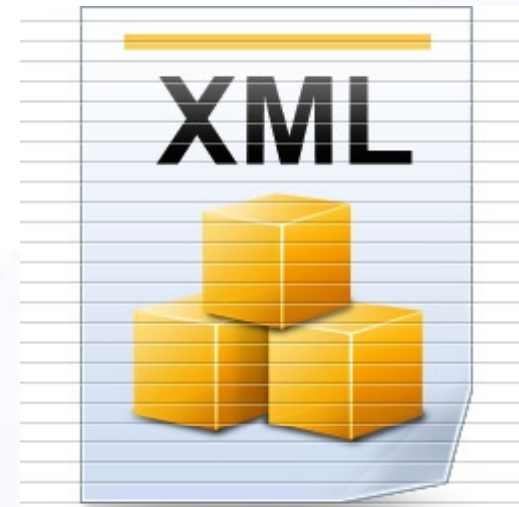


Fundamentos de XML para Java



Indice

- 👉 *Introducción XML*
- 👉 Validación XML
- 👉 Parseadores JAVA
- 👉 DOM
- 👉 JDOM
- 👉 SAX



➤ *Nacimiento:*

- Existía la necesidad de un mecanismo para el intercambio de información.
- Cada sistema informático era capaz de interpretar un conjunto de datos no genéricos.
- Aparece en los 70 de mano de IBM como GML (Generalized Markup Language)
- En 1986 la ISO lo normalizó, creando SGML (Standard Generalized Markup Language), ISO 8879
- Mejor ejemplo de SGML fue HTML (Hypertext Markup Language)
- En 1998 aparece XML como evolución de SGML.



➤ *Descripción y características:*

- Diseñado para transportar y almacenar información de una forma generalizada
- Permite el almacenamiento estructurado de la información basado en etiquetas (tags)
- Dispone de un conjunto de reglas para definir etiquetas semánticas que organizan el documento en distintas secciones
- Los principales objetivos de XML son:
 - XML debe ser directamente utilizable sobre Internet.
 - XML debe soportar una amplia variedad de aplicaciones.
 - Los documentos XML deben ser legibles por humanos y razonablemente claros
 - El diseño de XML debe ser formal y conciso



