XML en Java

Clases involucradas

- java.xml.parsers: Las APIs de JAXP (java API for XML Processing) que proporcionan una interfaz común para los distintos parsers de SAX y DOM
- org.w3c.dom: Define la “clase” Document (un DOM) y las clases componentes del DOM
- org.xml.sax: Define las APIs básicas del SAX
- java.xml.transform: Define las APIs XSLT para transformar XML a otros formatos.
- java.xml.stream: Proporciona APIs de transformación específica StAX

SAX (Simple API for XML)

La API simple para XML (SAX) es el mecanismo orientado a eventos y de acceso serie que hace el procesado elemento a elemento.

La API de este nivel lee y escribe XML a un repositorio de datos o la web. Para la mayoría de las aplicaciones basta con una comprensión mínima de la misma.

DOM (Document Object Model)

La API DOM es generalmente más fácil de usar. Proporciona una estructura de árbol de los objetos. Se puede utilizar DOM para manipular la jerarquía de los objetos que encapsula.

Es ideal para aplicaciones interactivas, ya que el modelo entero de objetos se presenta en memoria, donde puede ser accedido y manipulado por el usuario.

Por otra parte , la construcción del DOM requiere leer la estructura XML entera y mantener el árbol de objetos en memoria, con consumo intensivo de CPU y memoria. Por eso SAX es preferido para aplicaciones del lado de servidor o aplicaciones con buffer que no requieren una representación en memoria de los datos

XSLT (Extensible Stylesheet Language Trasformations)

La API XSLT definida en javax.xml.transform te permite escribir los datos XML a un archivo o convertirlo a otros formatos.

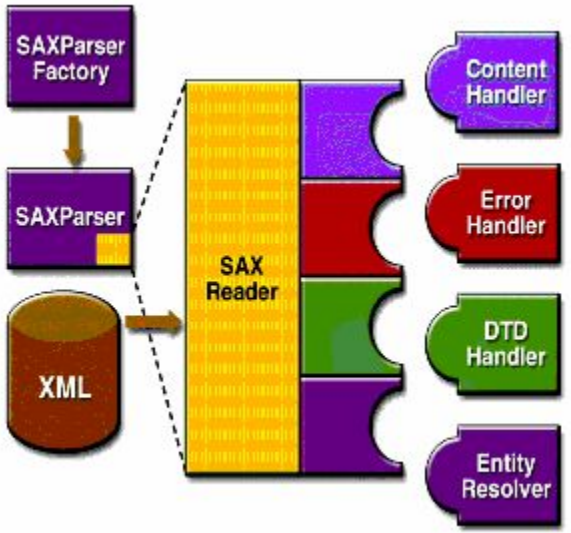
Incluso se puede utilizar conjuntamente con SAX para convertir datos legacy a XML (formatos obsoletos)

StAX (Streaming API for XML)

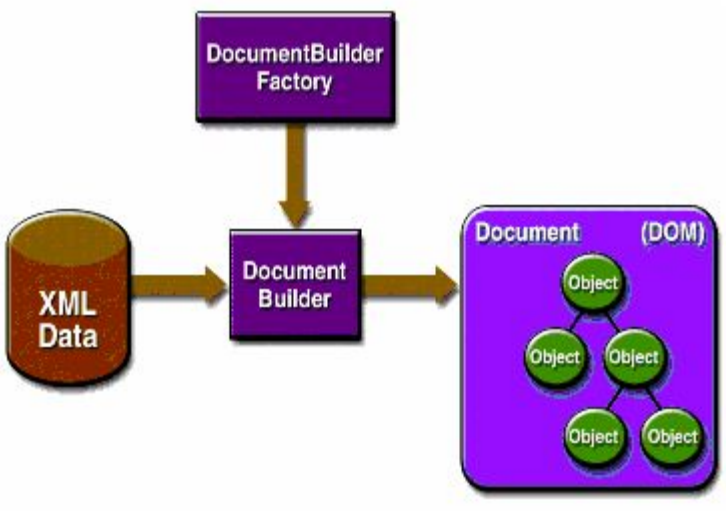
Las APIs definidas en javax.xml.stream proporcionan la tecnología de streaming de java para leer y escribir documentos XML.

StAX ofrece un modelo de programación más simple que SAX y más eficiente en la gestión de memoria que DOM

Uso de SAX


SAXParser Factory
SAXParser
XML
SAX Reader
Content Handler
Error Handler
DTD Handler
Entity Resolver

Uso de DOM


DocumentBuilder Factory
XML Data
Document Builder
Document (DOM)
Object
Object
Object
Object
Object

Paquetes importantes javax.xml

javax.xml.parsers
Provides the classes for processing XML documents with a SAX (Simple API for XML) parser or a DOM (Document Object Model) Document builder.

javax.xml.transform
Defines the generic APIs for processing transformation instructions, and performing a transformation from source to result.

javax.xml.transform.dom
Provides DOM specific transformation classes.

javax.xml.transform.sax
Provides SAX specific transformation classes.

org.w3c.dom
Provides the interfaces for the Document Object Model (DOM).

org.xml.sax
Provides the interfaces for the Simple API for XML (SAX).

