# Elaboración de una guía de uso y aplicación sobre DocBook.

**Alex Paños Bueno**

# Elaboración de una guía de uso y aplicación sobre DocBook.

Alex Paños Bueno

publicado 05-08-2009

# Dedicatoria

Esta memoria va dedicada a mi padre.

# Tabla de contenidos

# Lista de figuras

# Capítulo 1. Introducción.

## Tabla de contenidos



La intención principal de esta memoria es la de mostrar la potencia e idoneidad de DocBook para la creación de documentación. DocBook se creó originariamente para la creación de documentación técnica, pero realmente DocBook es útil para crear cualquier tipo de documento puesto que permite generar diferentes formatos de presentación a partir de un mismo documento.

## 1. Utilidad XML y DocBook.

DocBook es útil para por ejemplo, crear la documentación de una asignatura. En el capítulo 3 reproducimos la guía que se encuentra en http://futura.disca.upv.es/~smm/practiques/xml/practicaText.xml y que muestra como se trabaja con XML y algunas de sus herramientas. Este ejemplo de trabajo con XML es perfecto para introducir al uso de DocBook ya que está escrito en XML.

Viendo este ejemplo de edición XML observamos que con un editor de texto cualquiera podemos editar un ejemplo muy sencillo con unas pocas etiquetas. Pero a la hora de crear y editar un documento escrito en DocBook que conste de varios capítulos con imágenes, índices, tablas, enlaces web, y en definitiva cualquier tipo de elemento que se utilice para crear un libro, es necesario el poder contar con una serie de herramientas que faciliten la edición y trabajo a la hora de crear documentación.

Este es el objetivo principal de esta memoria, el mostrar que existen una serie de programas que facilitan la tarea de creación de un archivo XML sencillo o de un archivo XML de mayor envergadura como es un libro DocBook.

## 2. ¿Por que leer esta memoria?

Mediante la lectura de esta memoria se pretende evaluar las herramientas disponibles para la creación y edición de documentación en XML y DocBook. Dependiendo de la magnitud del trabajo que se quiera acometer se debe seleccionar una herramienta u otra. El lector de esta memoria podrá ver cómo editar un documento XML sencillo y un documento escrito en DocBook como es este proyecto. Seleccionaremos 3 o 4 editores con el fin de elegir los mejores para editar y crear la documentación.

Comenzamos con la edición de una guía de XML en la que se muestra como crear un ejemplo sencillo, como comprobar si está bien formado y como validarlo. También se muestra la edición sencilla de XSLT y de una DTD con la que validar los documentos XML. Tras este ejemplo sencillo pasaremos a crear un libro DocBook mostrando que elementos utilizar y las ayudas que el editor pone a disposición de los autores con el fin de facilitar la creación de contenidos.

Una vez vista la edición y creación de XML y DocBook veremos las herramientas de las que disponemos para crear las transformaciones de los documentos a diferentes formatos como por ejemplo HTML y PDF.

## 3. ¿A quién va dirigida la memoria?

Esta memoria puede ser leída por cualquier persona que desee introducirse en la creación de contenidos mediante el uso de XML y DocBook. Para que la memoria sea útil nos hemos esforzado en que la selección de herramientas utilizadas en el proyecto cumplan la condición de que sean de uso gratuito, aunque algunas de ellas con restricciones, de este modo cualquier usuario puede utilizar las herramientas analizadas.

En principio teniendo unas pocas nociones de XML se puede seguir sin ninguna dificultad tanto el capitulo de edición de XML como el de DocBook. En la sección 5 de este mismo capítulo podemos encontrar una pequeña introducción a qué es XML, DocBook, XSLT y DTD. Para encontrar información sobre la sintaxis a utilizar en XML, DocBook, XSLT y DTD el lector puede acudir a la bibliografía donde se detallan diversas páginas que el lector puede consultar para aclarar dudas que puedan surgir en la lectura de esta memoria.

# 4. Organización de la memoria.

El libro se divide en 6 capítulos.

1. Introducción.

   Qué aporta la lectura de esta memoria, a quien va dirigida y una pequeña introducción a la temática de la que es objeto.

2. Editores.

   Una pequeña descripción de la tipología de editores XML y DocBook existentes y el análisis y selección de los que a nuestro juicio se adaptan mejor a nuestras necesidades.

3. Ejemplo básico de edición sobre XML.

   Ejemplo sencillo de edición, validación, transformación y publicación de un documento XML.

4. Ejemplo avanzado: DocBook.

   Ejemplo de creación de un libro con todos sus elementos utilizando DocBook.

5. Conversión a otros formatos.

   Uso de programas para producir la conversión de documentos DocBook o XML a otros formatos como PDF o HTML.

6. Conclusiones.

   Qué hemos aprendido utilizando los diferentes programas para la realización de la memoria. Qué hemos cubierto y que nos hemos dejado para trabajos futuros.

7. Bibliografía e índice.

# 5. XML, DocBook, SGML y HTML.

En este apartado vamos a introducir toda la terminología que envuelve a XML . Para ello realizaremos una pequeña introducción en la que explicaremos que es SGML, HTML, XML, DocBook y la relación existente entre ellos.

Del mismo modo explicaremos brevemente el resto de tecnologías que rodea a XML y que se utilizan para otros fines como por ejemplo llevar acabo la comprobación de un documento escrito en XML, su validación y su transformación a otros formatos como por ejemplo HTML y PDF.

1. XML:

   XML, Extensible Markup Language, es un lenguaje de marcas que se utiliza para la creación de lenguajes de marcado específicos para diferentes necesidades. XML se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Es un estándar posterior a SGML con una funcionalidad similar aunque más sencillo y mas fácil de aplicar.

   XML viene a cubrir el vació existente entre SGML y HTML ya que HTML es fácil de aprender y utilizar pero su uso es limitado y SGML es bastante más complicado de utilizar debido a la gran cantidad de etiquetas disponibles. La idea que se persiguió con la creación de XML era el poder utilizar toda la potencia de SGML con la facilidad de uso de HTML.

XML es una tecnología que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. 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.

2. DocBook:

   DocBook es una DTD (Document Type Definition) definida en XML (Extensible Markup Language) o en SGML (Standard Generalised Markup Language).

   DocBook es un aplicación del estándar SGML/XML que se utiliza principalmente para documentar todo tipo de material y programas informáticos. Mediante su uso los autores centran en la creación y estructuración del contenido, no en su presentación final.

   DocBook se ajusta perfectamente a la escritura de documentación técnica pero no se limita únicamente a este fin. Una de las ventajas de utilizar DocBook es que un documento escrito en DocBook puede ser transformado a otros formatos de salida como HTML, PostScript, PDF, RTF, etc.

   Actualmente el uso de DocBook está en aumento y se utiliza para documentar una variedad de proyectos. Entre los más importante destacamos:

   - The GNOME Documentation Project [http://developer.gnome.org/projects/gdp/]
   - The KDE Documentation Project [http://i18n.kde.org/doc/]
   - The Debian Documentation Project [http://www.debian.org/doc/ddp]
   - The Linux Documentation Project [http://linuxdoc.org/]
   - The FreeBSD Documentation Project [http://www.freebsd.org/docproj/]

3. SGML:

   SGML, The Standard Generalized Markup Language, es un metalenguaje utilizado para la organización y etiquetado de documentos. SGML puede considerarse como el padre de los lenguajes de marcado. De SGML provienen HTML y XML entre otros lenguajes de marcado.

   SGML proviene del GML (Generalized Markup Language) definido por IBM en los años setenta y que surgió por la necesidad que tenían en la empresa de almacenar y organizar grandes cantidades de información para facilitar su intercambio y utilización.

   La Organización Internacional de Estándares (ISO) normalizó GML en 1986 creando el SGML que no era más que el GML pero estandarizado ya que el ámbito de aplicación de GML era particular a las necesidades de IBM.

4. HTML:

   HTML (Hypertext Markup Language) es otra DTD definida en SGML. La DTD especifica como ciertos componentes del lenguaje deben ser interpretados. El lenguaje HTML, no es mas que una aplicación de SGML para dar formato a documentos para su posterior representación en la web.

# 6. Herramientas para trabajar sobre XML y DocBook.

Para trabajar con DocBook se necesitan las siguientes herramientas:

1. Un editor de texto, un editor de XML o de DocBook:

   A lo largo de este libro utilizaremos principalmente XmlMind Personal Edition y EditiX 2009 Lite Version. El uso de uno u otro programa vendrá condicionado por las necesidades de edición y la idoneidad de cada una de las herramientas a la tarea que se desee realizar.

2. Tener DocBook instalado:

- El DTD de DocBook es el que define la estructura, elementos y atributos de un documento DocBook.
- El DTD generalmente se instala junto a los editores XML, EditiX 2009 Lite Version y XmlMind Personal Edition instalan una copia local de la DTD de DocBook.
- Las hojas de estilo XSLT DocBook, que permiten generar varios formatos de salida del documento como pueden ser: HTML, XHTML, PS, PDF, RTF, etc. Al igual que las DTD, la mayoría de editores incluyen las hojas de estilo necesarias para transformar los documentos a otros formatos.

3. Un procesador XSLT:

   Este procesador será el encargado de realizar la transformación del documento DocBook al formato elegido. Para ello utiliza las hojas de estilo que serán las que indiquen al procesador las transformaciones que se han de realizar sobre el documento. Entre los procesadores mas utilizados nos encontramos con xsltproc, Xalan y Saxon.

4. Un procesador FO, en caso de querer transformar a PDF:

   Un procesador FO permitirá la transformación de un archivo XML a otro formato preparado para realizar una impresión del documento en papel. Entre los procesadores mas utilizados nos encontramos con Apache FOP, RenderX XEp, PassiveTeX, xmlroff, etc.

# Capítulo 2. Editores.

## Tabla de contenidos



A la hora de elegir un editor para trabajar con XML debemos prestar atención a una serie de puntos que harán que el usuario se decante por uno u otro tipo de editor. La elección del editor dependerá de las necesidades finales del usuario.

Además del coloreado de sintaxis que ofrecen muchos editores de texto plano y editores de código, algunos editores XML presentan unas características como por ejemplo el facilitar la validación del documento que se está escribiendo, el autocompletado de elementos o el cerrado automático de etiquetas. Estas son algunas de las características que pueden ayudar a prevenir fallos en la escritura de un documento.

# 1. Tipos de editores.

Atendiendo a la manera de enfocar la edición del archivo, nos encontramos con los siguientes tipos de editores:

## 1.1. Editores textuales.

Los editores textuales de XML suelen facilitar el trabajo con las etiquetas de los elementos ya sea bien cerrando automáticamente las etiquetas o proponiendo etiquetas a utilizar. El resaltado o coloreado de sintaxis es una característica básica de cualquier editor textual de XML. El autocompletado de elementos y atributos basado en una DTD o SCHEMA es una característica que suele estar disponible en la mayoría de editores textuales de XML. El autoidentado del código y el mostrar el número de línea suelen ser también características bastante comunes en este tipo de editores.

La principal ventaja de los editores textuales es que presentan la información de la misma manera que esta guardada en el archivo XML.

## 1.2. Editores gráficos.

Los editores gráficos suelen ser bastante mas fáciles de utilizar para la mayoría de usuarios que los editores textuales y además pueden no requerir del conocimiento de la sintaxis de XML.

Los editores gráficos los podemos clasificar en WYSIWYG y WYSIWYM.

- WYSIWYG: What you see is what you get. Lo que ves es lo que obtienes. El editor nos muestra el resultado gráfico final, permite editar un documento viendo directamente cual será su formato de presentación.

- WYSIWYM: What you see is what you mean. Lo que ves es lo que quieres decir. Es un paradigma para la creación de documentos alternativo al mas conocido WYSIWYG. En este paradigma el usuario se encarga de

introducir los contenidos de forma estructurada siguiendo su valor semántico en lugar de indicando su formato de representación final. Por ejemplo, indicando que lo que se está escribiendo es un titulo, una sección, un autor, etc.

Para utilizar este tipo de editores es necesario conocer a priori la estructura del documento que se va a crear. La principal ventaja es que produce una total separación entre contenido y presentación por lo que el usuario solo ha de preocuparse en crear la estructura y agregar los contenidos.

# 2. Características a tener en cuenta en un editor.

1. Plataforma: GNU/Linux, Mac Os X, MSWindows.

2. Coste: si es software libre, comercial, freeware, shareware, etc.

3. Coloreado sintaxis: Posibilidad de mostrar el código XML en diferentes colores y tipografías

4. Validación XML: Comprobación de que el documento está bien formado y se ajusta a una estructura definida en una DTD.

5. Validación en tiempo real: Mientras editas el documento el programa comprueba continuamente si se están utilizando los elementos correctos.

6. Autocompletado de elementos, atributos y valores de los atributos. Listado de posibles elementos atributos o valores que se muestran cuando el usuario está escribiendo una etiqueta.

7. Actuar en la estructura del documento: Mover, cortar, pegar nodos u otros elementos del documento XML.

8. Crear elementos seleccionando texto: Seleccionando texto se permite seleccionar elementos de una lista.

9. Cambio de elementos: Cambiar un elemento por otro permitido.

10. Ver la estructura del documento: Posibilidad de ver la estructura del documento de manera jerárquica y de contraer y expandir sus nodos.

11. Activación-desactivación de la validación: Para poder trabajar con documentos no válidos.

# 3. Editores a evaluar

La metodología para la selección de los editores a evaluar en este proyecto ha consistido en realizar una búsqueda en Internet de editores XML y Docbook que cubran las necesidades principales de cualquier usuario de este tipo de programas.

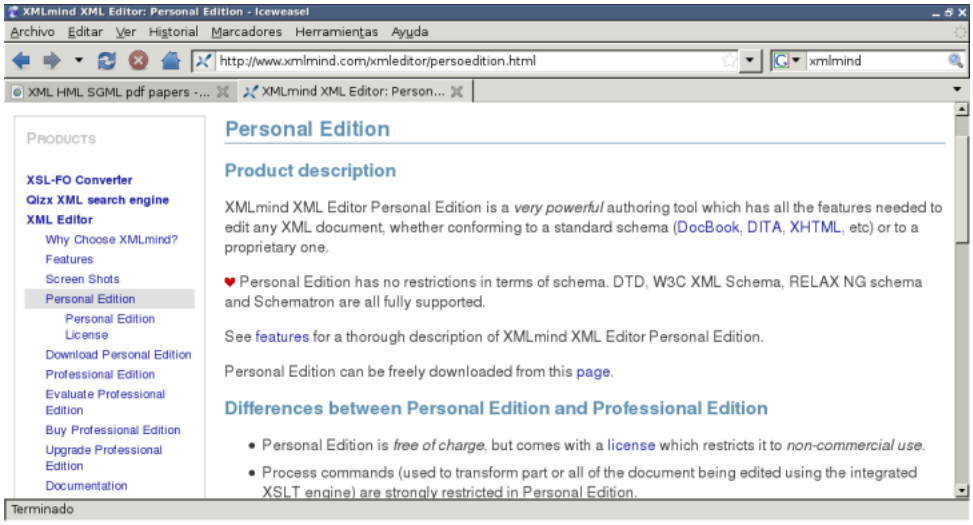
Principalmente se ha buscado que el coste del software sea nulo para el usuario, por ello incluimos en esta comparativa programas comerciales que incluyen versiones para un uso personal y un programa completamente libre.

Los editores seleccionados son los siguientes:

## 3.1. XmlMind Personal Edition.

XMLmind Personal Edition es un editor XML que permite la edición de documentos XML largos y complejos como pueden ser documentos escritos en DocBook. XmlMind Personal Edition no es una herramienta para programadores , está pensado para que lo puedan utilizar autores de documentación técnica. La página web en la que podemos encontrar el programa es http://www.xmlmind.com/xmleditor/persoedition.html

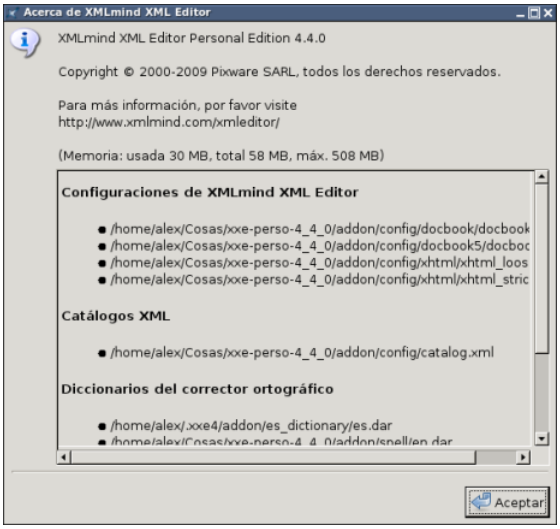
**Figura 2.1. Web XmlMind Personal Edition**




Para instalar la Edición Personal debemos ir a http://www.xmlmind.com/xmleditor/download.shtml , y elegir para qué sistema operativo queremos bajar el programa. Existen versiones para MSWindows, GNU/Linux y Mac Os X.