Readable™



➤ *Ejemplo de un nota:*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
```



➤ *Estructura:*

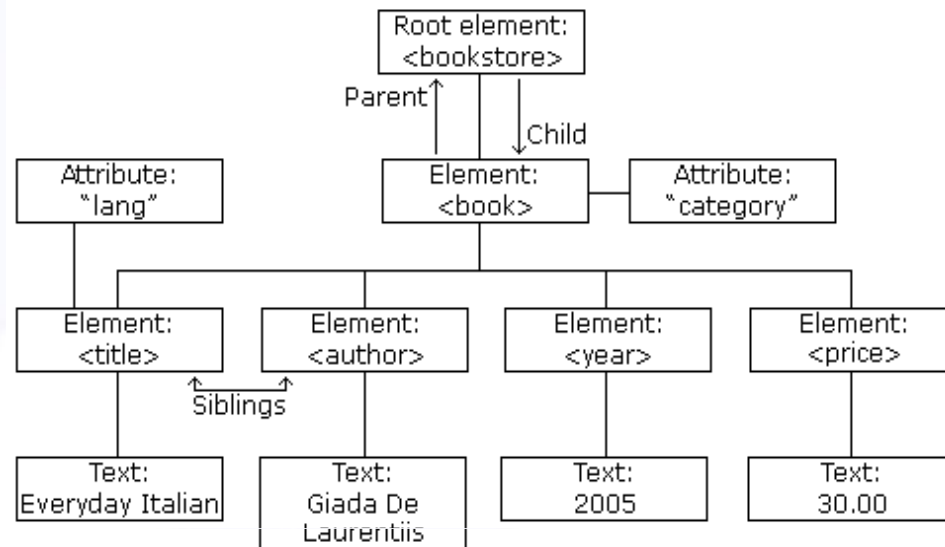
- <xml version> define la versión de XML y el encoding (carácteres)
- El tag <note> es el tag raíz (root), sólo puede haber uno en XML
- Los tags <to> <from> <heading> y <body> son elementos hijos (children) del raíz (root)
- Por lo tanto, el XML tiene forma de árbol lógico



Introducción a XML: ejemplo XML

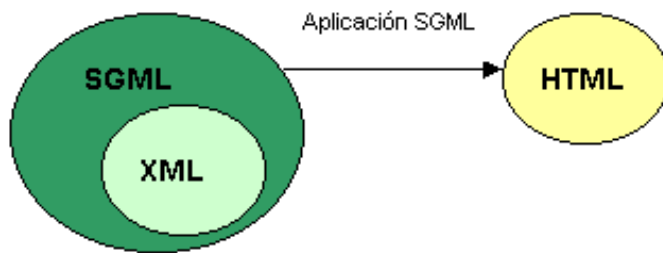
- Los tags a su vez pueden tener atributos opcionales

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```



Introducción a XML: diferencias XML y HTML

- XML no es un sustituto de HTML: se diseñaron con distintos objetivos
- XML se diseñó para el envío, transporte y almacenamiento de la información, enfocándose en la estructura de los datos.
- HTML se diseñó para mostrar la información, presentarla y formatearla, enfocándose en la apariencia de los datos
- XML no hace nada, solo estructura la información
- Con XML se pueden definir tags propios, con HTML ya están preestablecidos.
- HTML y XML son complementarios



- Si se quiere presentar información dinámica (por ejemplo en web con HTML) es muy costoso editar el HTML cada vez que el texto cambie.
- Con XML, los datos se pueden almacenar por un lado y el HTML para definir el layout (cómo se vera) visual.
- Existen muchas bases de datos y sistemas con formatos incompatibles, pero se pueden entender usando XML
- Se puede hacer disponible la información a distintos tipos de aplicaciones
- Se han creado muchos lenguajes (o metalenguajes) basados en XML:
 - XHTML
 - WSDL
 - WAP y WML
 - RSS

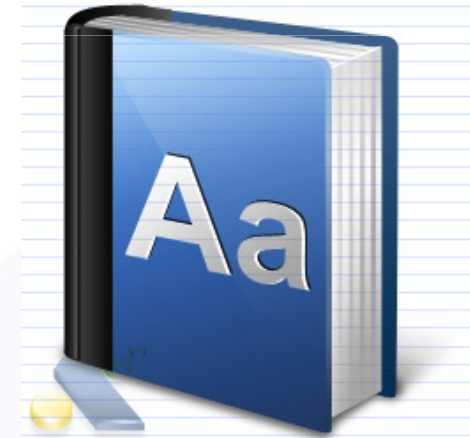


Indice

- 👉 Introducción XML
- 👉 *Validación XML*
- 👉 Parseadores JAVA
- 👉 DOM
- 👉 JDOM
- 👉 SAX



- *Existen una serie de reglas para los documentos XML:*
 - Todos los elementos deben tener un tag de apertura y cierre:
 - `<p>This is a paragraph</p>`
 - Los tags son sensibles a mayúsculas:
 - `<Message>This is incorrect</message>`
 - `<message>This is correct</message>`
 - Los tags se deben anidar correctamente:
 - `<i> Esto es incorrecto </i>`
 - `<i> Esto es correcto! </i>`
 - Los XML deben tener un único elemento raíz (root)
 - Los valores de atributos deben ir entre comillas:
 - `<note date="12/11/2007">...</note>`



➤ Otras reglas de la sintaxis:

- Los caracteres especiales se deben codificar para evitar confusión:
 - `<message>if salary < 1000 then</message>`
 - `<message>if salary < 1000 then</message>`
- Comentarios en XML:
 - `<!-- This is a comment -->`
- Restricciones en los nombres:
 - No pueden empezar con punto o número, ni contener espacio
 - No pueden empezar con las letras xml

" "	& &	< <	> >	← ←	↑ ↑	→ →
↔ ⇔	◇ ◊	♠ ♠	♣ ♣	♥ ♥	♦ ♦	∇ ∀
∏ ∏	∑ ∑	− −	* ∗	√ √	∞ ∝	∞ ∞
~ ∼	≅ ≅	≈ ≈	≠ &neq;	≡ ≡	≤ ≤	≥ ≥
⊥ ⊥	· ⋅	 	! !	¢ ¢	£ £	¤ ¤
¬ ¬	– ­	® ®	— ¯	° °	± ±	² ²



➤ Validación:

- XML con una sintaxis correcta es un XML “bien formado” (well-formed)
- XML validado contra un DTD es un XML “válido” (valid-xml). Alternativa al DTD es XML-schema. Necesario un validador. Ejemplo de validador: http://www.w3schools.com/xml/xml_validator.asp
- Para ser XML bien formado basta con cumplir las reglas acabas de comentar (sintácticas)
- El DTD (Document Type Definition) define la estructura que debe tener un XML concreto (atributos, obligatorios, opcionales, etc):

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE note SYSTEM "Note.dtd"> ...
```

```
<!DOCTYPE note  
[  
  <!ELEMENT note (to,from,heading,body)>  
  <!ELEMENT to (#PCDATA)>  
  <!ELEMENT from (#PCDATA)>  
  <!ELEMENT heading (#PCDATA)>  
  <!ELEMENT body (#PCDATA)>  
]>
```