Servicios en javax.xml

DocumentBuilderFactory

Define una API “fábrica” que permite a las aplicaciones obtener un parser que produce árboles de objeto DOM a partir de documentos XML.

SAXParserFactory

Define una APi “fábrica” que permite a las aplicaciones configurar y obtener un parser basado en SAX para interpretar documentos XML.

TransformerFactory

Una instancia de TransformerFactory puede usarse para crear objetos `Transformer` y `Templates`.

Paquete javax.xml.parsers

Clases

DocumentBuilder

Define la API para obtener instancias Document DOM desde un documento XML.

DocumentBuilderFactory

Define una API “fábrica” que permite a las aplicaciones obtenr un parser que produce árboles DOM a partir de documentos XML.

SAXParser

Define la API que contiene una implementación de la clase `XMLReader`.

SAXParserFactory

Define una API “fábrica” que permite a las aplicaciones obtener y configurar un parser basado en SAX para interpretar documentos XML.

Excepciones

ParserConfigurationException

Indica un error grave de configuración.

DocumentBuilderFactory

Utilizamos

`static` **`DocumentBuilderFactory`** **`newInstance`**`()` Obtener una nueva instancia de `DocumentBuilderFactory`.
A continuación utilizamos esta instancia para construir un DocumentBuilder mediante el método de DocumentBuilderFactory

`abstract` **`DocumentBuilder`** **`newDocumentBuilder`**`()` Crea una nueva instancia de `DocumentBuilder` con los parámetros actualmente configurados.

DocumentBuilder

A la instancia de DocumentBuilder le pasamos un parser a través del método sobrecargado parse()

`Document` **`parse`**`(File f)`
Parse the content of the given file as an XML document and return a new DOM `Document` object.

`Document` **`parse`**`(InputStream is)`
Parse the content of the given `InputStream` as an XML document and return a new DOM `Document` object.

`Document` **`parse`**`(InputStream is, String systemId)`
Parse the content of the given `InputStream` as an XML document and return a new DOM `Document` object.

`Document` **`parse`**`(String uri)`
Parse the content of the given URI as an XML document and return a new DOM `Document` object.

Nos devuelve un objeto de tipo Document (DOM)

Document

La interfaz Document representa el documento entero XML. Conceptualmente es la raíz del árbol del documento, y proporciona el acceso inicial a los datos del documento.

Los elementos, nodos de texto, comentarios, instrucciones de procesamiento, etc, no pueden existir fuera del contexto de un Document, y la interface Document contiene los métodos que permiten crear esos objetos. Los objetos Node tienen un atributo ownerDocument que los asocia al documento en cuyo contexto han sido creados

Métodos de la interfaz Document, creación de nodos

`Attr createAttribute(String name)` Creates an `Attr` of the given name.

`Comment createComment(String data)` Creates a `Comment` node given the specified string.

`Element createElement(String tagName)` Creates an element of the type specified.

`Text createTextNode(String data)` Creates a `Text` node given the specified string.

Métodos de la interfaz Document, obtención de nodos

`Element getDocumentElement()` This is a convenience attribute that allows direct access to the child node that is the document element of the document.

`NodeList getElementsByTagName(String tagname)` Returns a `NodeList` of all the `Elements` in document order with a given tag name and are contained in the document.

La interfaz Node

Es el tipo de datos principal para el Document. Representa un único nodo en el árbol del documento.

Todos los objetos que implementan Node tienen métodos para tratar con hijos, aunque no todos tengan hijos.

Por ejemplo, los Node tipo Text no tienen hijos, por lo que intentar añadir hijos a estos nodos generará el lanzamiento de una DOMException

Los atributos nodeName, nodeValue, y attributes están incluidos como un mecanismo para obtener la información del nodo sin tener que hacer el casteo a la interfaz derivada específica. En casos no obvios devolverá null, por lo que habría que tenerlo en cuenta.

Métodos de Node

`Node appendChild(Node newChild)` Adds the node `newChild` to the end of the list of children of this node.

`NamedNodeMap getAttributes()` A `NamedNodeMap` containing the attributes of this node (if it is an `Element`) or `null` otherwise.

`NodeList getChildNodes()` A `NodeList` that contains all children of this node.

`Node getFirstChild()` The first child of this node.

`Node getLastChild()` The last child of this node.

`String getLocalName()` Returns the local part of the qualified name of this node.

`String getNodeName()` The name of this node, depending on its type; see the table above.

`short getNodeType()` A code representing the type of the underlying object, as defined above.

`String getNodeValue()` The value of this node, depending on its type; see the table above.

`Node getParentNode()` The parent of this node.

`String getTextContent()` This attribute returns the text content of this node and its descendants.

`boolean hasAttributes()` Returns whether this node (if it is an element) has any attributes.

`boolean hasChildNodes()` Returns whether this node has any children.

`Node insertBefore(Node newChild, Node refChild)` Inserts the node `newChild` before the existing child node `refChild`.

`Node removeChild(Node oldChild)` Removes the child node indicated by `oldChild` from the list of children, and returns it.

`Node replaceChild(Node newChild, Node oldChild)` Replaces the child node `oldChild` with `newChild` in the list of children, and returns the `oldChild` node.

