

# JAVA+XML = JDOM (Parte 1)

## Introducción

JDOM es un API para leer, crear y manipular documentos XML de una manera sencilla y muy intuitiva para cualquier programador en Java, en contra de otras APIs tales como DOM y SAX, las cuales se idearon sin pensar en ningún lenguaje en concreto, de ahí que resulte un poco incomodo su utilización.

En este artículo exploraremos un poco el API, crearemos una pequeña aplicación que lea un documento XML y sacaremos de él aquellas partes que nos interese. Para abrir boca, imagínate este trozo de XML:

```
...
<site>javahispano</site>
...
```

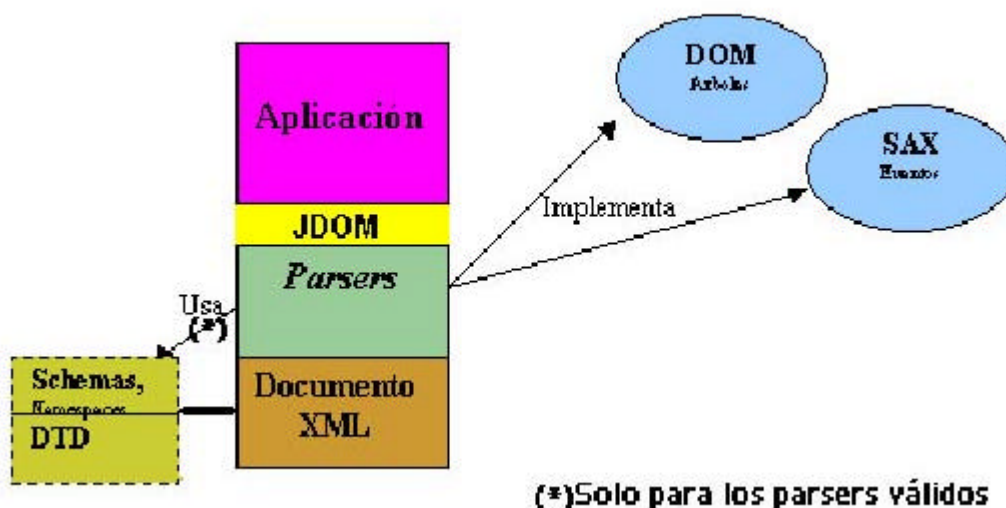
Se imagina acceder al texto del elemento site de esta manera:

```
String nombre = site.getText();
```

¿Te gusta? Esto es JDOM.

## Donde se encuentra JDOM

Para comenzar haremos una visión de pájaro para ver donde encajamos JDOM en un pequeño esquema.



Antes de seguir adelante permíteme una aclaración: DOM y SAX son dos especificaciones que como tal no podemos trabajar con ellas, pero si con las implementaciones de dichas especificaciones, es decir, los parsers: Xerces, XML4j, Crimson, Oracle's parsers,...

Pues bien un ultimo apunte, **la API JDOM no es un parser**, de echo, usa un parser para su trabajo, JDOM "solo" nos aporta una capa de abstracción en el tratado de documentos XML facilitándonos bastante la tarea como veremos enseguida, de echo no tendremos que ser unos gurus de DOM y SAX para poder trabajar con XML desde Java.

## Donde conseguir JDOM

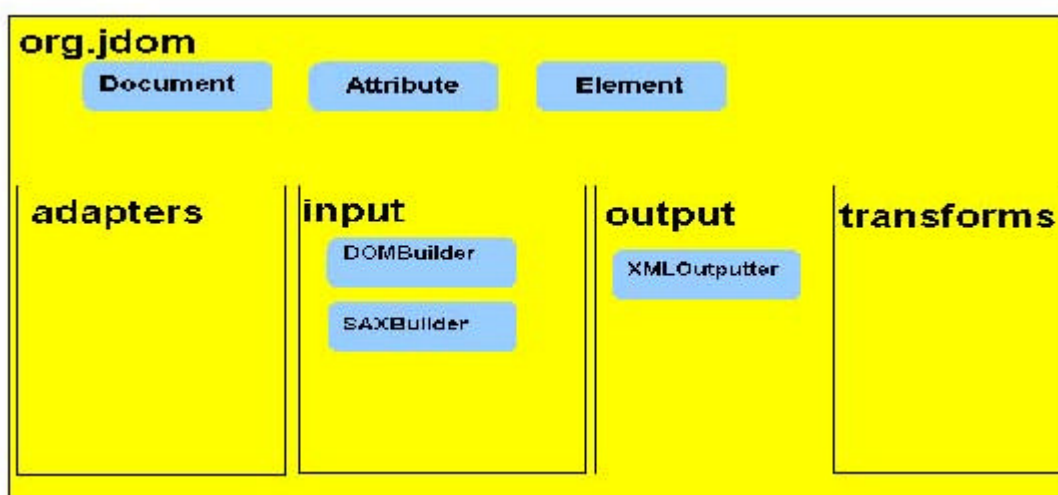
En [www.jdom.org](http://www.jdom.org) podremos descargar tanto el API como la documentación que necesitemos. Antes de que se me olvide, la licencia es open source. :-)

Como JDOM fue diseñado usando List y Map de la API Java 2 Collections, tendremos que utilizar el jdk 1.2 o si usas el jdk 1.1 deberás de instalar la librería Collections.

## Estructura de JDOM

El API está formado por 5 packages. De entre ellas comentamos lo siguiente que será mas que suficiente para utilizar el API.

