiqueta> DATO </Etiqueta>

```
<?xml version="1.0" ?>
- <AUTO>
    <COLOR>verde</COLOR>
    <VELOCIDAD>200Km/h</VELOCIDAD>
    <MOTOR>6 cilindros</MOTOR>
  </AUTO>
```

Estructura

```
<?xml version=”1.0”?>
<mensaje>
    <remitente servidor=”smtp.carloserver.com.ar”>
        <nombre>Juan Carlos</nombre>
        <mail>jc@palermo.edu</mail>
    </remitente>
    <destinatario>
        <nombre>Carlos Mendez</nombre>
        <mail>president@anillacovs.gov</mail>
    </destinatario>
    <asunto>Hola Carlo</asunto>
    <texto>¿Hola que tal?</texto>
</mensaje>
```

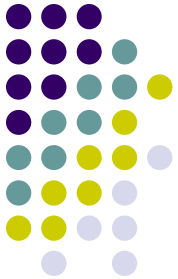
Documentos XML bien formados



Se llama documentos "bien formados" a los documentos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, ser analizados correctamente por cualquier "parser".

Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.

Documentos XML bien formados



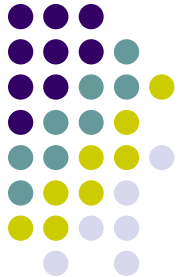
Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, sólo puede tener un elemento inicial.

Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

El XML es sensible a mayúsculas y minúsculas.

Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. Los nombres tienen alguna característica en común.

Documentos XML Validos



Que un documento sea "bien formado" solamente habla de su estructura sintáctica básica, es decir que se componga de elementos, atributos y comentarios como XML manda que se escriban.

Ahora bien, cada aplicación que utiliza XML, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento .

Esta relación entre elementos se especifica en un documento externo (expresada como DTD (Document Type Definition) o como XSchema).

Al confeccionar uno de estos documentos se crea una definición que equivale a crear un nuevo lenguaje de marcado, para una aplicación específica. Estamos creando la estructura que tendrán nuestros documentos XML para que sean validos.

DTD

Document Type Definition es una descripción de estructura y sintaxis de un documento XML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD. De esta forma, dichos documentos, pueden ser validados.
http://www.w3schools.com/dtd/default.asp

¿Qué describe una DTD?

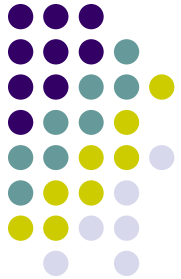
Elementos: indican qué etiquetas son permitidas y el contenido de dichas etiquetas.
Estructura: indica el orden en que van las etiquetas en el documento.
Anidamiento: indica qué etiquetas van dentro de otras.

DTD

persona.dtd

```
<!ELEMENT lista_de_personas (persona*)>
<!ELEMENT persona (nombre, fechanacimiento?, genero?, numeroseguridadsocial?)>
<!ELEMENT nombre (#PCDATA) >
<!ELEMENT fechanacimiento (#PCDATA) >
<!ELEMENT genero (#PCDATA) >
<!ELEMENT numeroseguridadsocial (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lista_de_personas SYSTEM “persona.dtd">
<lista_de_personas>
        <persona>
                <nombre>José García</nombre>
                <fechanacimiento>25/04/1984</fechanacimiento>
                <genero>Varón</genero>
        </persona>
</lista_de_personas>
```



DTD

Limitaciones de la DTD

Un esquema basado en una DTD tiene bastantes limitaciones. Una DTD no permite definir elementos locales que sólo sean válidos dentro de otros elementos (aninados). Además no es posible indicar a qué tipo de dato (número, fecha, moneda) ha de corresponder un atributo o el texto de un elemento.

El XML Schema soluciona estos problemas.

