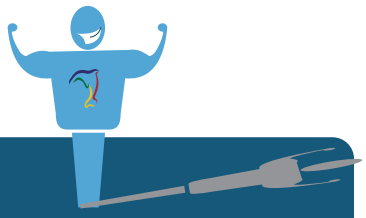
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

# ¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

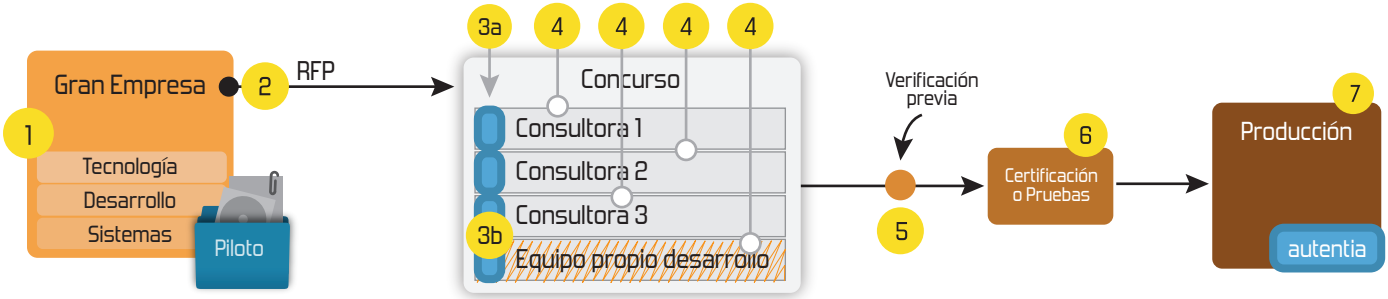
## 1. Desarrollo de componentes y proyectos a medida



## 2. Auditoría de código y recomendaciones de mejora

## 3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.




## 4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
**www.adictosaltrabajo.com**

Para más información visítenos en:
**www.autentia.com**

**Tutorial desarrollado por: Francisco Javier Martínez Páez**

**Puedes encontrarme en Autentia**
**Somos expertos en Java/J2EE**
**Contacta en info@autentia.com**



Descargar este documento en formato PDF XPATH.pdf



**INTRODUCCIÓN A XPATH**

Los ejemplos de este tutorial están hechos con el siguiente entorno de desarrollo:

- Jboss Eclipse IDE Milestone 5.
- JDK 1.5

Hemos visto a lo largo de estos últimos años como XML se ha ido imponiendo poco a poco, como un lenguaje válido para intercambio de datos, debido entre otras cosas a su sencillez y flexibilidad. También hemos visto aparecer gran cantidad de estándares, herramientas, interfaces, librerías, etc..., todas ellas relacionadas con éste metalenguaje. Os propongo en este tutorial dar a conocer un poco, otro estándar surgido alrededor de todo este mogollón y una librería que lo implementa.

XPath no es más que un lenguaje que nos permite buscar información dentro de un XML, navegar entre etiquetas y atributos de una manera relativamente sencilla, etc... , en definitiva, algo que nos hará la vida más fácil a la hora de tratar con documentos XML. XPath es uno de los elementos principales del estándar XSLT del W3C: http://www.w3.org/TR/xpath

XPath incluye:

- Una sintáxis para definir las partes de un documento XML.
- Un conjunto de expresiones para seleccionar partes de un documento XML.
- Un conjunto de funciones estándar para tratamiento de cadenas, fechas, valores numéricos, etc....

Alguno quizá se esté preguntando todavía: ¿Pero para que sirve esto?