`void setNodeValue(String nodeValue)` The value of this node, depending on its type; see the table above.

`void setTextContent(String textContent)` This attribute returns the text content of this node and its descendants.

Valores para los diferentes nodos

Interface	nodeName	nodeValue	attributes
Attr	same as Attr.name	same as Attr.value	null
CDATASection	"#cdata-section"	same as CharacterData.data, the content of the CDATA Section	null
Comment	"#comment"	same as CharacterData.data, the content of the comment	null
Document	"#document"	null	null
DocumentFragment	"#document-fragment"	null	null
DocumentType	same as DocumentType.name	null	null
Element	same as Element.tagName	null	NamedNodeMap
Entity	entity name	null	null
EntityReference	name of entity referenced	null	null
Notation	notation name	null	null
ProcessingInstruction	same as ProcessingInstruction.target	same as ProcessingInstruction.data	null
Text	"#text"	same as CharacterData.data, the content of the text node	null

Sumario de campos

`static short` **`ATTRIBUTE_NODE`** The node is an `Attr`.
`static short` **`ELEMENT_NODE`** The node is an `Element`.
`static short` **`TEXT_NODE`** The node is a `Text` node.

Obtener una lista de nodos y procesarla

```
NodeList nl=doc.getDocumentElement().getChildNodes();

for (int i=0; i<nl.getLength();i++) {

Node n=nl.item(i);

switch (n.getNodeType()){

case Node.ELEMENT_NODE: Element e=(Element)n;

System.out.println("Etiqueta:" + e.getTagName());

break;

case Node.TEXT_NODE: Text t=(Text)n;

System.out.println("Texto:" + t.getWholeText());

break;

}

}
```

XML DOM

De acuerdo con el DOM, cada cosa que hay en un documento XML es un nodo

- El documento entero es un nodo documento
- Todos los elementos XML son un nodo elemento
- El texto en los elementos XML son nodo texto
- Cada atributo es un nodo atributo
- Los comentarios son nodo comentario

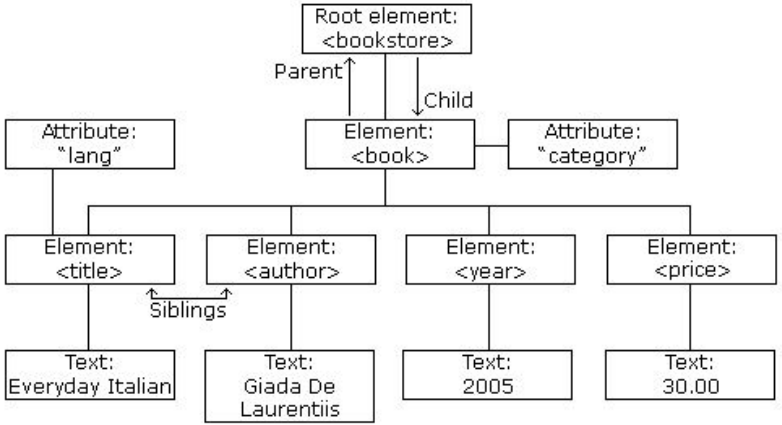
Recordar: Los textos están siempre almacenados en nodos texto.

Un elemento contiene un nodo texto, no es que el texto sea el valor del nodo

El árbol de nodos DOM de XML

Todos los nodos se pueden acceder a través del árbol. Sus contenidos pueden ser modificados o eliminados, y se pueden crear nuevos elementos.

El árbol muestra el conjunto de nodos y las conexiones entre ellos. Empieza en el nodo raíz y se va ramificando hasta los nodos de texto en el nivel más bajo del árbol.



Acceso a nodos

Se puede acceder un nodo de tres maneras:

1. Utilizando el método getElementsByTagName(“nombreDeLaEtiqueta”)
2. Haciendo un recorrido por todos los nodos del árbol.
3. Navegando por el árbol de nodos utilizando las relaciones entre nodos (padres, hijos…)

Ej:

x.getElementsByTagName(“titulo”);

Devuelve todos los elementos <titulo> bajo el elemento x

Interface NodeList

Proporciona la abstracción de una colección ordenada de nodos.

Los elementos de una NodeList son accesibles a través de su índice que comienza en 0.

Métodos

`int` **`getLength`**`()`	El número de nodos de la lista.
`Node` **`item`**`(int indice)`	Devuelve el elemento de la posición indice de la lista.

Tipos de nodo

La propiedad documentElement del documento XML es el nodo root

La propiedad nodeName de un nodo es el nombre del nodo

La propiedad nodeType de un nodo es el tipo de nodo que es

propiedad nodeName

Especifica el nombre del nodo

- Es de solo lectura
- El nodeName de un element node es el mismo que el nombre de la etiqueta
- El nodeName de un attribute node es el nombre del atributo
- El nodeName de un text node es siempre #text
- El nodeName de un document node es siempre #document

La propiedad nodeValue

Especifica el valor de un nodo

- El nodeValue para element node no está definido
- El nodeValue para text node es el propio texto
- El nodeValue para attribute node es el valor del atributo