En el momento de escribir esta memoria hemos utilizado la versión 4.4.0. Hemos utilizado tanto la versión para MSWindows como para GNU/Linux.

**Figura 2.2. Versión utilizada de XmlMind Personal Edition.**



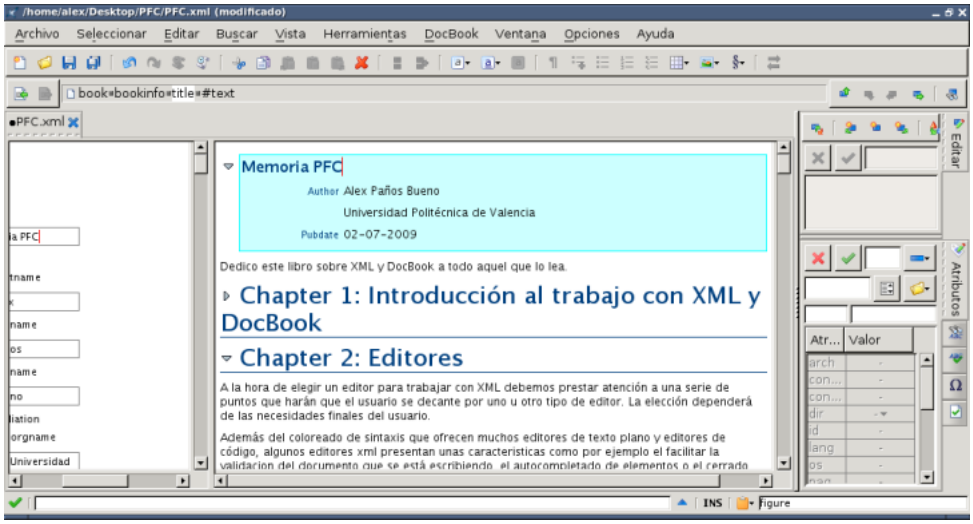

La instalación en ambos sistemas es bastante sencilla.

Si vamos a utilizar MSWindows simplemente debemos descargar el ejecutable con o sin la máquina virtual de java, dependiendo de si tenemos o no java instalado en el sistema.

En caso de utilizar GNU/Linux debemos bajar el archivo comprimido ya sea el zip o el tar.gz y descomprimirlo. Una vez descomprimido iremos a la carpeta en la que se encuentra el ejecutable que en nuestro caso es /home/alex/Cosas/xxe-perso-4_4_0/bin/ y ejecutaremos el archivo `xxe.sh` bien mediante un doble click si se utiliza un administrador de archivos como nautilus o mediante la terminal de ejecutando el siguiente comando, en nuestro caso: cd /home/alex/Cosas/xxe-perso-4_4_0/bin/ y una vez en el directorio ./`xxe.sh`.

El aspecto del programa es el siguiente:

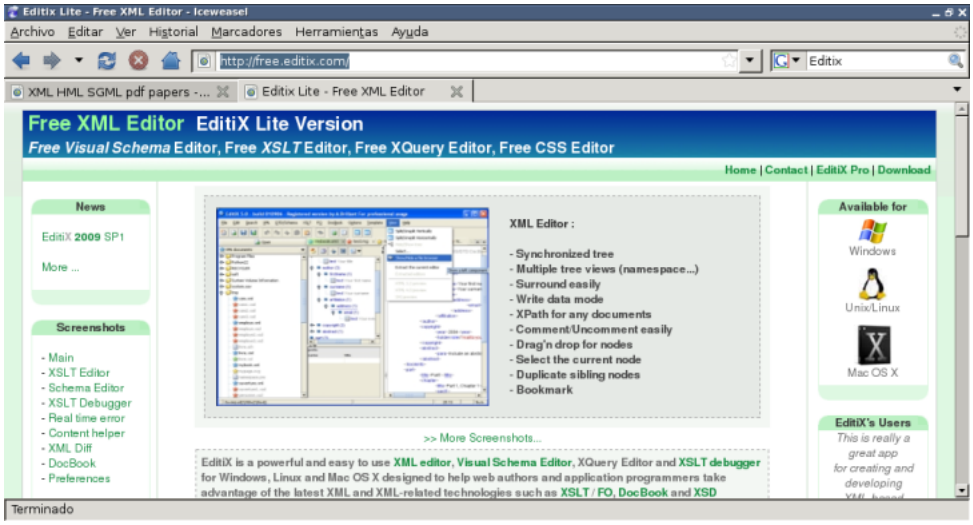
**Figura 2.3. XmlMind Personal Edition**




## 3.2. EditiX Lite Version

EditiX Lite Version es un editor XML sencillo de utilizar, diseñado para ayudar a los autores de documentación y programadores a utilizar las ultimas tecnologías relacionadas con XML como por ejemplo XSLT/FO, DocBook y XSD Schema. EditiX Lite Version se encuentra disponible en http://free.editix.com/

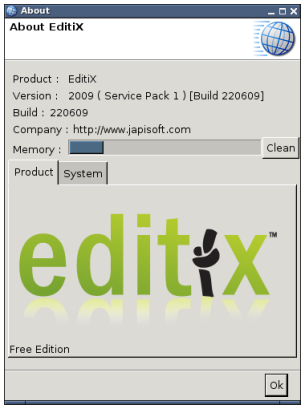
**Figura 2.4. Página Web Editix Lite Version**




Para instalar la Lite Version debemos ir a http://free.editix.com/download.html , aquí eligiéremos para que sistema operativo queremos bajar el programa. Existen versiones para MSWindows, GNU/Linux y Mac Os X.

En el momento de escribir esta memoria hemos utilizado la versión 209 (Service Pack 1) (Build 220609) tanto la versión para MSWindows como para GNU/Linux.

### Figura 2.5. Versión utilizada de EditiX Lite Version



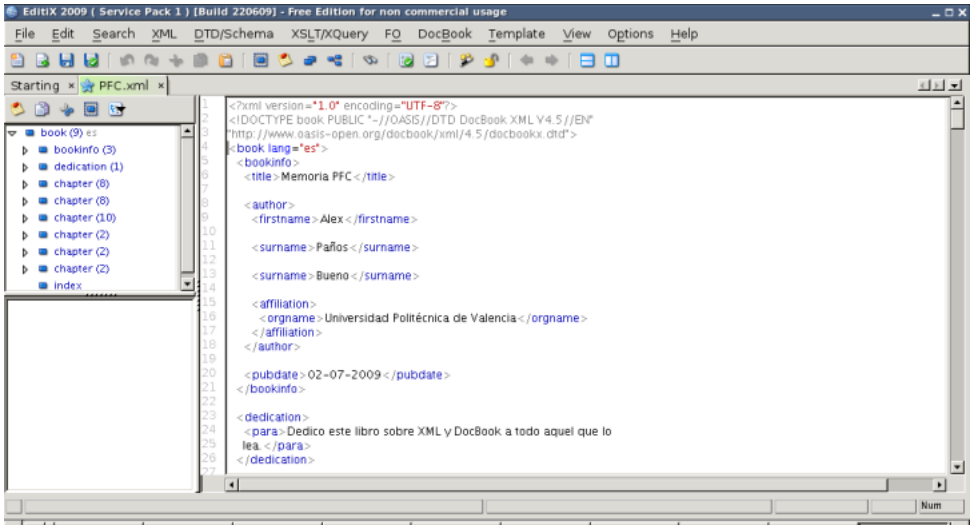

La instalación en ambos sistemas es bastante sencilla.

Si vamos a utilizar MSWindows simplemente debemos descargar el ejecutable con o sin la máquina virtual de java, dependiendo de si tenemos o no java instalado en el sistema.

En caso de utilizar GNU/Linux debemos bajar el archivo comprimido ya sea el zip o el tar.gz y descomprimirlo. Una vez descomprimido iremos a la carpeta en la que se encuentra el ejecutable que en nuestro caso es /home/alex/Cosas/editix-free-2009sp1/bin/ y ejecutaremos el archivo `run.sh` bien mediante un doble click si se utiliza un administrador de archivos como nautilus o mediante la terminal ejecutando el siguiente comando, en nuestro caso: cd /home/alex/Cosas/editix-free-2009sp1/bin/ y una vez en el directorio ./`run.sh`.

La ventana de edición de EditiX Lite Version:

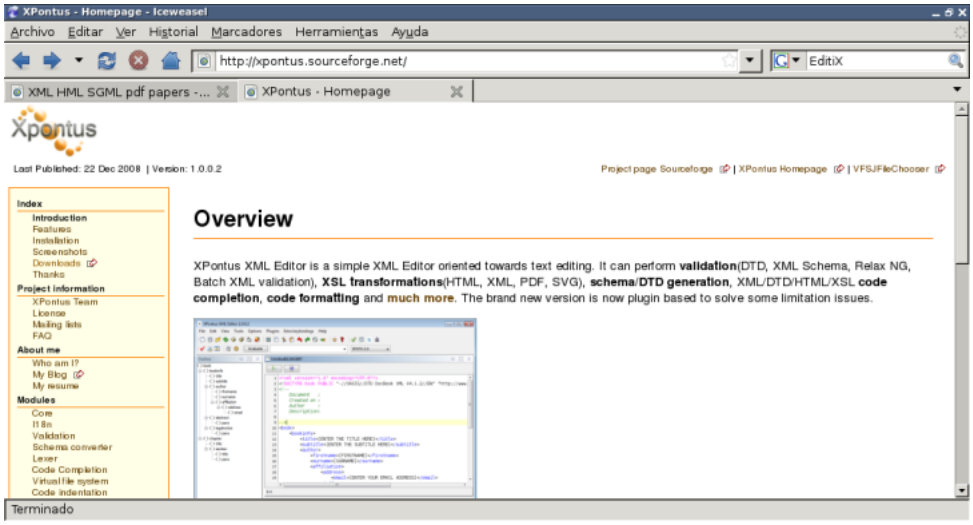
### Figura 2.6. EditiX Lite Version




## 3.3. Xpontus

Xpontus es un editor XML que pretende ser una alternativa libre a los editores comerciales. Esta aplicación intenta ofrecer las características que se pueden encontrar en otros entornos de edición y autoría XML. Podemos encontrar Xpontus en http://xpontus.sourceforge.net/

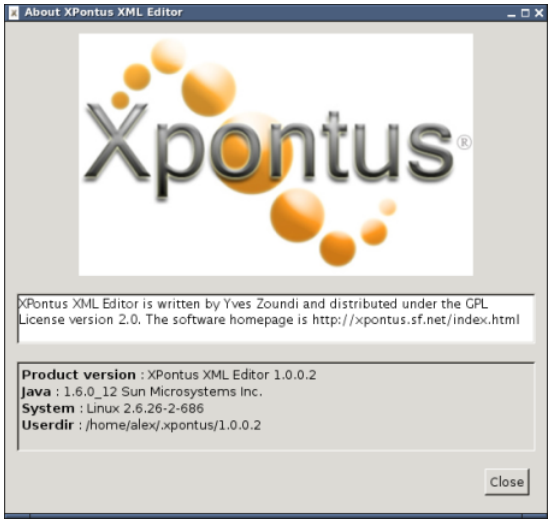
### Figura 2.7. Pagina Web Xpontus




Para instalar XPontus debemos ir a http://sourceforge.net/projects/xpontus/files/ , aquí elegiremos para que sistema operativo queremos bajar el programa. Existen versiones para MSWindows, GNU/Linux y Mac Os X.

En el momento de escribir esta memoria hemos utilizado la versión 1.0.0.2 para GNU/Linux.

### Figura 2.8. Versión utilizada de Xpontus



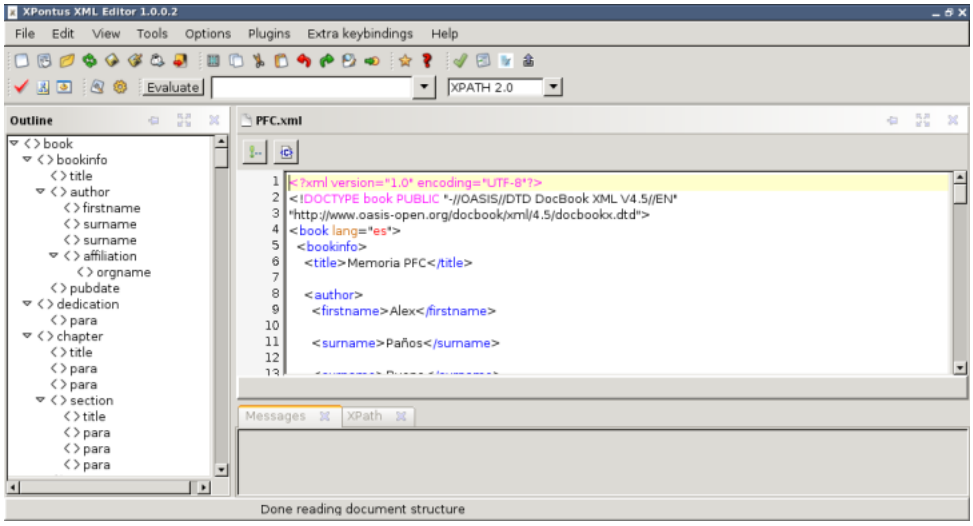

La instalación es similar a las anteriores.

Podemos descargar versiones para GNU/Linux, MSWindows, Mac OS X.

En caso de utilizar GNU/Linux debemos bajar el archivo comprimido ya sea el zip o el tar.gz y descomprimirlo. Una vez descomprimido iremos a la carpeta en la que se encuentra el ejecutable que en nuestro caso es /home/alex/Cosas/XPontus/ y ejecutaremos el archivo `xpontus.sh`, bien mediante un doble click si se utiliza un administrador de archivos como nautilus o mediante la terminal ejecutando el siguiente comando, en nuestro caso: cd /home/alex/Cosas/XPontus/ y una vez en el directorio ./`xpontus.sh`.

Una vez iniciemos el programa nos aparecerá la siguiente ventana:

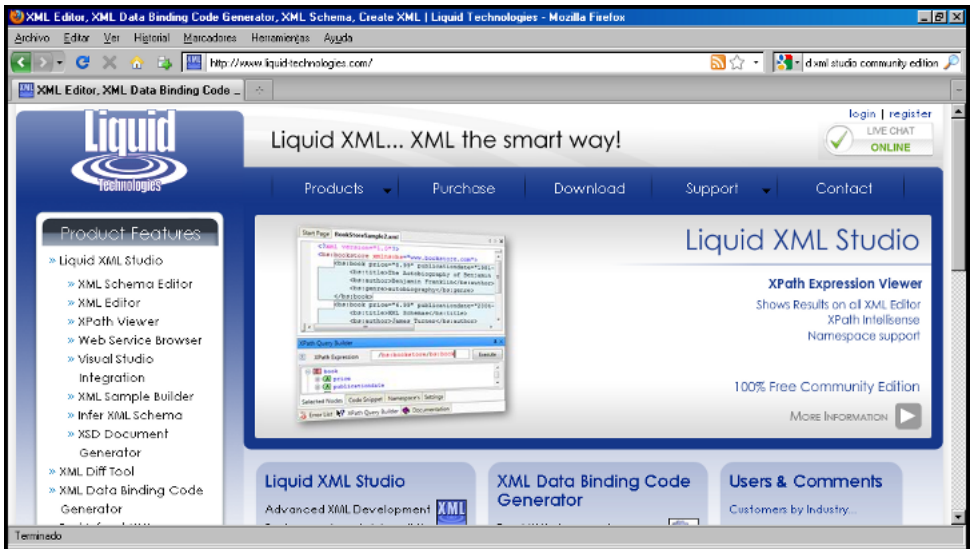
**Figura 2.9. Xpontus**




## 3.4. Liquid XML Editor Community Edition

Liquid XML Editor Community Edition es una suite de edición XML que permite la creación y validación de documentos XML. Podemos encontrar Liquid XML Editor Community Edition en http://www.liquid-technologies.com/Download.aspx

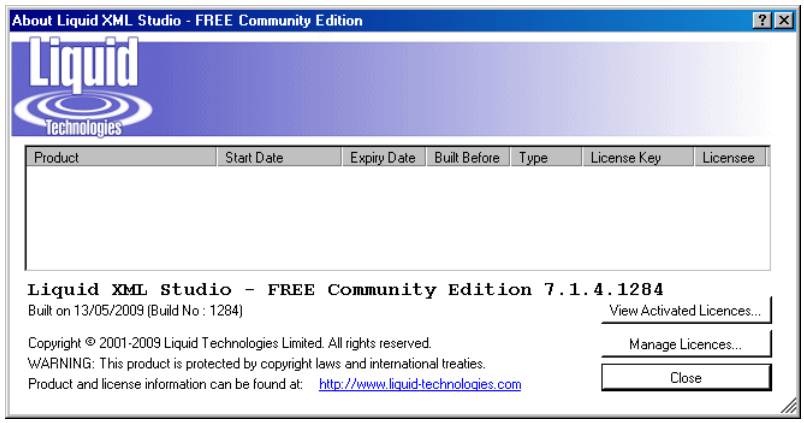
**Figura 2.10. Página Web Liquid XML Editor**




Para instalar Liquid XML Editor Community Edition debemos ir a http://www.liquid-technologies.com/Download.aspx, desde ese enlace podremos bajar el programa. Liquid XML Editor Community Edition solo está disponible para MSWindows .