Lo mejor, es un ejemplo:

Supongamos que tenemos un documento XML de este tipo:

```
<libros>
<libro>
   <autor>Soren Kierkegaard</autor>
   <titulo>Fear and Trembling</titulo>
   <precio>50.0</precio>
</libro>
<libro>
   <autor>Miguel de Unamuno</autor>
   <titulo>Niebla</titulo>
   <precio>40.0</precio>
</libro>
<libro>
   <autor>Miguel de Unamuno</autor>
   <titulo>La tía tula</titulo>
   <precio>23.0</precio>
</libro>
<libro>
   <autor>George Orwell</autor>
   <titulo>1984</titulo>
   <precio>20.0</precio>
</libro>
</libros>
```

Ahora, imaginad las siguientes preguntas:

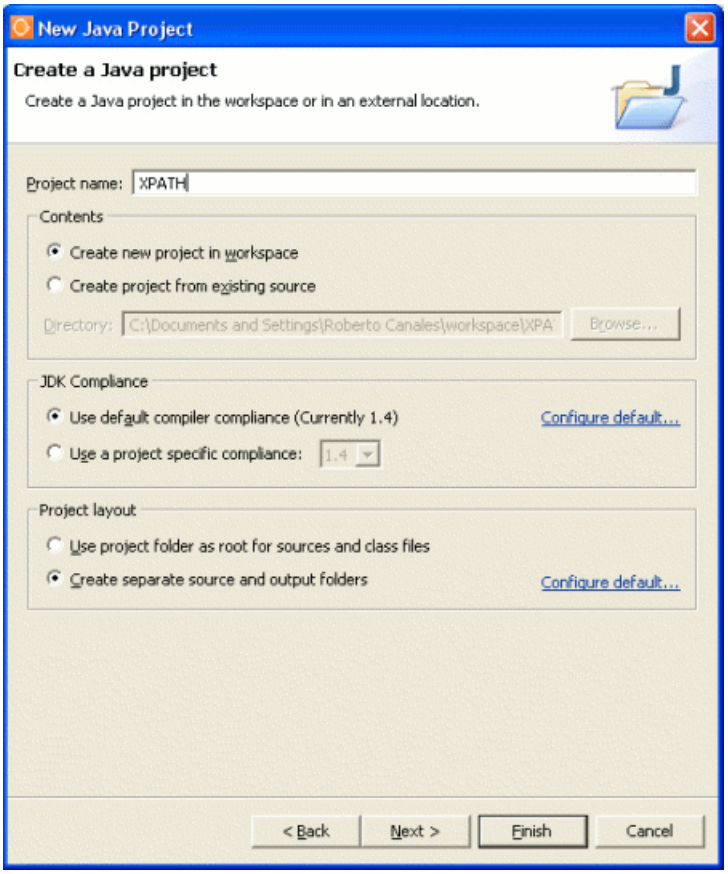
- ¿ Qué libros son de Miguel de Unamuno?
- ¿ Qué libros tienen un precio menor a 30 euros ?
- ¿ Hay algún libro que tenga como parte del título la palabra "Fear" ?

Seguro que se os ocurren muchas más, pero la pregunta es, ¿cómo lo hacemos? Probablemente, nos recorreríamos todo el documento mediante algun parser XML y buscaríamos cuales de ellos cumplen las condiciones. Si hiciesemos la analogía con una consulta SQL en una tabla de base de datos, hacer esto sería como hacer un SELECT * FROM LIBROS y posteriormente recorrer todos los registros para ver cuales son aquellos que cumplen las condiciones (poco eficiente me atrevería a decir). Pero, ¿no hay algo entonces que lo haga por nosotros? la respuesta es fácil: XPath.

**Nota**: *No debemos confundir no obstante XPath con XQuery, y no explico nada más porque lo dejaré para tutoriales posteriores, aunque ya os digo que XQuery es a XML lo que SQL a tablas de base de datos, y la confusión puede venir debido a que XQuery se apoya en XPath.*

**ENTRANDO EN MATERIA**

Vamos a preparar primero lo necesario para empezar. Primero nos crearemos un nuevo proyecto dentro de eclipse que llamaremos XPATH:




Después necesitaremos alguna librería java que nos implemente todo este mogollón, me voy a google y busco "xpath + java".