➤ *Validación:*

- XSLT (Extensible Stylesheet Language Transformations) es la hoja de estilos recomendada para XML
- Más sofisticado y potente que los CSS.
- Se utilizan para transformar XML en HTML antes de ser mostrado en el navegador
- Ejemplo:
 - <http://www.w3schools.com/xml/tryxslt.asp?xmlfile=simple&xsltfile=simple>
- Estas transformaciones se realizan en el servidor



Índice

- ❖ Introducción XML
- ❖ Validación XML
- ❖ *Parseadores JAVA*
- ❖ DOM
- ❖ JDOM
- ❖ SAX



Indice

- 👉 Introducción XML
- 👉 Validación XML
- 👉 *Parseadores JAVA*
- 👉 DOM
- 👉 JDOM
- 👉 SAX



➤ *Parsing:*

- Se necesita algún mecanismo para procesar y leer ficheros XML a nivel de aplicación.
- Estos mecanismos se denominan parsers, y ofrecen distintas maneras de realizar el procesado.
- En JAVA, existen básicamente tres modelos para realizar el parsing con sus fabricantes:

➤ DOM

- Sun JAXP
- IBM XML4J
- Apache Xerces
- Resin (Caucho)
- DXP (DataChannel)

➤ SAX

- Sun JAXP
- SAXON

➤ JDOM



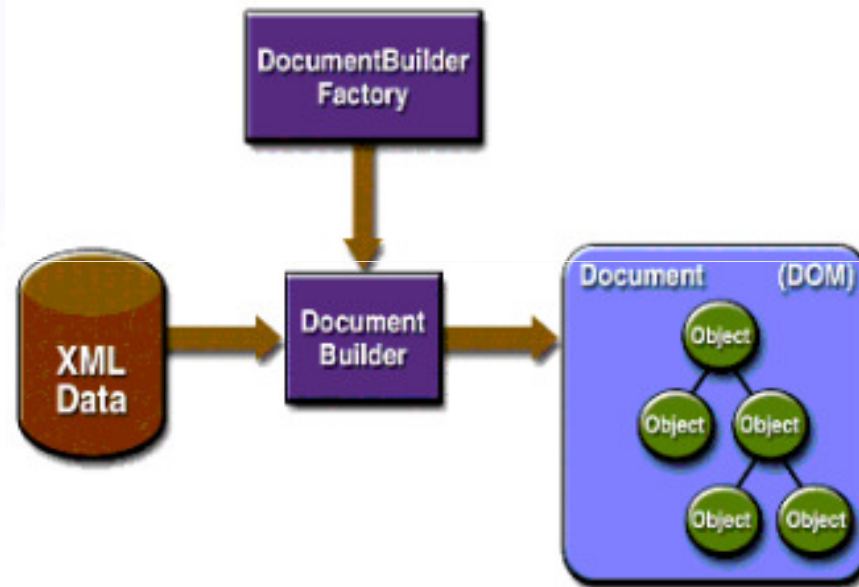
Indice

- 👉 Introducción XML
- 👉 Validación XML
- 👉 Parseadores JAVA
- 👉 *DOM*
- 👉 JDOM
- 👉 SAX



➤ *API DOM:*

- Basada en la estructura de árbol del XML
- La API gira entorno a la clase Node
- Permite ordenamiento y modificación de los elementos del XML
- Facilita el compartir el documento con otras aplicaciones
- Basado en interfaces: Document, Text, Element...