En el momento de escribir esta memoria hemos utilizado la versión 7.1.4.1284.

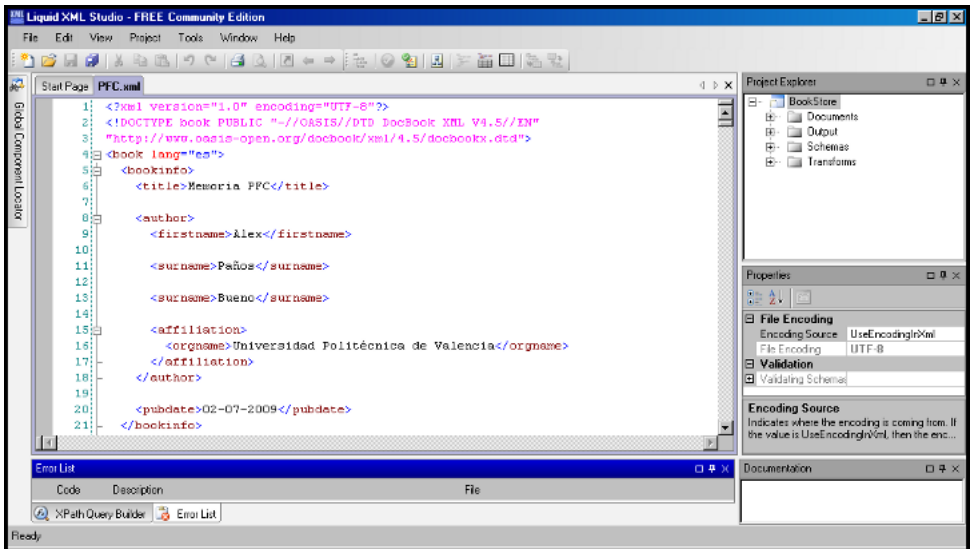
### Figura 2.11. Versión utilizada de Liquid XML Editor




La instalación del editor es muy sencilla , solo hay que bajar el ejecutable de la página mencionada anteriormente.

Una vez iniciemos el programa nos aparecerá la siguiente ventana:

### Figura 2.12. Liquid XML Editor




# 4. Comparativa de editores.

En esta tabla mostramos las principales características de los editores seleccionados:

### Tabla 2.1. Comparativa Editores XML

| | XMLMind Personal Edition | EditiX Lite Version | Xpontus | Liquid XML Editor Community Edition |
|---|---|---|---|---|
| **Plataforma** | MSWindows, Mac Os X, GNU/Linux | MSWindows, Mac Os X, GNU/Linux | MSWindows, Mac Os X, GNU/Linux | MSWindows |
| **Coste** | Gratuito para uso no comercial | Gratuito para uso no comercial | Software libre | Gratuito para uso no comercial |
| **Coloreado de sintaxis** | N | S | S | S |
| **Validación XML** | S | S | S | S |

| | | | | |
|---|---|---|---|---|
| **Validación en tiempo real** | S | S | N | N |
| **Autocompletado de elementos** | S | S | S | S |
| **Actuar en la estructura del documento** | S | S | N | N |
| **Crear elementos seleccionando texto** | S | S | N | N |
| **Cambio de elementos** | S | S | N | N |
| **Ver la estructura del documento** | S | S | S | N |
| **Activación-desactivación de la validación** | N | S | S | S |

# 5. Editores seleccionados para el proyecto.

Los editores seleccionados para la redacción de este proyecto han sido EditiX Lite Version y XmlMind Personal Edition.

EditiX Lite Versión lo hemos seleccionado para la edición del documento XML sencillo ya que permite la edición de documentos XML, DTD y XSLT de una manera sencilla. Permite la validación de XML frente a una DTD, permite la edición y depuración XSLT, comprobación de sintaxis en tiempo real, autocompletado de elementos, etc. Además dispone de una extensa documentación de referencia a la que acudir en caso de necesidad.

XmlMind Personal Edition ha sido seleccionado ya que resulta el editor mas cómodo en cuanto a uso para la creación y edición de documentos DocBook. Mediante el uso de la hoja de estilo que incorpora el editor podemos crear un libro o artículo sin escribir ni una línea de código. XmlMind Personal Edition es el editor que, a nuestro juicio, facilita en mayor medida la creación de un documento DocBook.

En el capitulo 3 utilizaremos EditiX Lite Version para la edición de un ejemplo en XML, la edición de una DTD y la edición XSLT. En este capítulo se muestra las principales características de este editor y como utilizarlo.

En el capítulo 4 utilizaremos XmlMind Personal Edition para crear un libro, mostraremos como estructurar el libro y como ir añadiendo los capítulos, imágenes, listados, tablas, etc.

En cuanto a Xpontus y Liquid XML Editor Community Edition los hemos descartado por diversas razones:

- Xpontus a pesar de ser un editor muy potente que incluye una serie de características idóneas para su uso en este proyecto tiene como principal defecto una documentación de referencia bastante escasa y esa ha sido la principal razón para descartar este editor.
- Liquid XML Editor Community Edition es un editor que en su versión Community Edition nos viene bastante recortado en cuanto a características disponibles, el programa, a diferencia de Xpontus, cuenta con una documentación bastante extensa, pero esta versión viene excesivamente limitada frente a la versión de pago.

# Capítulo 3. Ejemplo básico de edición sobre XML.

## Tabla de contenidos



## 1. ¿Qué es XML?

XML (eXtensible Markup Language) es un lenguaje extensible de marcado que describe una clase de objetos de datos llamados documentos XML y describe parcialmente el comportamiento de los programas que los procesan.

XML se puede considerar como un metalenguaje que permite diseñar un lenguaje propio de etiquetas para múltiples clases de documentos.

La estructura de un documento XML se basa en la declaración de tipos de documentos (Document Type Declaration o DTD) o en la definición de XML Schemas.

La DTD proporciona la gramática para una clase de documentos XML. Esta gramática contiene la definición del conjunto de etiquetas que puede contener esa clase de documentos XML.

Se proporciona un tutorial para ampliar la información sobre XML: Tutorial de XML. [http://futura.disca.upv.es/~smm/practiques/xml/enlaces/IntroXML.pdf]

## 2. EditiX 2009 Personal Edition.

EditiX Lite Version es un editor XML cuyo objetivo principal es permitir la creación de archivos XML de una manera sencilla. Utilizaremos la Personal Edition puesto que es gratuita para un uso no comercial. Desde la sección de descargas [http://free.editix.com/download.html] del sitio web de EditiX Lite Version puedes bajarte el programa. Esta versión es la que se va a utilizar para la realización de los ejercicios y para familiarizarnos con el proceso de creación, edición, comprobación, procesamiento y publicación de un documento escrito mediante XML.

En esta guía haremos especial hincapié en la ventaja de utilizar un editor como EditiX Lite Version frente a la utilización conjunta de un editor de texto plano como puede ser el bloc de Notas de MS/Windows (o el vi de UNIX) y un conjunto de scripts ejecutables desde MS/DOS o el terminal de GNU/Linux.

Para utilizar EditiX es necesario disponer de Java VM 5.x. Si no tienes Java VM 5.x o superior en tu maquina, puedes descargar EditiX Lite Version con la máquina virtual de java incluida o descargarla de http://www.java.com [http://www.java.com/en/].

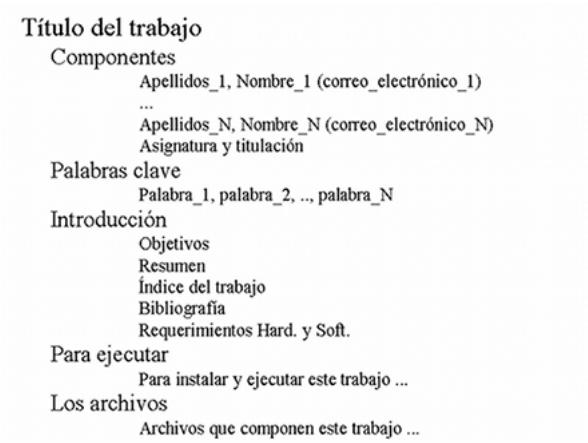
EditiX Lite Version está disponible tanto para MS/Windows como para GNU/Linux y Mac Os X con lo que se puede seguir la guía y realizar los ejercicios desde cualquiera de los tres sistemas operativos mencionados anteriormente.

## 3. Edición XML.

Crearemos un archivo XML bastante sencillo paso a paso para así poder ver tanto las ventajas como los inconvenientes del uso de esta herramienta frente a la edición con un editor de texto.

Dicho documento contendrá información relativa al trabajo de la asignatura como el título, componentes del grupo y resumen.

### Figura 3.1. Estructura documento XML




### Ejemplo 3.1. Ejemplo 1

En el siguiente enlace se incluye un ejemplo de documento XML para representar la anterior estructura.

Ejemplo 1 de documento XML [http://futura.disca.upv.es/~smm/practiques/xml/enlaces/trabajoIMDv1.zip]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <Trabajo>
  <Titulo> Práctica de XML </Titulo>
  <Grupo>
   <Componente>
    <Nombre>Manuel Agustí</Nombre>
   </Componente>
   <Componente>
    <Nombre>Vicente Atienza</Nombre>
   </Componente>
   <Componente>
    <Nombre>J.Vicente Benlloch</Nombre>
   </Componente>
   <Componente>
    <Nombre>Félix Buendía</Nombre>
   </Componente>
  </Grupo>
  <Resumen>
    <Parrafo>El trabajo consiste en explicar un lenguaje como XML</Parrafo>
    <Parrafo>Para ello, se describirá la edición de documentos XML</Parrafo>
    <Parrafo>A continuación se comprobará la validez de dichos documentos</Parrafo>
    <Parrafo>También se verán las opciones para procesar documentos XML</Parrafo>
  </Resumen>
 </Trabajo>
```

Un documento XML puede empezar con la siguiente declaración que indica la codificación y la versión de XML, `<?xml version="1.0" encoding="ISO-8859-1"?>`

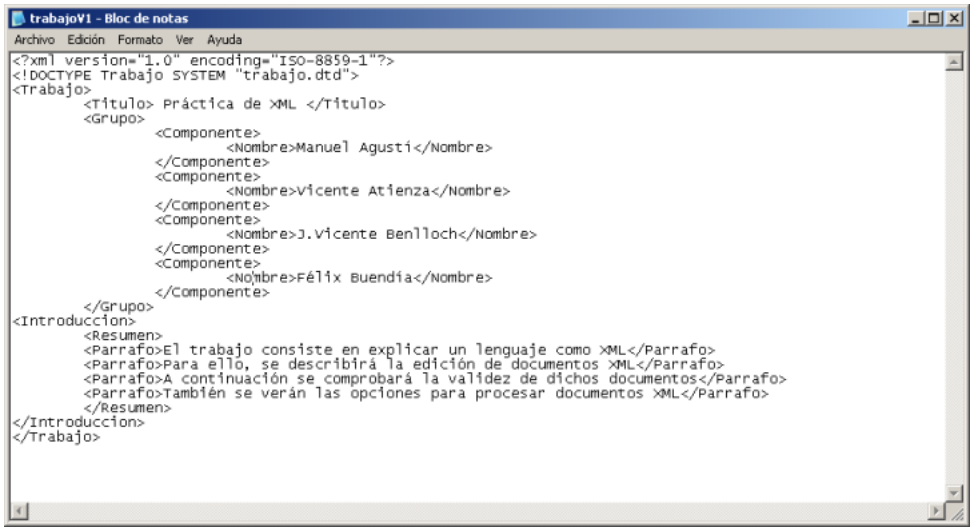
En el ejemplo del enlace, el elemento raíz está representado por la etiqueta `<Trabajo>`

El resto de elementos representan información relativa al título, a los componentes del grupo y a los párrafos del resumen.

El elemento Titulo contiene una cadena de caracteres. Los elementos Grupo y Resumen constan de otro nivel de elementos (Componente y Párrafo, respectivamente) que a su vez contienen cadenas de caracteres.

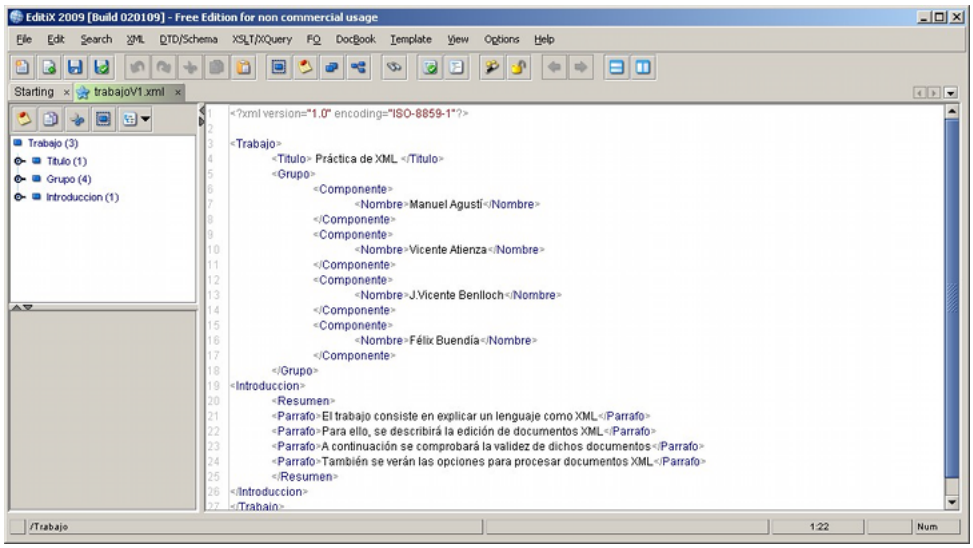
Editado con un editor de texto como el bloc de notas de MS/Windows, el archivo tendría el siguiente aspecto:

### Figura 3.2. Edición con el bloc de notas




Con EditiX Lite Version:

### Figura 3.3. Edición con EditiX Lite Version




## Ejemplo 3.2. Ejercicio 1

Se trataría ahora de escribir un documento XML que deberá contener datos como el título del trabajo, nombre de los componentes del grupo y resumen. La introducción de los datos se realizará con EditiX Lite Version. y el nombre del fichero deberá coincidir con "`trabajo.xml`".

Para crear el archivo en EditiX Lite Version, tras ejecutar el programa debemos ir a File # New # Standard XML Document, con lo que crearemos un nuevo documento XML.

Nos aparecerá un documento en blanco cuya primera línea, `<?xml version="1.0" encoding="UTF-8"?>`, es la versión de XML y la codificación a utilizar , con lo que nos ahorramos el introducirla manualmente como haríamos con editores de texto como el bloc de Notas.

Escribe el documento completo, el nombre del fichero introducido deberá coincidir con "`trabajo.xml`".

Una de las principales ventajas del uso de este editor frente a editores de texto simples es la ayuda que ofrecen a la hora de introducir las etiquetas. Comienza escribiendo la etiqueta `<Trabajo>` tras la línea 1, verás como inmediatamente el editor cierra la etiqueta introduciendo `</Trabajo>`. Escribe el documento completo, el nombre del fichero introducido deberá coincidir con "`trabajo.xml`".

Otra ventaja de este programa es el coloreado de sintaxis y el autoidentado del código. Una vez hayas introducido todas las etiquetas ve a XML # Format # Pretty Format (default), el texto junto a sus etiquetas se anidarán de forma jerárquica dando un aspecto mas claro al código escrito.

# 4. Comprobación de XML.

Tras la crear el documento debemos comprobar si está bien formado, es decir, si cumple con las especificaciones XML.

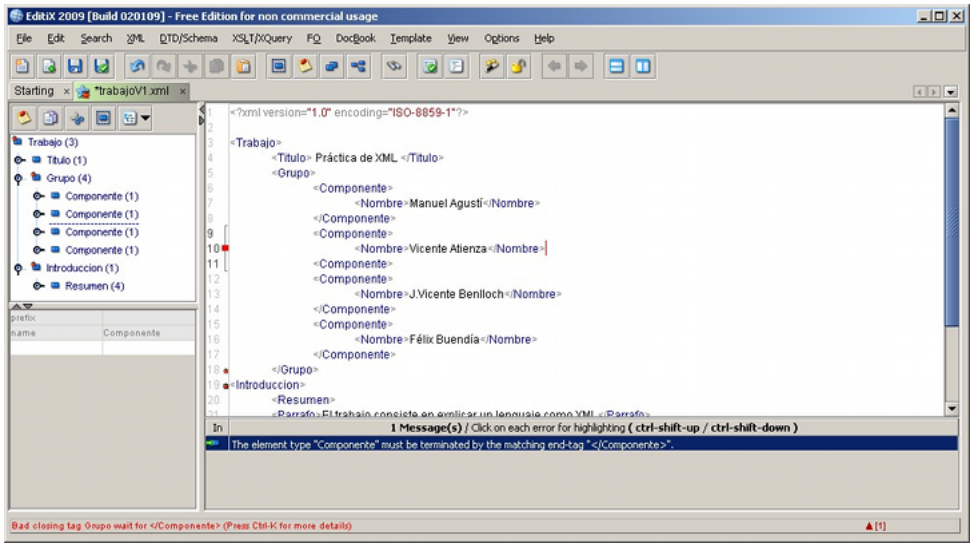
### Ejemplo 3.3. Ejemplo 2

Para comprobar si el documento esta bien formado utilizaremos un analizador (“parser”). EditiX Lite Versión incluye Xerces como analizador. Comprobar si el documento está bien formado es tan sencillo como ir a XML # Check This Document. En caso de que el documento esté bien formado el programa nos mostrará una ventana indicando que el documento es correcto.

### Figura 3.4. Comprobación del documento


En caso de que el documento no esté correctamente formado el programa nos indica en la parte inferior de la ventana tanto que elemento ha producido el error como además como subsanar el error para que el documento sea correcto. En la siguiente imagen hemos eliminado la marca de cierre de la etiqueta `<Componente>` de la línea 11.

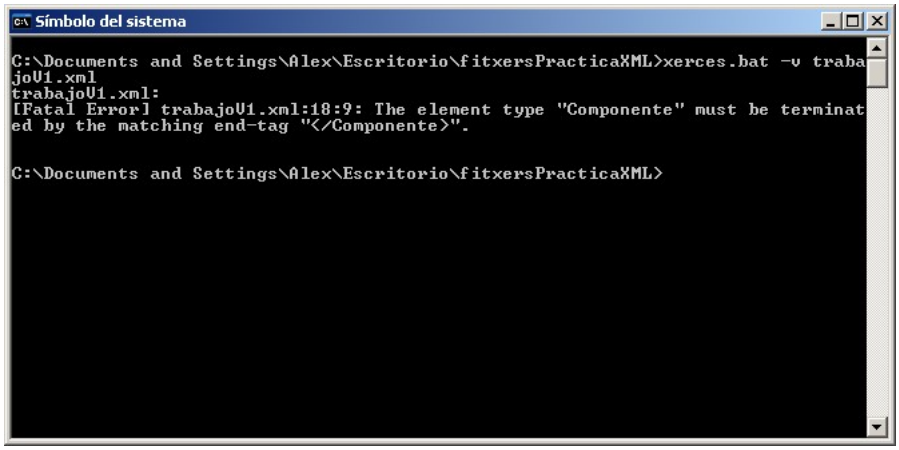
### Figura 3.5. Error de comprobación




La ventaja de utilizar el validador incluido en EditiX Lite Versión es clara, con un par de clicks de ratón el programa nos indica si el documento es correcto. En caso de utilizar un editor de texto plano deberíamos comprobar el archivo utilizando un analizador externo al editor ya que este no cuenta con uno propio. En caso de querer utilizar Xerces deberíamos abrir una ventana MS/-DOS, ir al directorio donde se encuentra la carpeta con los archivos de la práctica que anteriormente hemos bajado y ejecutar el siguiente orden: **`xerces.bat –v trabajoV1.xml`** .

La siguiente imagen muestra el error que arroja el parser analizador tras eliminar la marca de cierre de la etiqueta Componente del archivo `trabajoV1.xml`.

### Figura 3.6. Error en Xerces




## Ejemplo 3.4. Ejercicio 2

Este ejercicio consiste en indicar el resultado del análisis sintáctico para el siguiente ejemplo de documento XML.

Ejemplo 2 de documento XML [http://futura.disca.upv.es/~smm/practiques/xml/enlaces/trabajoIMDv2.zip]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Trabajo>
 <Titulo/>
 <Introduccion>
  <Resumen>
   <Parrafo>El trabajo consiste en ...</Parrafo
  </Resumen>
 </Introduccion>
 <Grupo>
  <Componente>
   <Nombre>Felix Buendía</Nombre>
  </Componente>
 </Grupo>
</Trabajo>
```