El primer resultado de la búsqueda nos remite al api de Java 5: javax.xml.xpath.
Pensaba haceros los ejemplos con la solución de apache (dentro de xalan) que ya he utilizado con éxito, pero probemos la otra (así aprendemos todos).

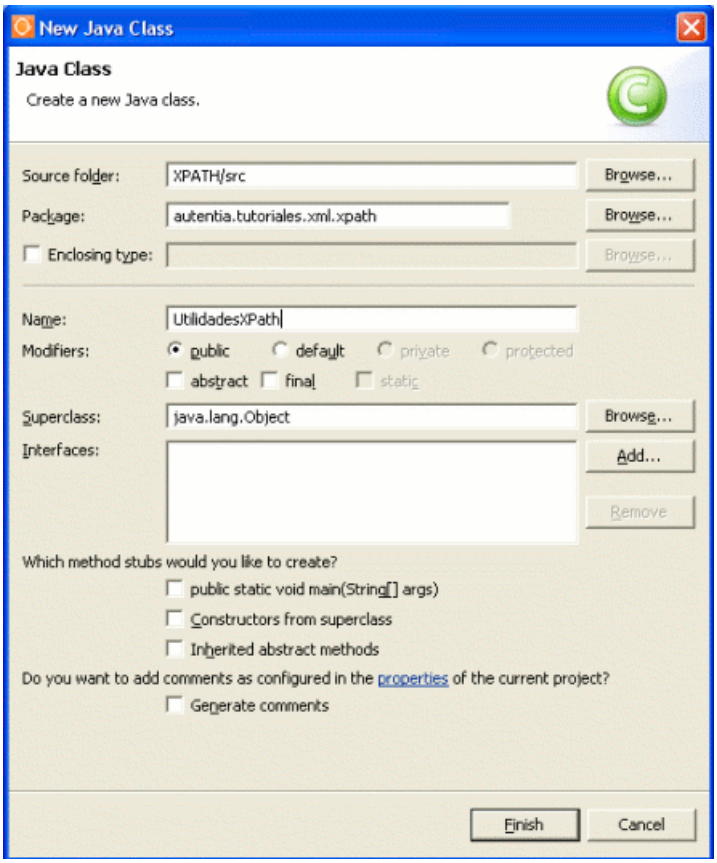
Vamos a escribir primero una clasecita de utilidades para facilitarnos el trabajo. La llamamos UtilidadesXML: (esta os la regalo)

```
package autentia.tutoriales.xml;

import java.io.StringWriter;
import java.net.URL;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

public class UtilidadesXML {

   /**
    * Este método busca un fichero de tipo XML en el classpath crea un objeto
    * de tipo org.w3c.dom.Document.
    * @param fichero: El nombre del fichero a procesar.
    * @return
    * @throws Exception
    */
   public static Document File2Document(String fichero) throws Exception {
     DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();         ClassLoader
loader=(UtilidadesXML.class).getClassLoader();
     URL urlfichero=loader.getResource(fichero);
     Document XMLDoc=factory.newDocumentBuilder().parse(new   InputSource
```