Introducción a XML: ejemplo de uso de la API DOM

```
public void print(Node node) { //recursive method call using DOM API...
    int type = node.getNodeType();
    case Node.ELEMENT_NODE: // print element with attributes
        out.print('<');
        out.print(node.getNodeName());
        Attr attrs[] = node.getAttributes();
        for (int i = 0; i < attrs.length; i++) {
            Attr attr = attrs[i];
            out.print(' '); out.print(attr.getNodeName()); out.print("=\");
            out.print(normalize(attr.getNodeValue())); out.print('"');
        }
        out.print('>');
        NodeList children = node.getChildNodes();
        if (children != null) {
            int len = children.getLength();
            for (int i = 0; i < len; i++) {
                print(children.item(i));
            }
        }
        break;

    case Node.ENTITY_REFERENCE_NODE: // handle entity reference nodes

// ...
```



- Node
 - getNodeName()
 - getNodeValue()
 - getAttributes()
 - getChildNodes()
 - getParentNode()
- Attr
 - attributes are not technically child nodes
 - getParent()
 - getName(), getValue()
- Document
 - ...
- Element
 - getTagName()
 - getElementsByTagName(String tag)
 - normalize()



Indice

- 👉 Introducción XML
- 👉 Validación XML
- 👉 Parseadores JAVA
- 👉 DOM
- 👉 *JDOM*
- 👉 SAX



➤ *DOM vs JDOM:*

- DOM está escrito en C
- API DOM no es del todo eficiente, optimizable en JAVA
- En DOM es tedioso recorrer partes del árbol XML
- DOM no soporta por defecto alguno estándares JAVA (java.util.Collections)
- JDOM es DOM mejorado para JAVA
- JDOM es Open Source
- Limpia, simple y directa API
- Aprovecha los beneficios de las librerías JAVA
- Mejora el recorrido por los árboles: métodos getNextSibling()...getChildText()... evita recorridos pesados



```
XMLOutputter out = new XMLOutputter();  
out.output( element, System.out );
```

También se puede hacer con:

```
public void print(Element node) { //recursive method call using JDOM  
API...  
out.print('<');  
    out.print(node.getName());  
    List attrs = node.getAttributes();  
    for (int i = 0; i < attrs.size(); i++) {  
        Attribute attr = (Attribute)attrs.get(i);  
        out.print(' '); out.print(attr.getName()); out.print("=\");  
        out.print(attr.getValue() ); out.print('"');  
    }  
    out.print('>');  
    List children = node.getChildren();  
    if (children != null) {  
        for (int i = 0; i < children.size(); i++) {  
            print(children.item(i));  
        }  
    }  
}
```



➤ *Ejemplo del workspace:*

- En la proyecto jdom-test se pueden ver dos ejemplos de cómo leer (LecturaXML.java) y escribir (EscrituraXML.java) utilizando la API JDOM.

El ejemplo de lectura recoge el fichero liga.xml y lo vuelca por pantalla.

El ejemplo de escritura genera un fichero ejemplo.xml con una serie de elementos especificados en la aplicación.



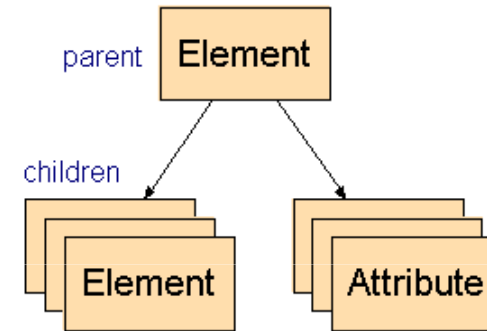
Indice

- 👉 Introducción XML
- 👉 Validación XML
- 👉 Parseadores JAVA
- 👉 DOM
- 👉 JDOM
- 👉 **SAX**



➤ SAX:

- Siglas de Simple API for XML
- Permite acceso secuencial al XML.
- Requiere menos recursos que DOM
- Enfocado a eventos: cuando el parser SAX encuentra un nodo de un tipo, invoca un método de la aplicación.
- El programador crea una serie de métodos de callback en una clase (hereda de DefaultHandler) que se llamarán cuando el parser encuentre elementos en el XML
- El parser convierte la información del XML en elementos de distintos tipos (Element, TextNode...) y invoca los métodos.
- Ejemplos de métodos: `startDocument()`; `endDocument()`; `startElement(..)`;



➤ *Ejemplo del workspace:*

- En la proyecto sax-test se puede ver un ejemplo de cómo realizar una lectura de un fichero XML (libreria.xml).

Lo más importante a ver es cómo se implementa el DefaultHandler (clase Manejador) y el método principal.