Para ello abriremos el archivo con EditiX Lite Version.

Como podéis ver no es necesario hacer nada ya que el programa al abrir el archivo indica cual es el error y la manera de solucionarlo.

Hasta ahora hemos comprobado que los archivos XML están bien formados. El siguiente paso será comprobará si el archivo es válido según una DTD, la que el archivo tenga asignada o la que se indique.

La validación del documento XML se basa en la "declaración de tipo de documento" (DTD). La siguiente línea representa un ejemplo de declaración que se puede añadir al principio del documento, a continuación de la versión de XML y el tipo de codificación.

```
<!DOCTYPE Trabajo SYSTEM "trabajo.dtd">
```

Dicha declaración no es obligatoria y define cómo encontrar la información de la DTD, mediante un identificador público (PUBLIC) o mediante un Identificador Universal de Recursos (URI) precedido por la palabra SYSTEM.

## Ejemplo 3.5. Ejemplo 3

En este caso se proporciona un fichero "`trabajo.dtd`" para definir la estructura del trabajo.

Ejemplo de DTD [http://futura.disca.upv.es/~smm/practiques/xml/enlaces/trabajoIMD.dtd.zip]

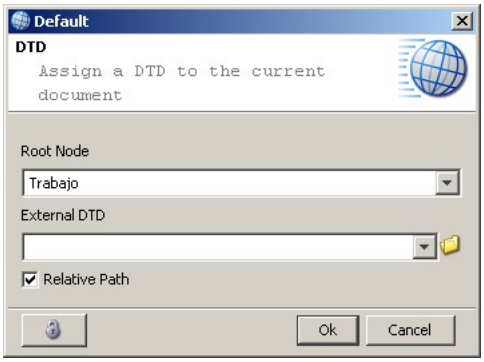
```
<!--
 Editat en Xemacs [versio 21.4] (http://www.xemacs.org/) per M. Agusti (UPV), 2003
 xml version="1.0" encoding="UTF-8"
 -->
<!-- edited with XML Spy v3.0.7 (http://www.xmlspy.com) by FBU (UPV) -->

<!ELEMENT Trabajo (Titulo, Grupo, PalabrasClave?, Introduccion, Desarrollo?, Archivos?)>
<!ELEMENT Titulo (#PCDATA)>

<!ELEMENT Grupo (Componente+)>
<!ELEMENT Componente (Nombre, Datos?, foto?)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Datos EMPTY>
<!ATTLIST Datos
 Asignatura CDATA #REQUIRED
 Titulacion (Inf | Doc) #IMPLIED
 Correo CDATA #IMPLIED
>
<!ELEMENT foto EMPTY>
<!ATTLIST foto
          fichero  CDATA #REQUIRED
          ancho    CDATA #REQUIRED
          alto     CDATA #REQUIRED
>


<!ELEMENT PalabrasClave (Item+)>
<!ELEMENT Introduccion (Objetivos?, Resumen, Requerimientos?, Bibliografia?)>
<!ELEMENT Objetivos (Frase+)>
<!ELEMENT Resumen (Parrafo+)>
<!ELEMENT Requerimientos (Parrafo+)>
<!ELEMENT Bibliografia (Item*)>
<!ELEMENT Desarrollo (Capitulo+)>
<!ELEMENT Capitulo (#PCDATA)>
<!ATTLIST Capitulo
 Titulo CDATA #REQUIRED
 Enlace CDATA #IMPLIED
>
<!ELEMENT Archivos (Enlace*)>
<!ELEMENT Item (#PCDATA)>
<!ELEMENT Parrafo (#PCDATA)>
<!ELEMENT Frase (#PCDATA)>
<!ELEMENT Enlace (#PCDATA)>
<!ATTLIST Enlace
 Descripcion CDATA #IMPLIED
>
```

Los pasos a seguir para validar el documento utilizando la DTD son los mismos que los utilizados para verificar el documento, a excepción que se incluye la declaración de DTD (`<!DOCTYPE Trabajo SYSTEM "trabajo.dtd">`). La única diferencia es que sin DTD la validación consiste en ver si el documento escrito está bien formado, e incluyendo la DTD además de comprobar si el documento está bien formado se comprueba si es valido conforme a la definición de tipo de documento (DTD).

Ahora procederemos a incluir la DTD dentro del archivo utilizando EditiX Lite Version, para ello iremos a DTD/Schema # Assign DTD to document.

Nos aparecerá una ventana, pinchando en el icono de la carpeta podremos seleccionar la DTD del documento. Si guardamos la DTD en la misma carpeta que el archivo XML dejaremos marcado el checkbox de Relative Path.

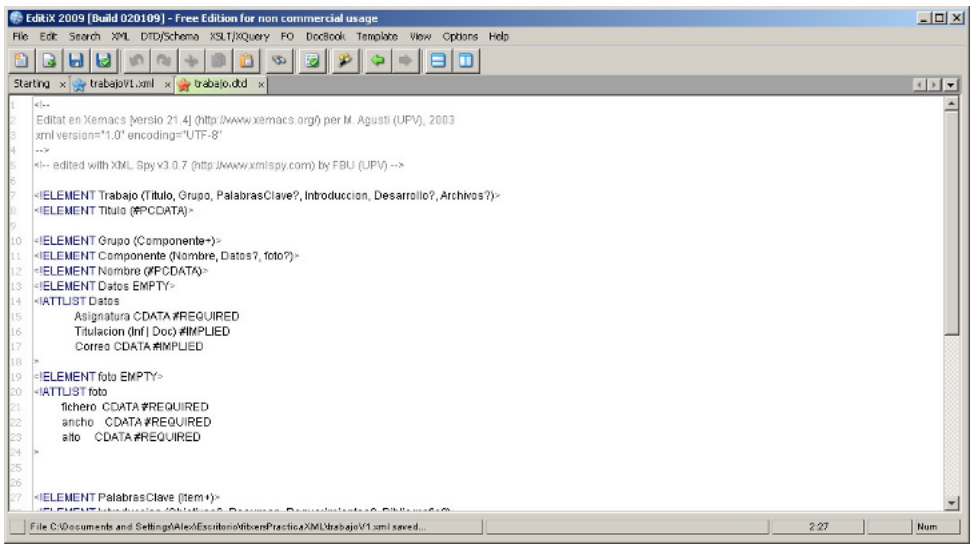
## Figura 3.7. Asignar DTD a un documento




Para validar el documento contra la DTD hay que ir a XML # Check This Document .

Puedes crear o editar una DTD mediante EditiX Lite Version, para ello has de ir a File # Open, en Archivos de tipo seleccionar DTD documents (*.dtd) para poder visualizar los DTD de la carpeta y así seleccionarlo y abrirlo.

## Figura 3.8. Edición de DTD




Una vez hayamos incluido la DTD cada vez que abramos una etiqueta se nos abrirá un cuadro con todas las posibles etiquetas que podemos utilizar en ese momento y que están permitidas conforme a la DTD. Esto reduce significativamente la posibilidad de cometer errores a la hora de editar un documento con EditiX Lite Version frente a la edición con un editor de texto como el Bloc de Notas.

EditiX Lite Version también permite comprobar la sintaxis de las DTD, para ello hemos de ir a DTD/Schema # Check This Document.

### Ejemplo 3.6. Ejercicio 3

Aplicar la validación al documento "`trabajoV2.xml`". Indicar el tipo de error y como se resolverá.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Trabajo>
 <Titulo/>
 <Introduccion>
  <Resumen>
   <Parrafo>El trabajo consiste en ...</Parrafo
  </Resumen>
 </Introduccion>
 <Grupo>
  <Componente>
   <Nombre>Felix Buendía</Nombre>
  </Componente>
 </Grupo>
</Trabajo>
```

### Ejemplo 3.7. Ejercicio 4

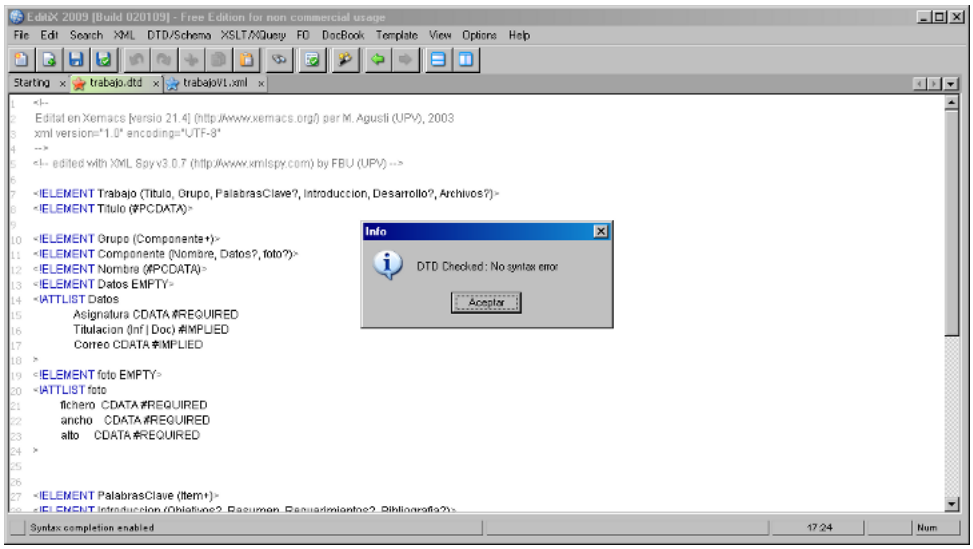
Se trata de aplicar el analizador al documento XML introducido en el apartado Edición de XML y observar el resultado.

Mientras el documento XML no sea validado, no podrá someterse a la siguiente fase de transformación.

# 5. Edición DTD.

Vamos a realizar un pequeño cambio en el fichero `trabajoV1.xml` y veremos como Editix Lite Version nos muestra los errores al verificar el archivo contra la DTD. Con la DTD `trabajo.dtd` abierta, vamos a abrir el fichero `trabajoV1.xml`. Una vez abierta la DTD comprobaremos si la sintaxis es correcta :

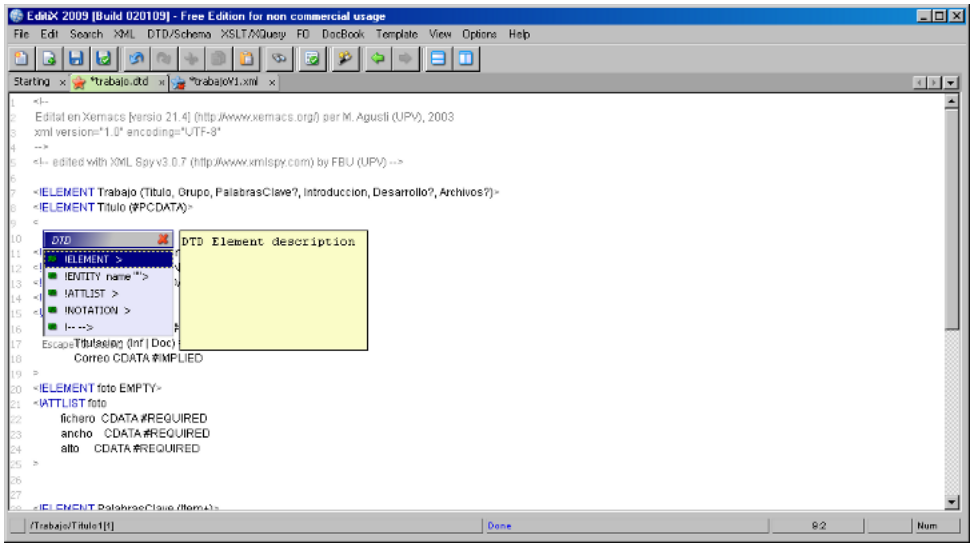
### Figura 3.9. Comprobación sintaxis DTD




Vemos que la sintaxis de la DTD si es correcta.

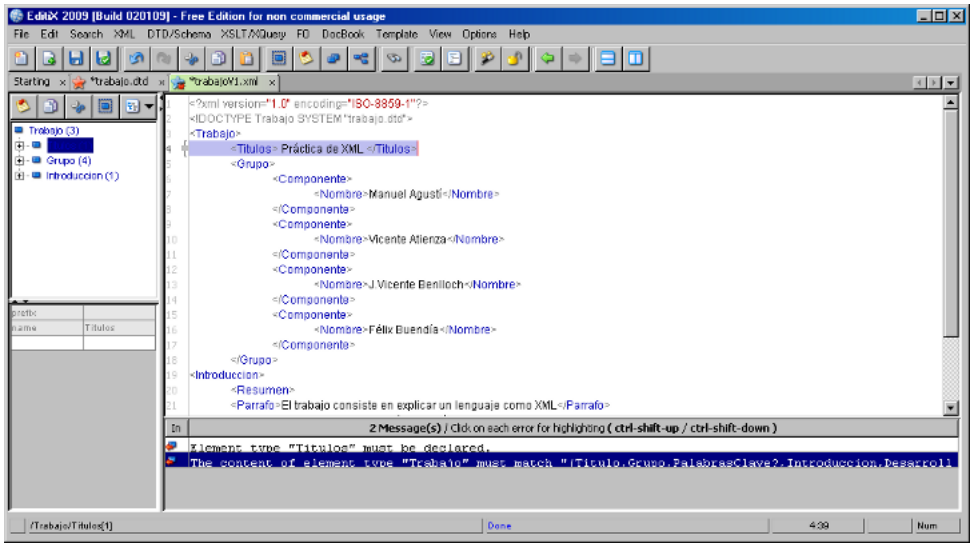
Al comenzar la escritura de un elemento en la DTD escribiendo "`<!`", EditiX Lite Version nos proporciona un asistente de contenido para facilitarnos la edición de la DTD.

**Figura 3.10. Asistente DTD**




Ahora vamos a probar a cambiar el elemento `<Titulo>` por `<Titulos>` en el fichero `trabajoV1.xml`. Como el fichero lleva asociada la DTD `trabajo.dtd`, el editor nos mostrará el siguiente error:

**Figura 3.11. Elemento no definido en la DTD**



Para solucionar este error tan solo debemos ir a la DTD `trabajo.dtd` y definir un nuevo elemento Titulos o simplemente añadir una "s" al elemento Titulo.

**Figura 3.12. Edición DTD**

