DocBook

Unai Aguilera

gkalgan @ gmail . com

Cursillos de Julio 2006

Introducción

- DocBook es un lenguaje de etiquetas para elaborar documentación técnica.
  - Fue creado para elaborar documentación relativa a hardware y software pero puede utilizarse para cualquier otro tema.
  - La principal característica de DocBook es que utiliza un lenguaje neutral para especificar el documento separándolo de la presentación final.
    - Se puede general HTML, PDF, páginas man, etc, sin necesidad de cambiar el fichero fuente.

Introducción

- DocBook comenzó como un proyecto de las empresas HAL Computer Systems y O' Reilly Associates en 1991.
  - En 1998 fue adoptado por el SGML Open Consortium,
- Actualmente es un estándar mantenido por el DocBook Technical Committee del OASIS Consortium.
  - OASIS es Organization for the Advancement of Structured Information Standards (www.oasis-open.org)

Introducción

- DocBook esta disponible tanto en SGML como en XML.
  - SGML (Standard Generalized Markup Language)
    - Es un metalenguaje para definir lenguajes de marcado de documentos. Es complicado.
    - Disponible como DTD (Document Type Definition).
  - XML (Extensible Markup Languaje)
    - Es un lenguaje de marcado que permite definir otros lenguajes de marcado.
    - Es un subconjunto simplificado de SGML.
    - Disponible como DTD y XML Schema.

Introducción

- En nuestro caso vamos a utilizar DocBook definido mediante XML Schema.
  - XML Schema
    - Permite expresar el conjunto de reglas que debe cumplir un documento XML para que se le considere del tipo determinado por el esquema.
  - El XML Schema de DocBook define las etiquetas y la estructura de un documento de este tipo.
    - Cualquier documento que no respete estas reglas no será considerado un documento DocBook válido.