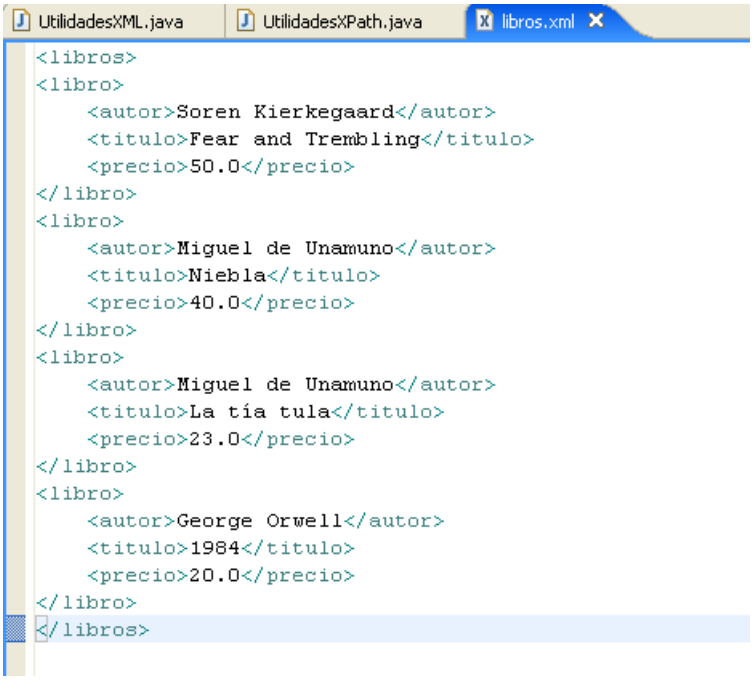
```
(urlfichero.openStream()));
        return XMLDoc;
    }

    /**
     * Este método convierte un objeto de tipo org.w3c.dom.Node a String
     * @param nodo
     * @return
     * @throws Exception
     */
    public static String Node2String (Node nodo) throws Exception {
        StringWriter sw = new StringWriter();
        StreamResult sR=new StreamResult(sw);
        Transformer transformer =      TransformerFactory.newInstance().newTransformer();
            transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            transformer.transform(new DOMSource(nodo),sR);
        return sw.toString();
    }

/**
     * Este método convierte un objeto de tipo org.w3c.dom.NodeList a String
     * Este método solo sirve para pintar los resultados.
     * @param nodo
     * @return
     * @throws Exception
     */
    public static String Nodes2String (NodeList nodes) throws Exception {
        StringBuffer lista = new StringBuffer();

        for(int i=0;i<nodes.getLength();i++) {
            Node node = (Node)nodes.item(i);
            lista.append(UtilidadesXML.Node2String(node));
        }

        return lista.toString();
    }

    /**
     * Devuelve true si el nodo es de tipo Node.ELEMENT_NODE y se llama "nombre"
     * @param nodo
     * @param nombre
     * @return
     */
    public static boolean esNodo(Node nodo, String nombre) {
        return (esNodo(nodo) && (nodo.getNodeName().equalsIgnoreCase(nombre)));
    }

    /**
     * Devuelve true si el nodo es de tipo Node.ELEMENT_NODE
     * @param nodo
     * @param nombre
     * @return
     */
    public static boolean esNodo(Node nodo) {
        return (nodo.getNodeType() == Node.ELEMENT_NODE);
    }

    /**
     * Devuelve true si el nodo es de tipo TEXT_NODE
     * @param nodo
     * @return
     */
    public static boolean esTexto(Node nodo) {
        return (nodo.getNodeType() == Node.TEXT_NODE);
    }

    /**
     * Devuelve el texto de un nodo: <tag>TEXTO</tag>
     * @param n
     * @return
     */
    public static String getTexto (Node n) {
        NodeList nl = n.getChildNodes();
        Node act=null;
        for (int i=0; i < nl.getLength(); i++) {
            act = nl.item(i);
            if (act == null) return null;
            if ((act.getNodeType() == Node.CDATA_SECTION_NODE)||(act.getNodeType() ==
Node.TEXT_NODE)) return act.getNodeValue();
        }
        return "";
      }

    /**
     * Devuelve el valor del atributo "nombre" de un nodo
     * @param nombre
     * @param nodo
     * @return
     */
    public static String dameAtributoNodo(String nombre, Node nodo) {
        NamedNodeMap mapa = nodo.getAttributes();
        String valor = null;
        if(mapa!=null) {
            Node nodoAt = mapa.getNamedItem(nombre);
            if(nodoAt!=null)
                valor = nodoAt.getNodeValue();
```

```
        }
        return valor;
    }

}
```