```
<!ELEMENT Trabajo (Titulos, Grupo, PalabrasClave?, Introduccion, Desarrollo?, Archivos?)>
<!ELEMENT Titulos (#PCDATA)>
```

# 6. Procesamiento de XML.

El siguiente paso consistirá en procesar el documento XML introducido, con el fin de obtener diferentes representaciones de la información en formatos como HTML, PDF,etc.

Para ello se utilizarán "scripts" XSL, es decir, programas que utilizan como datos de entrada documentos XML y obtienen como salida el formato elegido.

## Ejemplo 3.8. Ejemplo 4

Ejemplo de "script" XSL para convertir a HTML.

Ejemplo de XSL [http://futura.disca.upv.es/~smm/practiques/xml/enlaces/trabajoIMD.xsl.gz]

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:xlink="http://www.w3.org/1999/xlink">
 <!-- xsl:import href="xprueba2html.xsl"/
  -->
 <xsl:template match="Trabajo">
 <HTML>
 <HEAD>
 <TITLE>
  <xsl:value-of select="Titulo" />
  </TITLE>
  </HEAD>
 <BODY>
 <h1>
  <xsl:value-of select="Titulo" />
  </h1>
 <xsl:for-each select="Introduccion/Resumen/Parrafo">
 <p>
  <xsl:value-of select="." />
  </p>
  </xsl:for-each>
  </BODY>
  </HTML>
  </xsl:template>
  </xsl:stylesheet>
</Trabajo>
```

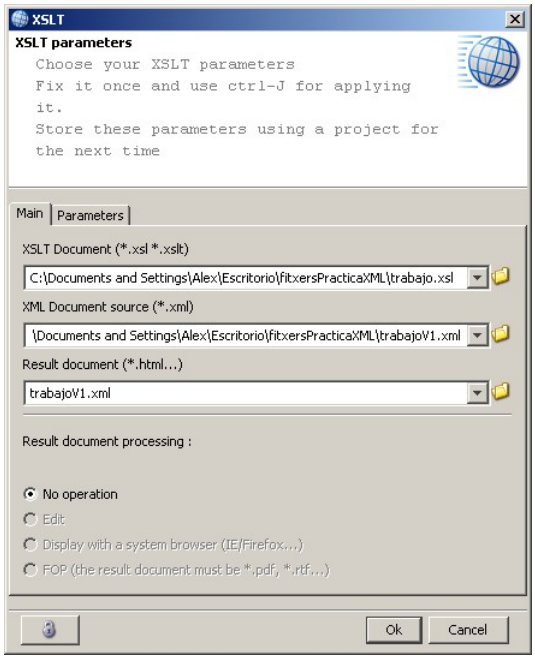
En el tutorial de XML proporcionado se incluye una introducción al lenguaje XSL.

## Ejemplo 3.9. Ejemplo 5

Vamos a proceder a utilizar el script XSL para convertir a HTML mediante EditiX Lite Version.

Iremos a XSLT/Xquery # Transform using XSLT..., seleccionamos `trabajo.xsl`, `trabajoV1.xml` y escribimos como queremos que se llame el archivo HTML generado tras la transformación

## Figura 3.13. Transformación a HTML utilizando XSLT



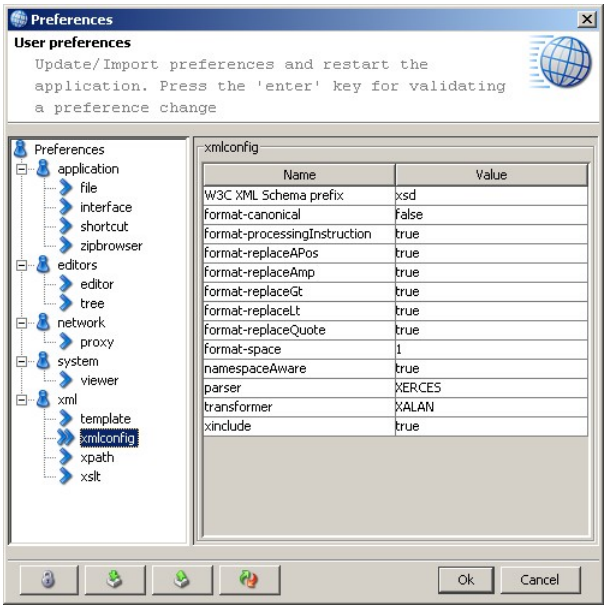

El programa, aplicará la transformación indicada en "`trabajo.xsl`" y generará como resultado un fichero HTML con el nombre que hayamos seleccionado.

EditiX Lite Version también se puede utilizar para la edición de archivos *.xsl. Los pasos a seguir para editar un archivo .xsl son los mismos que los utilizados para editar un archivo DTD pero seleccionando como tipo de archivos del tipo XSLT documets (*.xsl *.xslt).

Para la transformación del documento se utiliza una herramienta llamada Xalan.

En la siguiente imagen se puede ver que EditiX Lite Version utiliza como analizador Xerces y como herramienta de transformación Xalan.

**Figura 3.14. Xalan y Xerces en EditiX Lite Version**




La integración de ambas herramientas en EditiX Lite version es una ventaja. En caso de utilizar un editor de textos y un procesador por separado la metodología a seguir seria la apertura de una ventana MS/DOS, ir al directorio donde se encuentra la carpeta con los archivos de la práctica que anteriormente hemos bajado y ejecutar la orden en la ventana MS/DOS.

### Ejemplo 3.10. Ejercicio 5

Indique cuál es el resultado de la ejecución de la anterior transformación.

Anotar el código o instrucción XSL encargado de visualizar el elemento "Resumen" del trabajo.

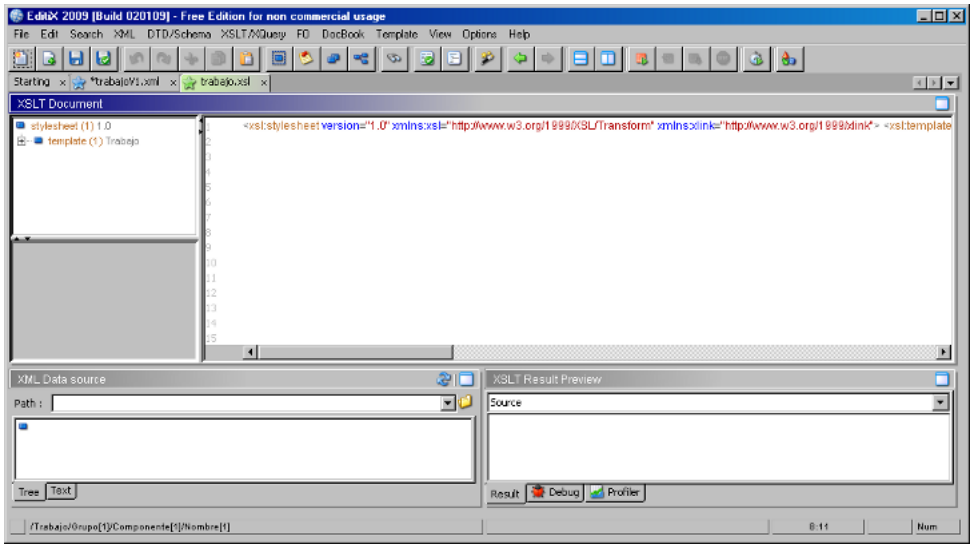
# 7. Edición XSLT.

EditiX Lite Version proporciona también ayuda a la hora de crear o editar una hoja XSL. Vamos a utilizar la hoja XSL de la guía para aplicársela al archivo `trabajoV1.xml`. En primer lugar abriremos tanto el archivo XML como el XSL. Con el archivo `trabajoV1.xml` abierto iremos a XSLT/XQuery # Assign XSLT to this document... y seleccionaremos `trabajo.xsl`. Aparecerá una línea similar a:

```
<?xml-stylesheet type="text/xsl" href="..\..\..\..\..\..\Escritorio\pfc\fitxersPracti-
caXML\trabajo.xsl"?>
```

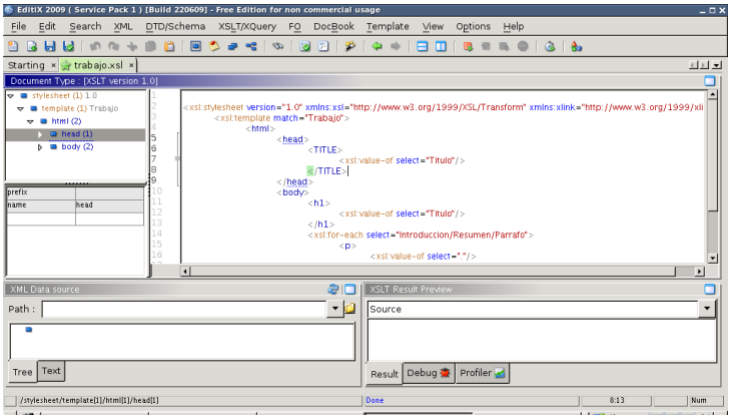
Ahora vamos a la pestaña donde hemos abierto la hoja XSL:

## Figura 3.15. Edición XSL



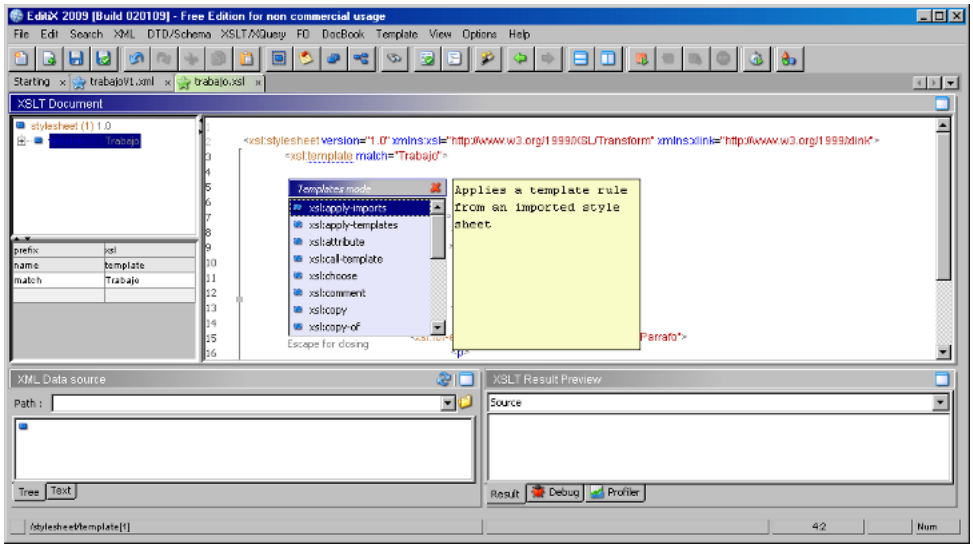

Como se puede ver el editor abre el fichero `trabajo.xsl` y muestra su contenido en una sola línea. Simplemente haciendo click con el botón derecho sobre el texto y seleccionando la opción Pretty format (default) pasaremos a ver el contenido del fichero de una manera mucho mas cómoda.

## Figura 3.16. Pretty format




Al igual que con XML y DTD, Editix Lite Version proporciona un asistente de contenido que se abre en cuanto abres una etiqueta con "<":.

**Figura 3.17. Asistente Contenido XSLT**




Una de las características que mas nos habría gustado probar es el depurador XSLT, pero no está disponible en la edición gratuita.

# 8. Publicación de XML.

Además del proceso de transformación descrito en el apartado Procesamiento de XML se puede utilizar una herramienta para aplicar dicha transformación cuando se accede al documento XML mediante un navegador Web.

Dicha herramienta se denomina Cocoon y sirve para publicar documentos XML en un entorno Web. Cocoon funciona dentro del servidor de Web Apache en forma de aplicación Java™ (servlet).

Su utilización es sencilla y se basa en guardar los documentos XML, DTD y XSL en un directorio `public_html` de vuestra cuenta UNIX.

### Ejemplo 3.11. Ejemplo 6

Se pueden utilizar los ficheros .xml, .xsl y .dtd asociados al nombre de fichero "`trabajo`", que se han descrito en los anteriores apartados.

Una vez se han almacenado en el directorio `public_html` habrá que modificar el fichero "`trabajo.xml`". Para ello se incluirá el siguiente código tras la declaración del tipo de documento:

```
<?xml-stylesheet type="text/xsl" href="trabajo.xsl"?> <?cocoon-process type="xslt"?>
```

Se puede comprobar el funcionamiento de Cocoon accediendo al fichero "`trabajo.xml`" en un servidor Web mediante un navegador y observando que el código HTML es el mismo que el obtenido en la fase de Procesamiento. Por ejemplo:

```
http://futura.disca.upv.es/~cuenta_usuario/trabajo.xml
```

Se puede visualizar el fichero "`trabajo.xml`" en un servidor Web mediante un navegador moderno como Mozilla (versión superior a 1.0), Netscape (versión superior a 6.2), ... y observando que el código HTML es el mismo que el obtenido en la fase de Procesamiento. Por ejemplo, lanzando la URL:

```
http://futura.disca.upv.es/~cuenta_usuario/trabajo.xml
```

y viendo el código fuente del documento recibido.

## Ejemplo 3.12. Ejercicio 6

Para publicar la propuesta de trabajo en cada una de las cuentas, lo primero que se debe hacer es confirmar (y si no crear), desde el directorio raíz de cada usuario, un directorio de nombre '`public_html`' donde guardar todos los archivos .xml, .xsl y .dtd.

Tanto el directorio raíz como el '`public_html`' deben tener permisos de lectura y ejecución para el resto del mundo y los ficheros en cuestión, permisos de lectura para este mismo grupo.

Para cambiar los permisos, desde el 'SSH Secure File Transfer Client', con el botón derecho se pueden cambiar las propiedades correspondientes tanto del directorio como de cada uno de los ficheros. También se pueden cambiar los permisos desde un terminal de Unix utilizando la orden '**chmod**'.

De forma resumida, estos son los pasos desde un terminal de Unix para crear la estructura mencionada:

```
$ mkdir public_html
$ chmod o+r,o+x
$ chmod o+r,o+x public_html
```

# Capítulo 4. Ejemplo avanzado: DocBook.

## Tabla de contenidos



En el capítulo anterior hemos visto como se utiliza el editor EditiX Lite Version para la creación de un documento XML sencillo. En este capítulo veremos como se utiliza XmlMind Personal Edition para redactar un documento escrito en XML pero que entrañe una mayor dificultad. Utilizaremos XmlMind Personal Edition para la creación de un libro. El libro tendrá varios capítulos, tabla de contenidos, índice de imágenes, índice de figuras, figuras, etc. En definitiva mostraremos la idoneidad del uso de la herramienta para una tarea especifica como es escribir un libro utilizando DocBook.