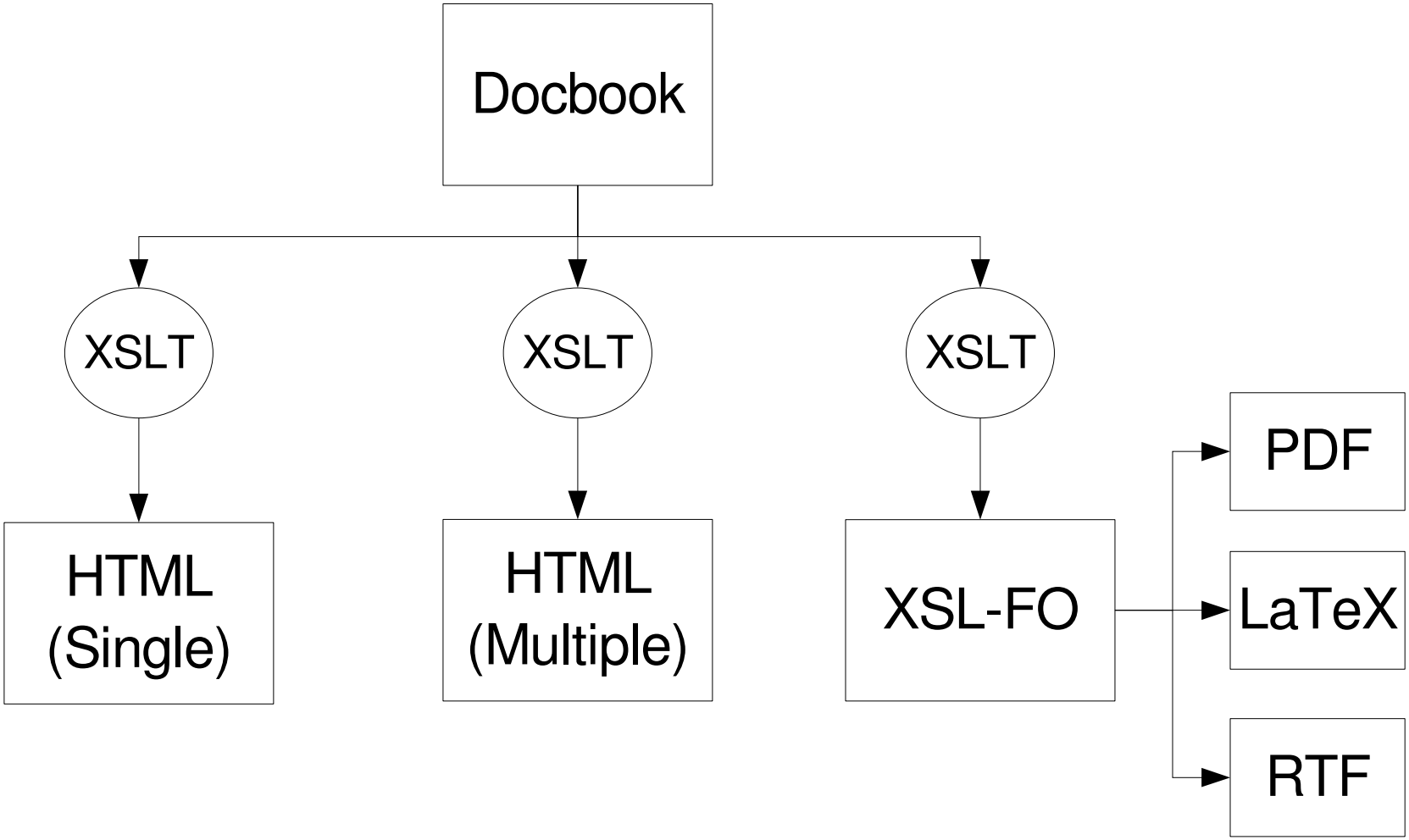
Introducción

- Por lo tanto escribir un documento DocBook es similar a escribir HTML pero con otras etiquetas y reglas.
- ¿Y después de escribir el documento DocBook?
  - Es necesario realizar un proceso para generar el documento.
    - En HTML el navegador "**hace algo**" para pintar la página.
  - En DocBook este proceso se realiza utilizando hojas de estilo llamadas XSL.

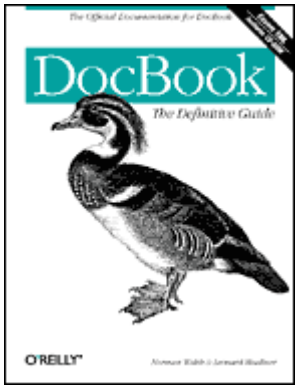
Introducción

- XSL (eXtensible Stylesheet Languaje)
  - Es un lenguaje que permite definir como se debe transformar otro fichero XML.
    - Existen varios tipos
      - **XSLT:** lenguaje XML para transformar otros documentos XML.
      - **XSL-FO:** lenguaje XML para especificar el formateo visual de otros documentos XML.
      - **XPath:** lenguaje no XML que permite acceder a partes de documentos XML.
    - Las transformaciones para DocBook se realizan utilizando hojas XSTL y XSL-FO

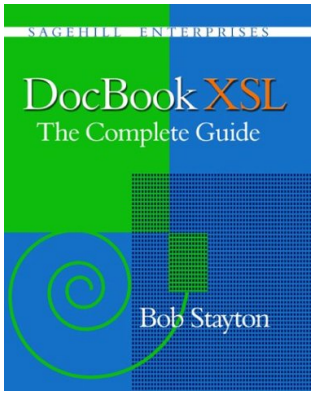
Conversión a otros formatos



Referencias imprescindibles



DocBook: The Definitive Guide
Norman Walsh, Leonard Muellner
http://www.docbook.org/tdg/



DocBook XSL: The Complete Guide
Bob Stayton
http://www.sagehill.net/docbookxsl/

Introducción

- Un ejemplo rápido

Sintaxis XML

- DocBook XML es un dialecto XML, con lo que hay que tener muy presentes las normas de sintaxis de XML.
- En XML una etiqueta empieza por el símbolo ‘menor que’ < y termina con el símbolo ‘mayor que’ >
- Justo después del símbolo < se encuentra el nombre de la etiqueta.

```
<book>    <chapter>    <section>
```