Ahora nos crearemos otra clase de utilidades que denominaremos UtilidadesXPath, que nos servirá de base para nuestros ejemplos:




Nos creamos un XML como el que pusimos de ejemplo (el de los libros) y lo colocamos en el classpath (directamente sobre src para que lo copie a bin) y le llamamos libros.xml:

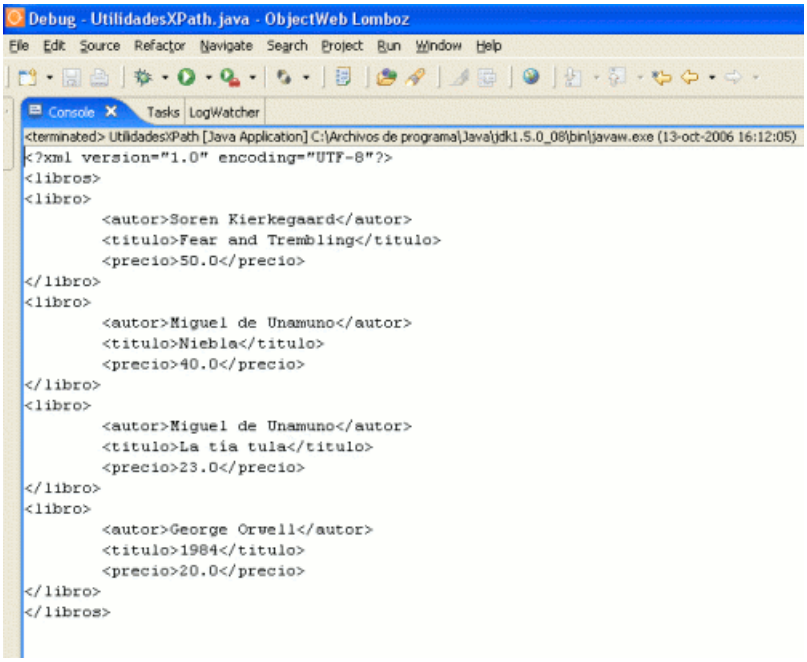



Empecemos con una pequeña prueba para comprobar que nos funcionan algunos métodos de la clase UtilidadesXML (vamos a leer el fichero libros.xml y lo vamos a imprimir en la consola):

```
package autentia.tutoriales.xml.xpath;

import org.w3c.dom.Document;
import autentia.tutoriales.xml.UtilidadesXML;

public class UtilidadesXPath {
```

```
    public static void main(String[] args) {
        try {
            Document doc = UtilidadesXML.File2Document("libros.xml");
             System.out.println(UtilidadesXML.Node2String(doc.getDocumentElement()));
        } catch (Exception e) {
            e.printStackTrace();
        }
     }

}
```

Ejecutamos y mostramos la consola:




Empecemos a usar XPath. Primero vamos a recordar las preguntas que nos hicimos:

- ¿ Qué libros son de Miguel de Unamuno?
- ¿ Qué libros tienen un precio menor a 30 euros ?
- ¿ Hay algún libro que tenga como parte del título la palabra "Fear" ?

Traduzcamos a expresiones XPath:

- /libros/libro[autor="Miguel de Unamuno"]
- /libros/libro[precio < 30.0]
- /libros/libro[contains(titulo,"Fear")]