# 1. Uso general de XmlMind Personal Edition.

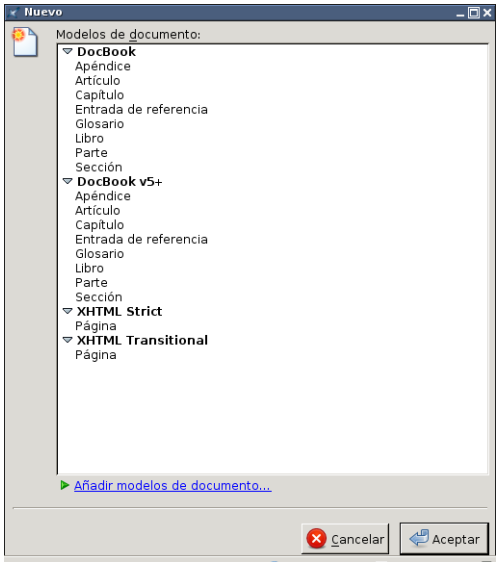
Lo primero que hay que tener claro a la hora de crear un documento DocBook con XmlMind Personal Edition es desterrar la idea de que editar con este programa, aun siendo en cierto modo, WYSIWYG, tiene cualquier tipo de parecido a la edición de un documento con un procesador de textos como Microsoft Word.

Principalmente hemos de partir de la idea de que lo que hace XmlMind Personal Editor es facilitar al usuario la edición de un documento permitiéndole editar el archivo de una manera mucho más amena que un editor XML común como es por ejemplo el EditiX Lite Version.

## 1.1. Crear documento.

Para crear un documento nuevo iremos a Archivo # Nuevo y seleccionaremos el tipo de documento que queremos crear.

### Figura 4.1. Nuevo documento en XmlMind Personal Edition



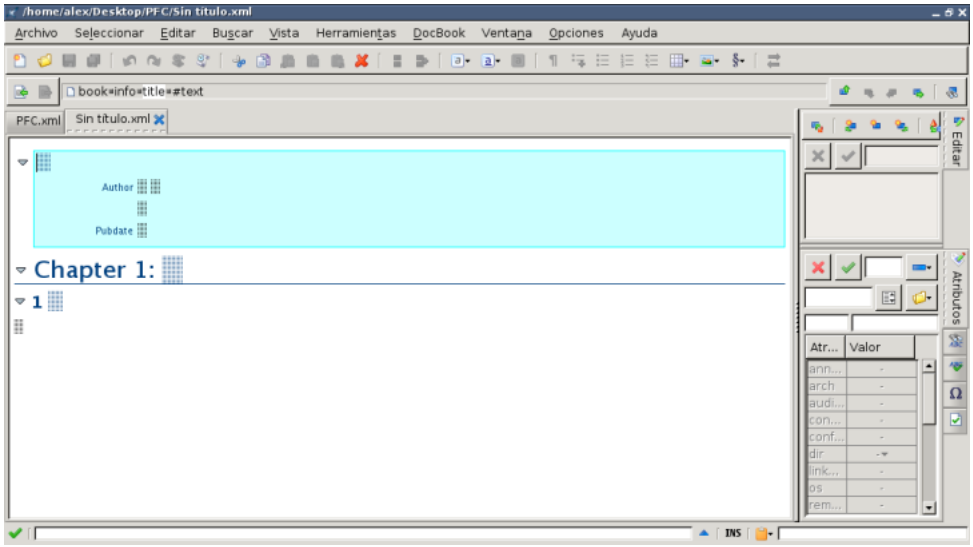

Para el caso que nos ocupa seleccionaremos DocBook v5 para poder crear un libro DocBook.

Podemos ver que XmlMind Personal Edition permite la creación de libros, artículos, glosarios, partes de libro y, además de documentos DocBook, permite crear documentos XHTML.

Un típico libro tiene titulo, autor, un prefacio y varios capítulos. Un libro también puede tener bibliografía, índices, glosarios y un colofón.

Una vez creado el libro, el editor nos muestra una plantilla sobre la que empezar a insertar contenidos como por ejemplo el nombre del libro, el nombre del autor, el título del capítulo.

### Figura 4.2. Edición de un libro




Un primer impulso podría ser el de comenzar a escribir de manera similar a como se haría en cualquier otro procesador de texto. Pero debemos tener en cuenta que estamos creando un libro utilizando DocBook que como ya sabemos no es ni más ni menos que una DTD escrita en XML que nos proporciona toda una lista de etiquetas y atributos que nos permite crear el libro.

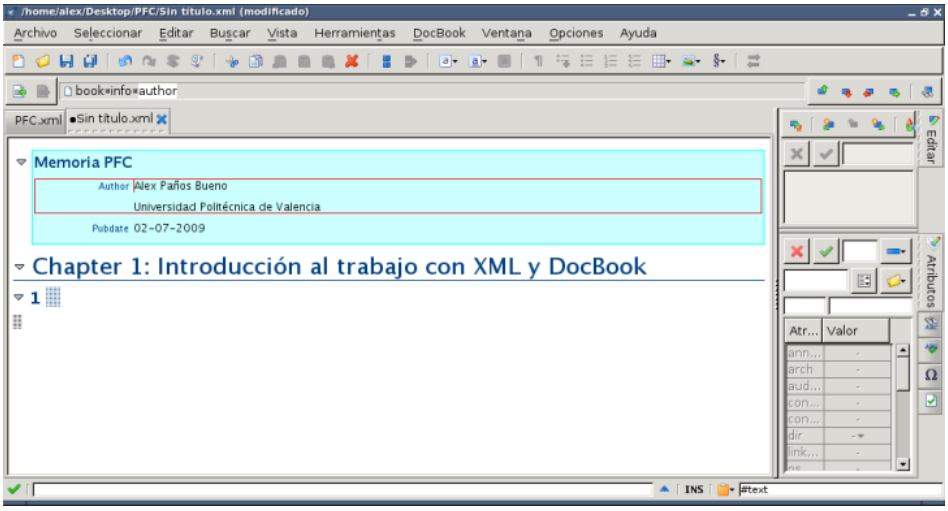
Para introducir texto en el documento deberemos situar el cursor sobre los indicadores.

### Figura 4.3. Indicador de inserción de texto


De este modo podemos completar el nombre del libro, autor, fecha de publicación y título de capítulo.

El documento quedaría así:

### Figura 4.4. Añadir datos de autor



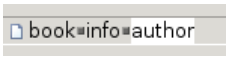

## 1.2. Seleccionar elementos.

En XmlMind Personal Edition podemos seleccionar elementos de tres formas:

1. La selección de texto normal, que es la selección de texto que podemos encontrar en cualquier editor. Puedes hacer click sobre la U en Universidad Politécnica y arrastrar el puntero para seleccionar el texto.

2. La selección de un nodo, con la que podemos elegir uno o varios de estos nodos. Para seleccionar un nodo la manera mas sencilla que tenemos es ir al NodePath:

   ### Figura 4.5. NodePath

   En este ejemplo al seleccionar el elemento author veremos que tanto author como los nodos hijos son seleccionados. La selección de un nodo se indica mediante un fino borde rojo que lo envuelve.

   Otra manera de elegir un nodo es hacer click sobre texto, si hacemos click sobre la etiqueta author, veremos que el elemento se selecciona de la misma manera que si lo seleccionáramos directamente del NodePath.

3. La selección implícita de elementos. El elemento que tiene el cursor es seleccionado de forma implícita. No hace falta hacer nada en especial para aplicar comandos sobre un elemento seleccionado implícitamente.

## 1.3. Añadir elementos.

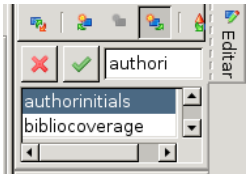
Pero, ¿y si queremos añadir alguna información extra como por ejemplo las iniciales del autor?. Para ello deberemos situar el cursor sobre la palabra autor, veremos que un borde rojo rodeará toda la etiqueta autor y los elementos que se anidan bajo autor y veremos también bajo la barra de herramientas la barra (NodePath) que nos muestra que el elemento author ha sido seleccionado. El NodePath muestra donde se encuentra el cursor en todo momento.

### Figura 4.6. NodePath


Una vez seleccionado el elemento author iremos a Editar # Insertar después... y escribiremos en la caja de edición situada a la derecha authorinitials.

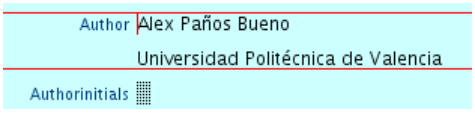
Conforme vayamos escribiendo nos irá apareciendo una lista de elementos que se pueden utilizar bajo la etiqueta author.

### Figura 4.7. Añadir authorinitials




Una vez seleccionado nos aparecerá authorinitials bajo la etiqueta author y podremos escribir las iniciales.

### Figura 4.8. Añadir iniciales




En el menú Editar tenemos Insertar antes..., Insertar... e Insertar después.... Esta seria la mecánica de trabajo con XmlMind Personal Edition. Iremos añadiendo, eliminando y cambiando elementos mediante Insertar, Borrar o Cambiar del menú Editar. No se trata de escribir conforme vayan viviendo las ideas, el autor ha de tener bien claro qué quiere escribir y qué estructura debe tener el documento que quiere escribir.

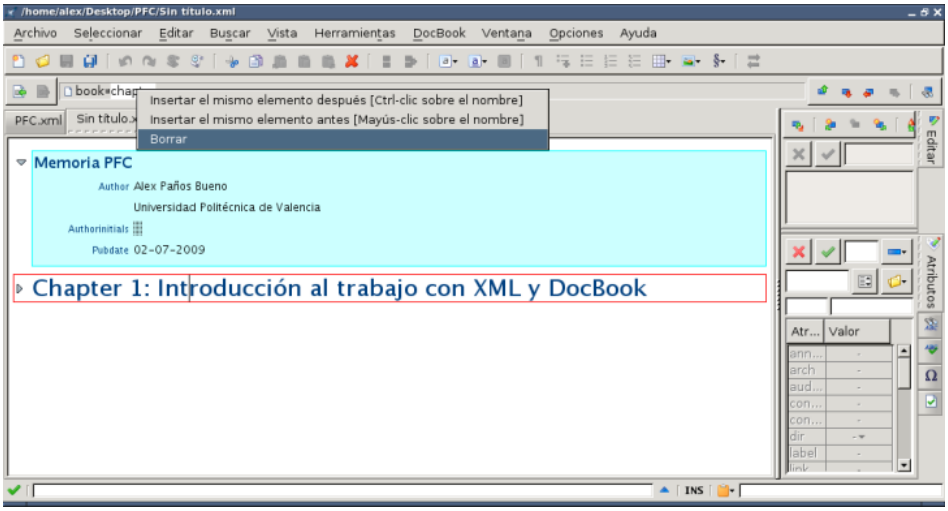
## 1.4. Eliminar elementos.

El proceso para eliminar un elemento del libro es idéntico al proceso de añadir elementos. Podemos eliminar el elemento directamente seleccionándolo y con el botón derecho del ratón haciendo click en Borrar , yendo a Editar # Borrar, o seleccionándolo en el NodePath y con el botón derecho seleccionando Borrar.

A la hora de eliminar elementos deberemos tener cuidado ya que al eliminar un elemento padre también eliminaremos los elementos que contiene y cuelgan de el. XmlMind Personal Edition permite tener una vista jerárquica del documento que está editando. Esta vista es muy útil puesto que muestra de una manera estructurada que elementos afecta la eliminación de un nodo padre.

Por ejemplo si quisiéramos eliminar el capítulo 1 puedes hacer click sobre cualquier palabra del titulo del capítulo seleccionando el elemento título, con lo que el NodePath mostrará book->Chapter->title->#text. Ahora simplemente se puede hacer click sobre la palabra Chapter y con el botón secundario del ratón seleccionar Borrar. De esta manera desaparecerá el nodo capítulo y todos los nodos que de el dependen.

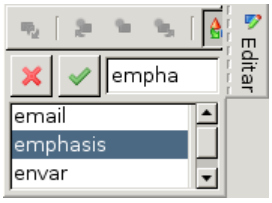
**Figura 4.9. Eliminar un nodo**




## 1.5. Convertir elementos.

De la misma manera que podemos insertar y eliminar elementos también podemos convertir unos elementos en otros siempre que esta conversión sea posible. Podemos por ejemplo convertir un texto sin formato a emphasis. Para ello seleccionaremos el texto que queremos convertir, por ejemplo la palabra XML e iremos a Editar # Convertir... y, en la caja de edición empezar a escribir emphasis hasta que aparezca y seleccionarlo.

**Figura 4.10. Conversión de elementos**




La conversión de elementos es muy útil. Podemos seleccionar por ejemplo una dirección web, www.java.com, Editar # Convertir... o botón derecho del ratón Convertir y seleccionar ulink.

# 2. Caso real: Edición del Proyecto.

En este capítulo vamos a mostrar la potencia del XmlMind Personal Edition para la creación de este mismo proyecto. Una vez hemos visto el uso general del editor, como se seleccionan, insertan, eliminan y convierten elementos vamos a ver casos mas específicos que muestran la potencia de la edición de DocBook con XmlMind Personal Edition. Crearemos capítulos, secciones, insertaremos, imágenes, tablas, enlaces web, trozos de código, etc.

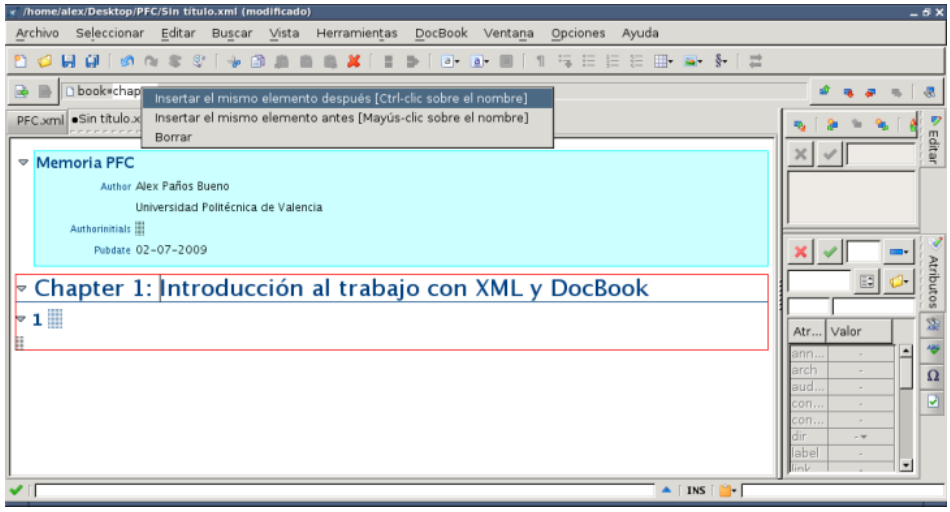
Un típico libro consta de una portada, dedicatoria, capítulos, secciones, párrafos, índices de tablas, de figuras, de términos, etc.

Veremos como incluir cada uno de estos elementos.

## 2.1. Capítulos.

Vamos a añadir un segundo capítulo al libro que hemos abierto anteriormente. Un capítulo tiene como elementos un título y una o varias secciones. Para ello debemos seleccionar el primer capítulo ya sea seleccionando la etiqueta chapter 1 y seleccionando Editar # Insertar después..., o seleccionado la etiqueta chapter desde el NodePath y con el botón derecho del ratón seleccionando Insertar el mismo elemento después.

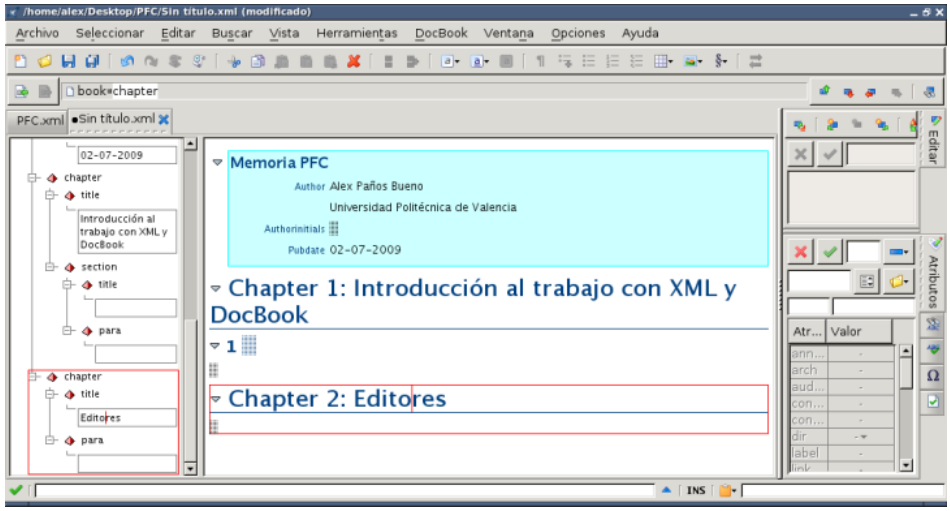
### Figura 4.11. Añadir Capitulo




Tras añadirlo nos aparecerá un segundo capítulo en el libro que estamos editando. Si abrimos la vista sin hoja de estilo podremos ver la estructura en forma de árbol del documento.