Sintaxis XML

- Hay dos tipos de etiquetas:
  - *Non-Empty* o ‘contenedora’: Contienen texto o más etiquetas. Tiene que haber una *etiqueta de comienzo* y una *etiqueta de fin*. En la etiqueta de fin, el símbolo < siempre va seguido del símbolo /

```
<book>
...
...
</book>
```

Sintaxis XML

- *Empty* o vacía: No contienen nada. El símbolo > debe ir *siempre* precedido del símbolo /

```
<imagedata fileref="imagen.png" />
```

También se admite la forma:

```
<imagedata fileref="imagen.png"></imagedata>
```

Sintaxis XML

- Las etiquetas pueden tener atributos. Estos atributos indican opciones de la etiqueta.

```
<imagedata fileref="imagen.png" />
```

- Los atributos tienen la siguiente sintaxis:

nombre_atributo="valor"

- Si hay mas de un atributo, tienen que ir separados por espacios.

Sintaxis XML

- Toda etiqueta de comienzo debe tener etiqueta de fin.
- Las etiquetas no deben solaparse.
- Debe haber una única etiqueta en la raíz del documento (`<direcciones>`).
- Los valores de los atributos deben ir entre comillas (dobles o simples)
- Una etiqueta no puede tener dos atributos con el mismo nombre.

Estructura Básica

- El documento siempre debe empezar por:
  - Declaración XML
  - Declaración del tipo de documento

```
<?xml version="1.0"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
```

Estructura Básica (II)

- El elemento raíz de un documento DocBook es <book>
  - También podemos utilizar otros elementos raíz.

```
<book>
...
...
</book>
```

Estructura Básica

- La etiqueta <book> puede contener:
  - Metadatos (autor, copyright, etc.)
  - Prólogos
  - Apéndices
  - Bibliografías
  - Glosarios
  - Índices
  - etc.
- Los veremos en detalle más adelante.

Estructura Básica

- Un ejemplo de una etiqueta para añadir metadatos: <title>

```
<book>
  <title>Ejemplo Sencillo</title>
  ...
</book>
```

Estructura Básica (V)

- Añadimos capitulos con <chapter>
  - También pueden contener una etiq.<title>

```
<book>
  <title>Ejemplo Sencillo</title>
  <chapter>
    <title>Primer capitulo</title>
    ...
  </chapter>
</book>
```

Estructura Básica (VI)

- Dentro de los capítulos nos encontraremos con etiquetas de *bloque*. Por ejemplo, parrafos: <para>
  - Hay una gran cantidad de etiquetas de bloque: imagenes, tablas, etc.
- Los capítulos también pueden dividirse en secciones: <section>

Estructura Básica (VII)

```
<book>
  <title>Ejemplo Sencillo</title>
  <chapter>
    <title>Primer capitulo</title>
    <para>Primer parrafo</para>
    <para>Segundo parrafo</para>
  </chapter>
</book>
```

Índice

Generación Básica

- A partir del documento DocBook podemos generar documentos orientados a la presentación:
  - XHTML (un único fichero)
  - XHTML (un fichero por capítulo)
  - XSL-FO. A partir de este formato:
    - PDF
    - LaTeX
    - RTF

Generación Básica

- Esta conversión se hace con hojas de transformación XSLT.
  - No tenemos que escribirlas nosotros mismos.
  - Existen unas hojas de transformación prácticamente estándares:
    - http://docbook.sourceforge.net/
    - Debian: paquetes “docbook-xml” y “docbook-xsl”
  - Veremos que estas hojas son fácilmente parametrizables para ajustarlas a nuestras necesidades.