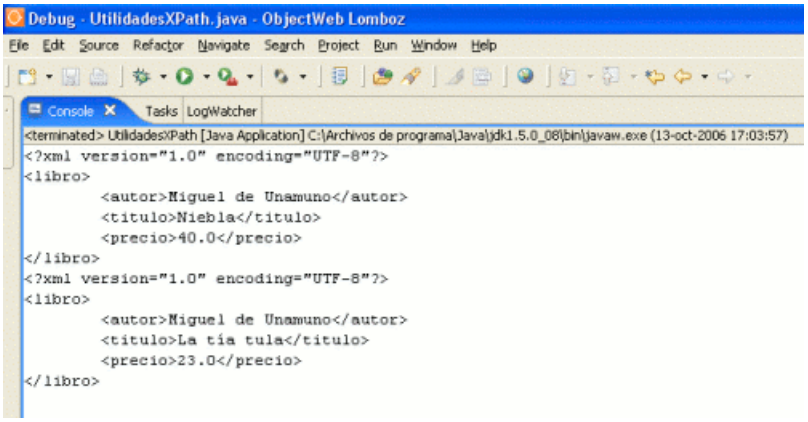
**Nota:** *Mi intención no es explicar exhaustivamente todas las posibilidades de las expresiones, operadores y funciones de XPath, sino hacer aparecer en el lector el deseo de aprenderlas al contemplar lo que facilita el trabajo.*

Usemos las expresiones:

¿ Qué libros son de Miguel de Unamuno?:

```
public static void main(String[] args) {
    try {
        Document doc = UtilidadesXML.File2Document("libros.xml");

        XPath xpath = XPathFactory.newInstance().newXPath();
        String expression = "/libros/libro[autor=\"Miguel de Unamuno\"]";
        NodeList nodes = (NodeList) xpath.evaluate(expression, doc.getDocumentElement(), XPathConstants.NODESET);

        System.out.println(UtilidadesXML.Nodes2String(nodes));

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
```

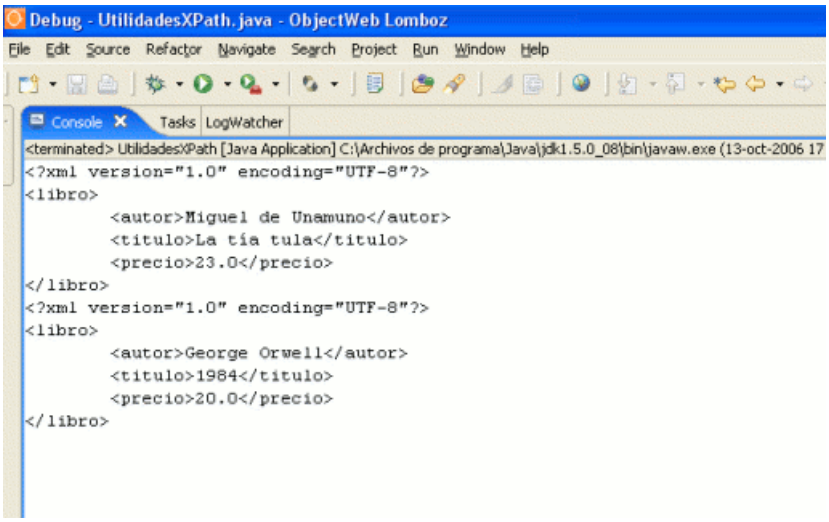
Ejecutamos y mostramos la consola:




¿ Qué libros tienen un precio menor a 30 euros ?

```
public static void main(String[] args) {
      try {
         Document doc = UtilidadesXML.File2Document("libros.xml");

         XPath xpath = XPathFactory.newInstance().newXPath();
         String expression = "/libros/libro[precio < 30.0]";
         NodeList nodes = (NodeList) xpath.evaluate(expression, doc.getDocumentElement(), XPathConstants.NODESET);

         System.out.println(UtilidadesXML.Nodes2String(nodes));

      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

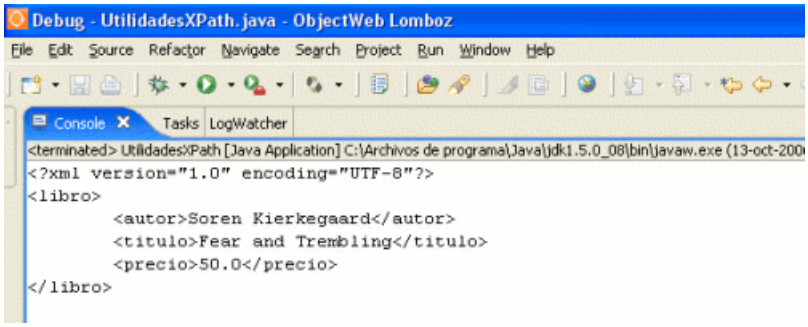
Ejecutamos y mostramos la consola:




¿ Hay algún libro que tenga como parte del título la palabra "Fear" ?