- El package org.jdom destacamos las clases: Document que representará el documento XML, Element que representará el elemento o etiqueta que forma el documento, y la clase Attribute que como bien imaginaras representa los atributos que puedan tener los elementos.
- El package org.jdom.adapters albergará todas las clases adaptadoras (ver patrón de diseño Adapter, Thinking in patterns) ya que no todos los parsers DOM tienen la misma API. Mas tarde quedara mas claro su función.
- El package org.jdom.input albergara las clases builder para construir los documentos XML.
- El package org.jdom.output albergara las clases que utilizaremos para dar salida a nuestra clase Document.



## Un poco de teoría

Como dije al principio del artículo JDOM usaba los parsers para hacer su trabajo, pues bien, para decirle a JDOM que parser utilizar utilizaremos uno de los siguientes métodos:

```
public SAXBuilder(String parserClass, boolean validation)
```

El primer parámetro es el parser que vamos a utilizar, por defecto se utilizará el parser Xerces.

El segundo parámetro es para decirle si queremos que el parser cumpla sus obligaciones de validación.

```
public DOMBuilder(String adapterClass, boolean validation)
```

El primer parámetro es la clase adaptadora que vamos a utilizar para el parser que utilizaremos .

El segundo parámetro es igual que el del SAXBuilder.

Ahora al builder le daremos la orden de parsear el documento XML con el método build(), cuya forma es:

```
Document build(File file)
```

Muy bien ya tenemos el documento almacenado en la clase Document. Finalmente vamos a aprender unos cuantos métodos mas para recuperar la información que deseemos:

Este método pertenece a la clase Document

```
Element getRootElement(); //cojer el nodo raiz del documento.
```

Estos métodos pertenecen a la clase Element:

```
String getText();
//Capturar el texto de una etiqueta o elemento.
List getChildren();
//Coger todos los elementos que cuelgan del Element.
List getChildren(String nombre);
//Coger todos los elementos que tengan ese nombre
List getMixedContent();
//Para recuperar todo(comentarios, PIs, elementos,...)
// de lo que cuelga del Element.
Element getChild (String nombre);
//Coger el primer hijo que tenga ese nombre.
String getAttributeValue(String nombre);
//Coger el valor del atributo que pasamos como parámetro.
Attribute getAttribute(String nombre);
//Coger el atributo que tenga ese nombre y para recuperar
//el valor de ese atributo se
//utilizaría el método del attribute: String getValue();
```

## Nuestro primer programa

Dado el siguiente documento XML:

```
<?xml version="1.0"?>
<liga tipo="Champions League">
  <equipo>
    <club valoracion="10" ciudad="Bilbao">Athletic Club Bilbao</club>
    <presidente>Uria</presidente>
```

```
<plantilla>
  <nombre>Julen Guerrero</nombre>
  <nombre>Joseba Etxeberria</nombre>
  <nombre>Ismael Urzaiz</nombre>
</plantilla>
</equipo>
<equipo>
  <!-- no os piqueis ;-) -->
  <club valoracion="5" ciudad="Madrid">Real Madrid</club>
  <presidente>Mandamas</presidente>
  <plantilla>
    <!-- no pongo nombres propios por si acaso -->
    <nombre>Bota de oro</nombre>
    <nombre>Milloneti</nombre>
    <nombre>Canterano quemado</nombre>
  </plantilla>
  <conextranjeros/>
</equipo>
<arbitros>
  <nombre>No doy una</nombre>
  <nombre>Rafanomejodas</nombre>
</arbitros>
</liga>
```

Vamos a hacer un programa que lo lea y nos muestra cierta información. Este sería el código:

```
import java.io.*;
import java.util.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;

public class Ejemplo {

    public static void main(String[] args) {
        try {
            SAXBuilder builder=new SAXBuilder(false);
            //usar el parser Xerces y no queremos
            //que valide el documento
            Document doc=builder.build("liga.xml");
            //construyo el arbol en memoria desde el fichero
            // que se lo pasaré por parametro.
            Element raiz=doc.getRootElement();
            //cojo el elemento raiz
            System.out.println("La liga es de tipo:"+
                raiz.getAttributeValue("tipo"));
            //todos los hijos que tengan como nombre plantilla
            List equipos=raiz.getChildren("equipo");
            System.out.println("Formada por:"+equipos.size()+" equipos");
            Iterator i = equipos.iterator();
            while (i.hasNext()){
                Element e= (Element)i.next();
                //primer hijo que tenga como nombre club
                Element club =e.getChild("club");
                List plantilla=e.getChildren("plantilla");
                System.out.println
                    (club.getText()+":"+valoracion="+
                    club.getAttributeValue("valoracion")+","+
                    "ciudad="+club.getAttributeValue("ciudad")+","+
                    "formada por:"+plantilla.size()+" jugadores");
                if (e.getChildren("conextranjeros").size()==0)
                    System.out.println("No tiene extranjeros");
            }
        }
    }
}
```

```
        else System.out.println("Tiene extranjeros");
    }
    // Dejamos de mano del lector el sacar el nombre
    //de los arbitros, animate!!
}catch (Exception e){
    e.printStackTrace();
}
}
```

## Conclusión

Como hemos visto el leer documentos XML y sacar la información de dicho documento es un trabajo realmente fácil con JDOM, como veremos en el siguiente artículo el crear documentos con esta API no es mucho más complicado.

**Javier Teso**, actualmente está cursando 5º de Ing. Informática de San Sebastián.  
Para cualquier duda o comentario: [jte\\_job@hotmail.com](mailto:jte_job@hotmail.com) ó [jabotxa@latinmail.com](mailto:jabotxa@latinmail.com)