Schema

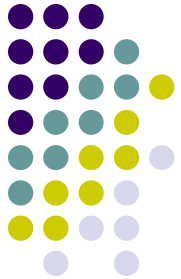
XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así, una percepción del tipo de documento con un nivel alto de abstracción.
http://www.w3schools.com/schema/default.asp

Schema

pedido.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name=“pedido" type=“Pedido"/>
<xs:complexType name=“Pedido">
  <xs:sequence>
     <xs:element name=“articulo" type="xs:string"/>
     <xs:element name=“cantidad" type="xs:decimal"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<pedido xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=“pedido.xsd">
   <articulo>Metanol</articulo>
   <cantidad>55</cantidad>
</pedido>
```



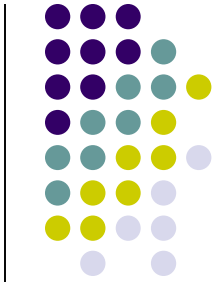
Parsers

DOM

El Document Object Model, comúnmente llamado DOM, define un conjunto de interfaces para manejar un documento XML. El parser lee el documento completo y construye un árbol en la memoria, de forma que el código puede entonces utilizar las interfaces DOM para manipular ese árbol.

Es posible moverse a través del árbol para ver lo que el documento original contenía, se pueden borrar secciones del árbol, reordenarlo, añadir nuevas ramas, etc.

DOM - Limitaciones



DOM proporciona un rico conjunto de funciones que pueden usarse para interpretar y manipular un documento XML, pero esas funciones tienen un precio.

DOM construye un árbol en la memoria con el documento entero. Si el documento es muy grande ser requiere una cantidad significativa de memoria.

DOM crea objetos que los representan todo en el documento original, incluyendo elementos, texto, atributos y espacios en blanco. Si solo se tiene interés en una pequeña parte del documento original, es extremadamente costoso crear todos esos objetos que nunca se usarán.

Un parser DOM tiene que leer el documento entero antes de que el código pueda tomar el control. Para documentos muy grandes esto puede causar un retraso significativo.

Estas son algunas de las debilidades derivadas del diseño del Document Object Model; independientemente de estas, DOM es una forma muy útil de parsear documentos
XML.

SAX

Simple API for XML (SAX) tiene diversas características que solucionan las limitaciones de DOM.

Un parser SAX envía eventos al código. El parser le dice cuando ha encontrado el comienzo y/o fín del documento, elemento, texto, etc. Uno decide que eventos son importantes y que tipo de estructura de datos desea crear para almacenarlos. Si no se salva explícitamente los datos de un evento, se perderán.

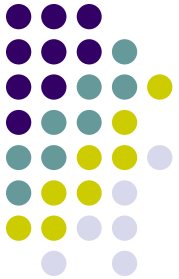
Un parser SAX no crea ningún objeto en absoluto, simplemente envía eventos hacía su aplicación.

Un parser SAX empieza a enviar eventos tan pronto como se inicia. Su código recibirá un evento cuando el parser encuentre el inicio del documento, cuando encuentre el inicio de un elemento, texto, etc. (generar resultados inmediatamente sin tener que esperar hasta que el documento entero haya sido parseado).

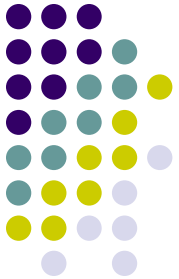
SAX - Limitaciones

Los eventos SAX no tienen estado. Cuando un parser SAX encuentra texto en un documento XML, envía un evento a su código. Este evento solo le indica el texto encontrado; no le dice que elemento contenía ese texto. Si quiere saberlo, tiene que escribir el código para la gestión de estado usted mismo.

Los eventos SAX no son permanentes. Si su aplicación necesita una estructura de datos que modele el documento XML, debe escribir ese código usted mismo. Si necesita acceder a los datos de un evento SAX y no los ha almacenado en su código, tiene que parsear el documento de nuevo.



Ventajas y Desventajas del XML

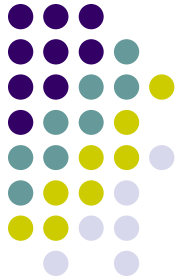


Ventajas

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

- El Parser es un componente estándar, no es necesario crear uno específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

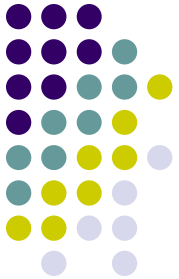


Desventajas

- El parseo es una operación costosa en tiempo.

- La relación entre la información transmitida y la definición de la estructura del documento XML es muy desproporcionada. En muchos casos hay mas definición de etiquetas que información útil.

- En casos donde se trabaja con grandes volúmenes de datos el XML es descartado (Ej: en intercambio de resúmenes de cuenta entre bancos donde hay archivos de texto de 2 GB)



XSLT