Utilizando esta vista se puede navegar por todo el libro con la ventaja de saber en todo momento en que lugar nos encontramos. En este momento es sencillo saber donde estamos, pero conforme se van añadiendo etiquetas , imágenes, secciones y demás elementos, resulta mas complicado ubicarse en la estructura del documento.

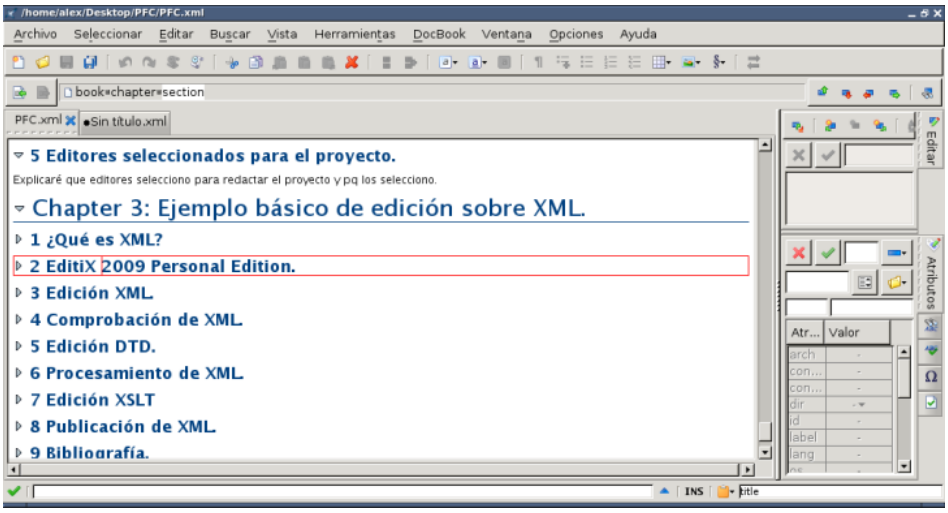
### Figura 4.12. Vista en Árbol




## 2.2. Secciones.

Un capítulo de un libro puede dividirse en diferentes secciones, una sección divide conceptualmente un capítulo en diferentes partes. Las secciones se añaden de la misma manera que los capítulos. Únicamente deberemos seleccionar un capítulo e insertar una sección. En la siguiente captura puedes ver un capítulo del libro con sus respectivas secciones.

### Figura 4.13. Añadir Sección




La hoja de estilo de DocBook se encargará de numerar tanto los capítulos como las secciones que vayamos añadiendo.

## 2.3. Imágenes

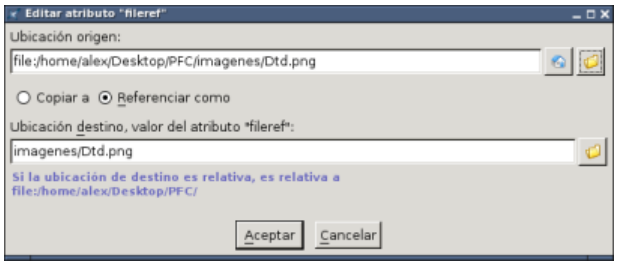
A lo largo del documento han sido utilizadas diversas imágenes. La etiqueta utilizada es figure, si no quisiéramos que apareciera una lista de imágenes utilizaríamos la etiqueta informalfigure.

Para añadir una imagen iremos a Editar # Insertar después... , en la caja de edición escribiremos figure y seleccionaremos la etiqueta. Nos aparecerá un icono en el que haremos doble click

### Figura 4.14. Añadir Imagen


y nos aparecerá un cuadro de dialogo para poder insertar una imagen.

### Figura 4.15. Cuadro Imagen




## 2.4. Tablas

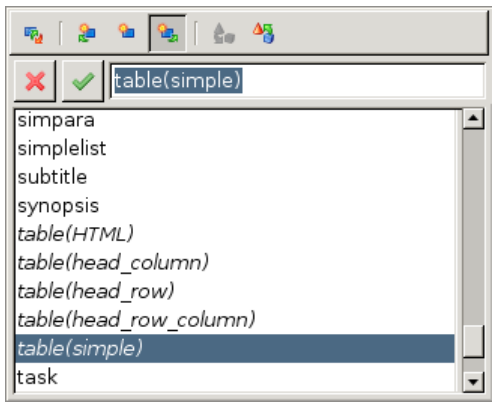
Es muy sencillo añadir una tabla en un libro escrito en DocBook. La sintaxis a utilizar para crear una tabla es muy similar a la de HTML pero al contrario que con HTML no podemos utilizar una tabla en DocBook para maquetar el contenido.

- Tabla – tbody

- Fila – row
- Celda – entry

Para crear una tabla iremos a Editar # Insertar después... y empezaremos a escribir la etiqueta table seleccionando de la caja de edición table(simple).

### Figura 4.16. Añadir Tabla




Nos aparecerá una tabla como la de la imagen en la que podremos empezar a introducir datos además del título de la tabla.

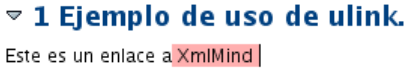
### Figura 4.17. Edición de tabla


## 2.5. Enlaces Web

Como ya ha sido explicado en el apartado de Conversión de elementos, la manera mas sencilla de crear un enlace seria la selección del texto e ir a Editar # Convertir... y seleccionar ulink.

Vamos a crear un enlace a la web de XmlMind.

### Figura 4.18. Selección texto del enlace




Una vez seleccionado el texto iremos a Editar # Convertir... y seleccionaremos en la caja de edición la etiqueta ulink. El texto seleccionado mostrará el formato típico de un hiperenlace.

### Figura 4.19. Añadir enlace


Únicamente nos faltará indicar la dirección web de la página web de XmlMind Personal Edition, para ello debemos ir a la caja de atributos, seleccionar el atributo url y copiar la dirección.

**Figura 4.20. Edición de ulink**


## 2.6. Código.

La inserción de código en DocBook con XMLMInd Personal Edition es muy sencillo, Únicamente deberemos copiar el código que necesitemos:

<?xml version="1.0" encoding="ISO-8859-1"?>

<Trabajo>

<Titulo> Práctica de XML </Titulo>

<Grupo>

<Componente>

<Nombre>Manuel Agustí</Nombre>

</Componente>

<Componente>

<Nombre>Vicente Atienza</Nombre>

</Componente>

<Componente>

<Nombre>J.Vicente Benlloch</Nombre>

</Componente> <Componente>

<Nombre>Félix Buendía</Nombre>

</Componente> </Grupo>

<Resumen>

<Parrafo>El trabajo consiste en explicar un lenguaje como XML</Parrafo>

<Parrafo>Para ello, se describirá la edición de documentos XML</Parrafo>

<Parrafo>A continuación se comprobará la validez de dichos documentos</Parrafo>

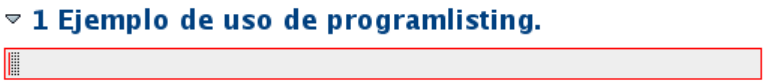
<Parrafo>También se verán las opciones para procesar documentos XML</Parrafo>

</Resumen>

</Trabajo>

La manera mas sencilla de añadir el código sería: Editar # Insertar después... y seleccionar programlisting en la caja de edición. Aparecerá una región coloreada para que escribamos el código.

### Figura 4.21. Añadir programlisting




Podemos directamente pegar el código desde un editor de textos común. La apariencia del código pegado sería la siguiente.