```
public static void main(String[] args) {
      try {
         Document doc = UtilidadesXML.File2Document("libros.xml");

         XPath xpath = XPathFactory.newInstance().newXPath();
         String expression = "/libros/libro[contains(titulo,\"Fear\")]";
         NodeList nodes = (NodeList) xpath.evaluate(expression, doc.getDocumentElement(), XPathConstants.NODESET);

         System.out.println(UtilidadesXML.Nodes2String(nodes));

      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

Ejecutamos y mostramos la consola:




Creo que con esto es suficiente para hacernos una idea de lo que se puede hacer con XPath.

Nos hemos dejado muchos operadores por usar, no hemos buscado en los atributos, no hemos usado "axes", pero eso lo dejo para la inquietud del lector, yo ya me lo sé.

Si necesitas que te echemos una mano, pues ya sabes, a eso nos dedicamos en Autentia.

Puedes opinar sobre este tutorial aquí

## Recuerda

que el personal de Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales)

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?

**¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?**

info@autentia.com

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos .....
**Autentia = Soporte a Desarrollo & Formación**

Formación en nuevas tecnologías

Autentia S.L. Somos expertos en:
**J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..**
y muchas otras cosas

---

## Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

| Subscribirse a Novedades | |
|---|---|
| ***e-mail*** | |
| | Enviar |

---

## Otros Tutoriales Recomendados (También ver todos)

| Nombre Corto | Descripción |
|---|---|
| Soporte XML en Eclipse con X-MEN | Alejandro Perez nos enseña como potenciar el entorno eclipse para facilitarnos el trabajo con ficheros xml, gracias al pluggin X-MEN |
| XML básico | Si quieres ver de un modo visual como crear un documento XML, este es tu tutorial. Este es el primero de un conjunto de tutoriales que iremos publicando sobre esta fascinante y amplia tecnología |
| XMLEncryption en Java | En este magnífico tutorial, Alberto Carrasco nos enseña los fundamentos y un ejemplo práctico de XMLEncryption. |
| Schemas XML. Introducción | Los esquemas XML (schemas XML) son una evolución natural de las DTDs. Os mostramos |

| | |
|---|---|
| esquemas XML | como emprezar con esta tecnología. |
| Analizar ejecución de programa Java | Os mostramos como investigar el comportamiento de vuestros programas Java, en ejecución, a través del profiling. |
| Novedades en Java 1.5 | Ya está disponible la versión Beta del J2SDK 1.5. Os mostramos algunas de las nuevas características introducidas en el lenguaje Java: Clases genéricas, enumeraciones, bucles simplificados, etc. |
| XML y XSL en Cliente | En este tutorial os enseñamos como formaterar documentos XML directamente en vuestro navegador a través de Plantillas XSL. En cursos sucesivos veremos como hacerlo en el servidor, para no crear dependencias con el navegador del cliente. |
| Transformación de XML y XSL en JSPs | Os mostramos como poder utilizar XML y XSL en JSPS, combinado con el Patrón MVC |
| Optimización Java con Eclipse Profiler Plugin | Alejandro Pérez nos enseña como analizar el rendimiento de nuestras aplicaciones con Eclipse Profiler Plugin. |
| Procesamiento XML en Java con JAXB y WSDP 1.6 | Os mostramos como instalar la versión 1.6 de WSDP y como procesar los ficheros XML con uno de sus componentes, JAXB |