### Figura 4.22. Aspecto de programlisting

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <Trabajo>
  <Titulo> Práctica de XML </Titulo>
  <Grupo>
   <Componente>
    <Nombre>Manuel Agustí</Nombre>
   </Componente>
   <Componente>
    <Nombre>Vicente Atienza</Nombre>
   </Componente>
   <Componente>
    <Nombre>J.Vicente Benlloch</Nombre>
   </Componente>
   <Componente>
    <Nombre>Félix Buendía</Nombre>
   </Componente>
  </Grupo>
  <Resumen>
    <Parrafo>El trabajo consiste en explicar un lenguaje como XML</Parrafo>
    <Parrafo>Para ello, se describirá la edición de documentos XML</Parrafo>
    <Parrafo>A continuación se comprobará la validez de dichos documentos</Parrafo>
    <Parrafo>También se verán las opciones para procesar documentos XML</Parrafo>
  </Resumen>
 </Trabajo>
↵
```

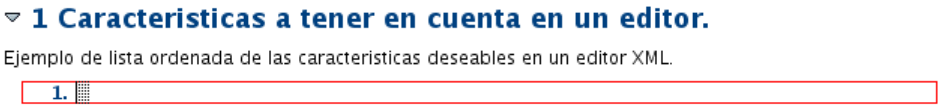
## 2.7. Listas.

Al igual que en HTML podemos incluir tres tipos de listas en DocBook.

- Itemizedlist: Una lista en la que los elementos se muestran junto un pequeño circulo u otro símbolo.
- Orderedlist: Una lista en la que cada elemento se ordena secuencialmente.
- Variablelist; Una lista en la que cada termino tiene asociado una descripción.

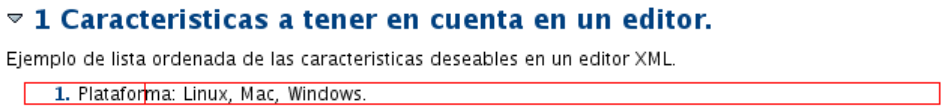
Crearemos una lista ordenada de elementos para enumerar las características a tener en cuenta en un editor XML. El mecanismo es el de siempre, pinchas en el Párrafo que contiene “Ejemplo de lista ordenada..” y vas a editar Editar # Insertar después... y seleccionas orderedlist de la caja de edición.

### Figura 4.23. Añadir orderedlist



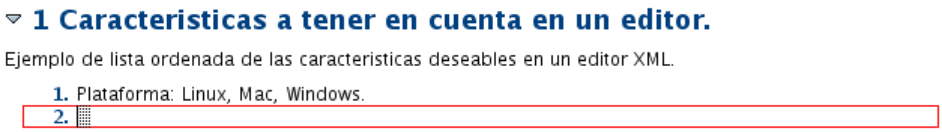

Se puede escribir la primera característica.

### Figura 4.24. Primer elemento de orderedlist




Siguiendo la forma habitual de añadir elementos para añadir mas características se debe seleccionar el elemento listitem, que como se puede ver arriba es rodeado por un cuadro de color rojo, e ir a editar insertar después, la única opción que saldrá en el cuadro de edición será itemizedlist, se selecciona y se crea un segundo elemento.

### Figura 4.25. Segundo elemento de orderedlist

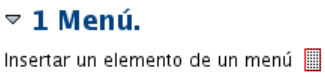



## 2.8. Menú.

Como indicamos al principio del libro, DocBook se utiliza principalmente para la creación de documentación técnica como son por ejemplo los manuales de software. Por ello DocBook cuenta con etiquetas tanto para mostrar código, como para mostrar menús de software. Esta etiqueta es muy útil y es utilizada a lo largo de este libro en numerosas ocasiones.

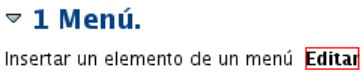
Para mostrar un menú, contamos con la etiqueta menuchoice. Para insertar un menú debemos ir a Editar # Insertar... y seleccionar la etiqueta menuchoice.

### Figura 4.26. Añadir menuchoice



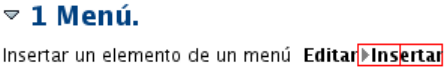

Escribiremos la palabra editar en la caja de edición. XmlMind Personal Edition toma la palabra Editar como guiicon, nosotros lo convertiremos a guimenu.

### Figura 4.27. Conversión de guiicon a guimenu




Ahora añadiremos el ítem del menú Insertar después, para ello seleccionaremos el elemento guimenu Editar # Insertar después... y le añadiremos desde la caja de edición un elemento guimenuitem Insertar:

### Figura 4.28. Añadir guimenuitem



# Capítulo 5. Conversión a otros formatos.

## Tabla de contenidos



Una vez tenemos el archivo XML o DocBook acabado tan solo nos falta aplicar las transformaciones que deseemos obtener. Podemos transformar un DocBook a HTML, a PDF, PS, XHTML, OpenOffice, etc.

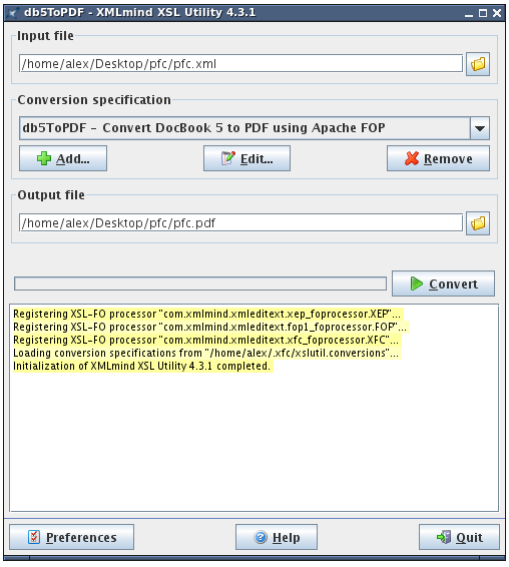
Al igual que en el caso de edición XML y DocBook nos vamos a centrar en analizar aquellos programas con un interfaz gráfico que facilite a los usuarios la transformación de sus archivos. Analizaremos XmlMind XSL Utility y DocMan.

## 1. XmlMind XSL Utility.

XmlMind XSL Utility es una aplicación que permite la transformación de archivos DocBook a varios formatos. El programa se puede descargar de http://www.xmlmind.com/foconverter/downloadperso.shtml. Para poder utilizarlo debemos disponer de una maquina virtual de java con una version superior o igual a 1.5+. Para su instalación debemos bajar el archivo `xslutil_perso-4_3_1.zip`, descomprimirlo y en caso de utilizar MSWindows ejecutar `xslutil.exe` o en caso de utilizar GNU/Linux dar permisos de ejecución a `xslutil.sh` y ejecutarlo desde una terminal **./xslutil.sh** o desde un gestor de archivos como nautilus. Hemos utilizado la versión de GNU/Linux.

La apariencia del programa tras arrancarlo es la siguiente:

**Figura 5.1. XmlMind XSL Utility**



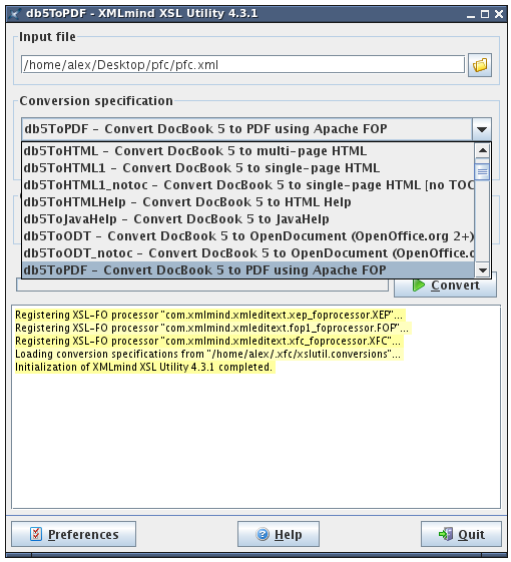

XMLmind XSL Utility permite convertir DocBook 4, DocBook 5, DocBook simplificado, DocBook Slides, y XHTML a: PDF, RTF, WordprocessingML, Office Open XML, OpenOffice, HTML, Ayuda de Eclipse, Ayuda HTML y Ayuda Java.

XMLmind XSL Utility utiliza dos procesadores XSLT: Saxon 6 y Saxon 9, y 3 procesadores XSL-FO: XMLmind XSL-FO Converter, Apache FOP, y RenderX XEP aunque este procesador no viene incluido en la aplicación pero

se puede descargar y utilizar una versión de prueba [http://www.renderx.com/download/] o una personal Edition [http://www.renderx.com/download/personal.html].

Para poder transformar un documento XML en otro formato debemos añadir el archivo a transformar seleccionándolo dando doble click al icono de la carpeta junto al input file. Una vez seleccionado el archivo a transformar podemos elegir el formato de salida desde el combo de selección:

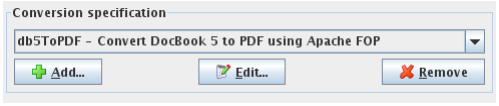
## Figura 5.2. Seleccionar transformación




Ahora tan solo debemos hacer click en Convert y esperar a que la conversión termine.

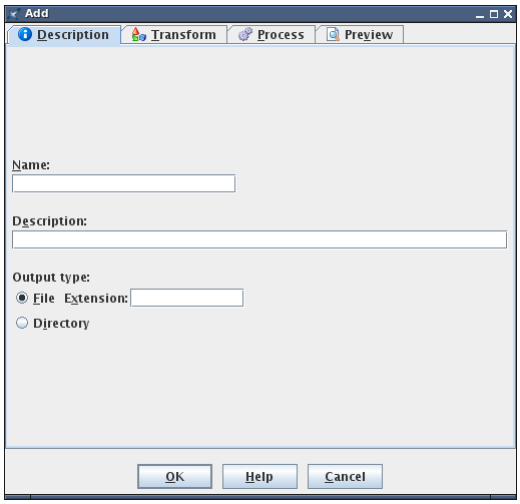
Este programa permite añadir nuevas conversiones y editar las existentes. Para añadir una nueva conversión debes hacer click en el botón Add del apartado Conversion specification:

## Figura 5.3. Añadir transformación



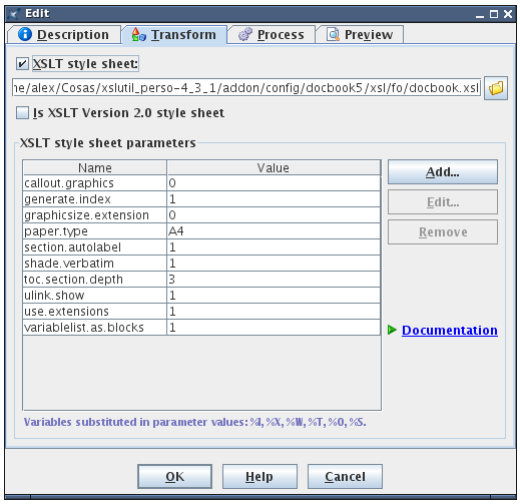

Aparecerá una nueva ventana que nos permitirá añadir una nueva conversión. En la pestaña Description deberemos darle un nombre, una descripción y una extensión de fichero.

## Figura 5.4. Pestaña Description



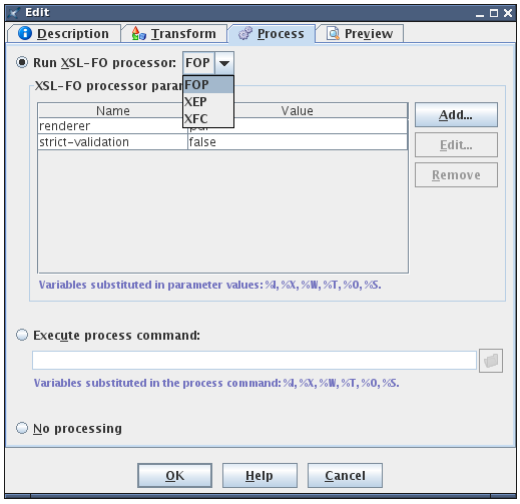
En la pestaña Transform podemos seleccionar la hoja de estilo XSLT a utilizar por el procesador y añadir o editar parámetros individuales. Si haces click en el enlace Documentation de la pestaña, se abrirá la página DocBook XSL Stylesheets: Reference Documentation [http://docbook.sourceforge.net/release/xsl/current/doc/index.html] que contiene todos los parámetros que se pueden utilizar para personalizar la salida del archivo XML. En esta imagen vemos la pestaña Transform del procesador Apache FOP para convertir a PDF.

## Figura 5.5. Pestaña Transform



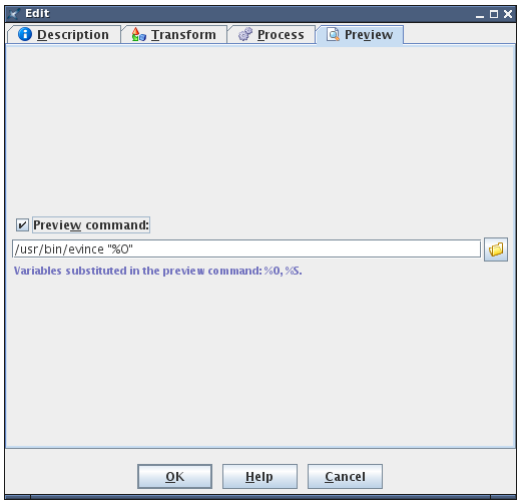

La siguiente pestaña es la de Process. En esta pestaña elegimos el procesador FO, en este caso el de Apache, pero también disponemos del procesador XFC de XmlMind y XEP de RenderX que como hemos comentado anteriormente viene preinstalado por lo que es necesario bajar una versión de prueba o una Personal Edition para poder utilizar. En esta pestaña encontramos también una serie de parámetros que podemos añadir o editar y la posibilidad de añadir comandos de proceso.

## Figura 5.6. Pestaña Process




La ultima pestaña que tenemos es la de Preview. En esta pestaña podemos definir el programa que utilizaremos para visualizar el contenido e la transformación. En este caso podemos seleccionar un programa que visualice archivos PDF. Como estamos utilizando GNU/Linux seleccionamos el visor de documentos evince para que abra el PDF tras finalizar la transformación.

**Figura 5.7. Pestaña Preview**




# 2. DocMan.

DocMan (DocBook Toolchain Manager), es una pequeña herramienta cuya utilidad es convertir archivos DocBook-XML en otros formatos como HTML, XHTML, CHM y PDF. Su autor es Lars Trieloff y se puede conseguir el programa en http://www.goshaky.com/docman-distfiles/DocMan/.

Hemos utilizado la versión para GNU/Linux, pero en la página web del autor podemos encontrar también una versión para MSWindows.

Para poder utilizar DocMan es necesario tener instalado una versión de la máquina virtual de java superior a la versión 1.4.

La instalación del programa es sencilla. En caso de utilizar MSWindows deberemos bajar el ejecutable `DocMan-0.99d-Installer.exe` o `DocMan-0.99c-Installer.exe` e instalarlo. Si utilizamos GNU/Linux debemos bajar `docman-0.99c-bin.zip` o `docman-0.99d-bin.zip`, descomprimirlo, ir a la carpeta que contiene el archivo `docman.sh` y darle permisos de ejecución. Para dar permisos de ejecución utilizaremos el siguiente comando:

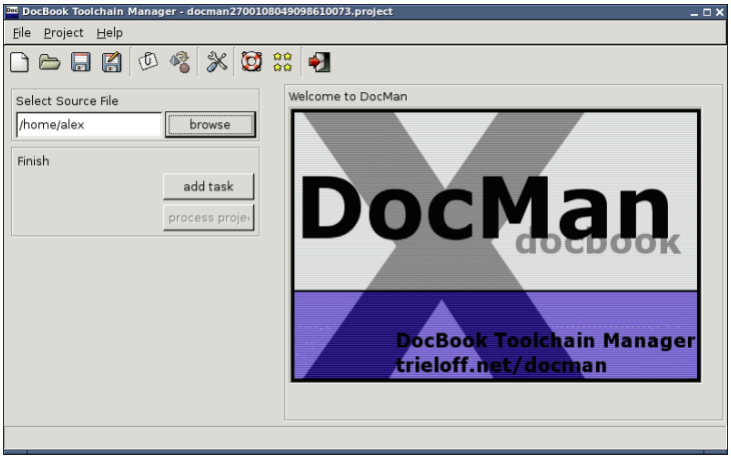
**chmod +x /home/alex/Desktop/DocMan/docman.sh**

Una vez concedido el permiso de ejecución pasaremos a ejecutarlo con el siguiente comando dentro de la carpeta /home/alex/Desktop/DocMan/:

**./docman.sh**

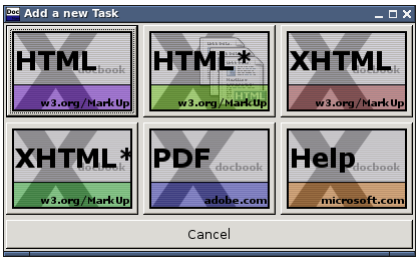
La apariencia del programa tras arrancarlo es la siguiente:

**Figura 5.8. DocMan**



Para transformar un documento iremos a browse, seleccionaremos el archivo DocBook a transformar y haremos click sobre el botón add task. Nos aparecerá una ventana en la que seleccionaremos el tipo de salida que queremos obtener:

### Figura 5.9. Selección de transformación

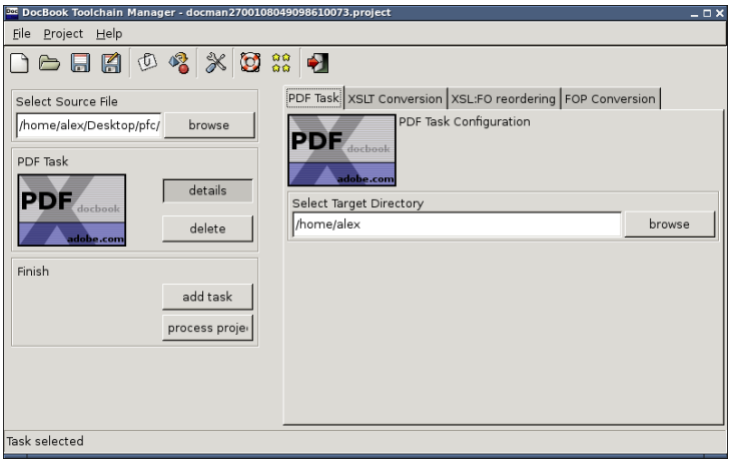



Podemos seleccionar los siguientes tipos de salida:

- HTML. En una sola página.
- HTML. Dividido en diversos ficheros.
- XHTML. En una sola página.
- XHTML. Dividido en diversos ficheros.
- PDF.
- Microsoft HTML Help.

Seleccionaremos PDF como salida del documento DocBook. Una vez seleccionado PDF nos aparecerá la tarea PDF, Si hacemos click en el botón details nos aparecerán 4 pestañas.

### Figura 5.10. Tarea PDF




En la primera pestaña PDF Task haciendo click en browse puedes seleccionar el directorio donde guardar el archivo PDF.

Las otras tres pestañas se utilizan para añadir o editar parámetros del procesador. Puedes editar los parámetros de esta pestaña seleccionando el parámetro y pulsando a la tecla Enter. Para añadir un parámetro deberás hacer click en Add Parameter.

Una vez seleccionado el directorio de salida de la transformación tan solo debemos hacer click en process project. La tarea añadida comenzará a procesarse y tardará mas o menos tiempo en completarse dependiendo del tamaño del archivo DocBook y de la potencia del ordenador que procese la tarea.

# 3. Personalización de la conversión.

XmlMind XSL Utility ha sido el software seleccionado para llevar a cabo la transformación de la memoria del proyecto a un documento HTML y PDF.

Hemos encontrado dos problemas a la hora de llevar a cabo la transformación de la memoria.

1. Tamaño de las imágenes diferente dependiendo si transformas a PDF o a HTML.

   Al transformar a HTML las imágenes se visualizan de una forma correcta. El problema viene cuando se lleva a cabo la transformación a PDF.

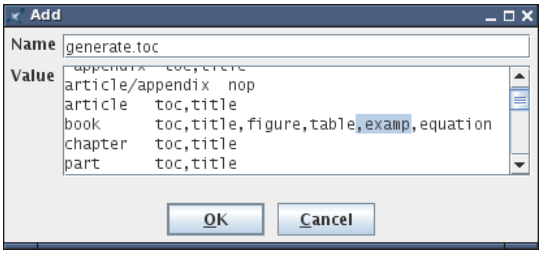
   La solución al problema consistiría en definir un tamaño de imagen diferente para la transformación HTML y la transformación PDF. Si se visualiza con XmlMind Personal Edition el archivo XML de la memoria se puede ver que las imágenes se repiten dos veces, una con un mayor tamaño y una un poco mas pequeña. La solución consiste en añadir un segundo imageobject a figure. Un imageobject tendrá como atributo role el valor html y el otro image object tendrá como valor del atributo role el valor fo. De este modo haremos que el procesador utilice una imagen u otra dependiendo del tipo de transformación que queramos realizar. Para cambiar el tamaño de la imagen para PDF, iremos al imagedata dentro del imageobject definido con el atributo role fo y cambiaremos el atributo scale para escalar la imagen a la proporción que deseemos.

   Para que las imágenes se visualicen de una manera correcta en el archivo PDF hemos tenido que escalar las imágenes al 50% en la mayoría de los casos.

2. Creación automática de table of contents de ejemplos y tablas.

   En el capítulo 3 hemos utilizado el elemento example para los ejemplos y para los ejercicios ya que en DocBook no existe una etiqueta para los ejercicios. XmlMind XSL Utility genera tablas de contenidos por defecto para los ejemplos. El problema con el que nos encontramos es que se genera una lista de ejemplos bajo la que encontramos no solamente ejemplos si no también ejercicicios. Vamos a utilizar la opción que nos brinda XmlMind XSL Utility para poder pasar al procesador los parámetros necesarios para poder eliminar la generación automática de la lista de ejemplos. En la sección 1 de este capítulo hemos enseñado como añadir los parametros al procesador.

### Figura 5.11. Eliminar toc de los ejemplos




Eliminaremos la generación de la table of contents de ejemplos cuando se procesan documentos DocBook en forma de libro. Como se puede ver en la captura borramos el elemento example perteneciente a book.

Del mismo modo eliminaremos la generación automática de la lista de tablas ya que la memoria solo tiene la tabla comparativa de los editores.

# Capítulo 6. Conclusiones.

El objetivo de esta memoria es el de introducir al uso de herramientas para la creación de documentos XML y DocBook. A lo largo de los capítulos de esta memoria hemos pasado de editar un ejemplo sencillo de XML a saber como utilizar una herramienta para la creación de un articulo o un libro escrito en DocBook.

Para poder editar un archivo en XML sin utilizar ninguna hoja de estilo integrada como hemos hecho con Editix Lite Version debemos conocer mínimamente la sintaxis y las reglas de marcado de XML. Sin embargo si utilizamos XmlMind Personal Edition tan solo es necesario conocer como funciona el programa, como estructurar el documento y anidar las etiquetas. Son dos maneras diferentes de abordar la creación y edición de un documento.

Siguiendo los capítulos de la memoria, hemos trazado el camino que lleva desde la creación del documento hasta su posterior transformación. Hemos aprendido qué tipo de editor se adecua a una tarea u otra, qué son los analizadores o parsers y su uso, qué es que esté bien formado un documento y como solucionar los errores que puedan surgir al escribir un documento.

En el capítulo 5 hemos utilizado dos herramientas gráficas cuya utilidad es la de permitir la conversión de los documentos a otros formatos como HTML y PDF. De hecho esta memoria viene acompañada de la transformación de la misma a HTML y PDF utilizando XMLmind XSL Utility.

Hemos intentado cubrir la mayoría de los aspectos de edición en DocBook mediante el empleo de los ejemplos mas significativos como son la inclusión de imágenes, creación de capítulos, de listas, enlaces web, etc.

Escapa al alcance de esta memoria el análisis de todas los elementos que se pueden incluir en un libro puesto que para eso ya se dispone de la documentación original de DocBook en la que encontramos todos los elementos utilizables y, en que condiciones se puede hacer uso de ellos. Al igual que en la edición, también nos dejamos aspectos por analizar a la hora de llevar a cabo las transformaciones de los documentos. Las opciones que podemos pasar a los procesadores para que cambien el aspecto de la transformación son, al igual que en el caso de los elementos de Docbook, muy numerosos y se pueden encontrar en la documentación de referencia de las hojas XSL de Docbook.

La intención original era la de proporcionar una memoria que fuera útil en general para cualquier persona interesada en XML y DocBook y en especial para todos aquellos estudiantes que en sus asignaturas se encuentran por primera vez con XML. El capitulo 3 sería un punto de partida interesante para familiarizarse con todos los programas involucrados en la creación de un archivo XML. EL capítulo 4 puede ser un punto de partida interesante para todo aquel que quiera empezar desde cero la creación de un libro en DocBook.

# Bibliografía

Norman Walsh y Leonard Muellner. *DocBook.* The Definitive Guide. 1999. 1. ISBN: 156592-580-7. 648. Una versión On-Line está disponible [http://www.oasis-open.org/docbook/documentation/reference/html/docbook.html].

Lars Marius Garshol. *Catálogo de herramientas XML [http://www.garshol.priv.no/download/xmltools/]*. 16 Noviembre 1999.

Elliotte Rusty Harold. *About XML [http://www.cafeconleche.org/]*.

Juan Julian Merelo. *Generación de páginas Web usando XSLT y XML [http://geneura.ugr.es/~jmerelo/XSLT/]*.

O'Reilly. *XML Editors [http://www.xml.com/pub/rg/XML_Editors]*.

DocBook Wiki. *DocBook Wiki [http://wiki.docbook.org/topic/FrontPage]*.

Dave Pawson. *Docbook Basics and References [http://www.dpawson.co.uk/docbook/reference.html]*.

Wikipedia. *DocBook [http://en.wikipedia.org/wiki/DocBook]*.

Bob Stayton. *DocBook XSL: The Complete Guide [http://www.sagehill.net/docbookxsl/]*. 2007. 4.

Joe Brockmeier. *A gentle guide to DocBook [http://www.ibm.com/developerworks/library/l-docbk.html]*. 2000.

# Índice