Generación Básica

- Una vez hayamos instalado las hojas de transformación, seguiremos dos pasos para generar un documento XHTML:
  - Validar el documento DocBook (paso opcional, pero recomendable)
  - Realizar la transformación

Generación Básica

- Para validar el documento podemos utilizar la utilidad “xmllint”
  - Parte de libxml: http://xmlsoft.org/
  - Debian: paquete libxml2-utils

```
xmllint --valid --noout fichero_docbook
```

  - Nos indicará si hay algún error en el documento (falta alguna etiqueta, errores de sintaxis, etc.)

Generación Básica

- Para realizar la transformación, podemos utilizar cualquier procesador XSLT:
  - Saxon: http://saxon.sourceforge.net/
  - Xalan: http://xml.apache.org/
  - xsltproc: http://xmlsoft.org/XSLT/
- Nosotros utilizaremos xsltproc

Generación Básica

- Para aplicar una transformación:

```
xsltproc \
  --output fichero_salida \
  fichero_XSLT \
  fichero_DocBook
```

- En Debian:

```
xsltproc \
  --output fichero_salida \
  /usr/share/xml/docbook/stylesheet/nwalsh/xhtml/docbook.xsl \
  fichero_DocBook
```

Generación Básica

- En general, siempre es válido lo siguiente:

```
xsltproc \
--output fichero_salida \
http://docbook.sourceforge.net/release/xsl/current/xhtml/docbook.xsl \
fichero_DocBook
```

- El procesador XSLT recuperará el fichero de Internet, o lo mapeará a un fichero local (si tenemos soporte para XML Catalogs)
  - Debian Sarge incluye soporte para XML Catalogs.

Etiquetas DocBook

- Las etiquetas en DocBook se dividen en las siguientes categorías:
  - Conjuntos (set)
  - Libros (book)
  - Divisiones (part)
  - Componentes (chapter)
  - Secciones (section)
  - Metadatos (author)
  - Bloques (para)
  - Elementos inline (citation)

Conjuntos

- Podemos definir varios libros en un único documento utilizando conjuntos:

```
<set>
    <book>
    ...
    </book>
    <book>
    ...
    </book>
</set>
```

Libros, Divisiones, Componentes (I)

- Un libro (<book>) se divide en...
  - Divisiones (<part>) que se dividen en...
    - Componentes
      - <preface>
      - <chapter>
      - <appendix>
      - <glossary>
      - <bibliography>
  - Un libro también puede dividirse directamente en componentes.

Libros, Divisiones, Componentes (II)

```
<book>
    <part>
        <preface></preface>
    </part>
    ...
    <part>
        <chapter></chapter>
        <appendix></appendix>
    </part>
</book>
```

Secciones (I)

- Un componente se divide en secciones:
  - <section> (pueden anidarse recursivamente)
  - <bridgehead> (sección no numerada)
- excepto <glossary>,<bibliography> e <index> que solo pueden dividirse en:
  - <glossdiv>
  - <bibliodiv>
  - <indexdiv>
- No pueden anidarse

Secciones (II)

```
<book>
     <title>Ejemplo Sencillo</title>
     <chapter>
          <title>Primer capitulo</title>
          <section>
               <title>Sección</title>
               <para>Primer parrafo</para>
               <para>Segundo parrafo</para>
          </section>
     </chapter>
</book>
```

Metadatos (I)

- Es posible agregar metadatos a...
  - <set>, <book>, <part>, <chapter>, <section>, <preface>, <appendix>, etc.
- mediante el tag <info>
- En estas etiquetas podemos incluir información sobre:
  - Autor/es, Copyright y otros avisos legales (licencias, etc.)
  - Lista de cambios (change log), etc.

Metadatos (II)

- Por ejemplo:

```
<book>
  <title>Ejemplo con Metadatos</title>
  <info>
    <author>
      <firstname>Pablo</firstname><surname>Perez</surname>
    </author>
    <copyright><year>2004</year></copyright>
   <legalnotice>
   <para>
   Este ejemplo esta disponible bajo los terminos de la GFDL
   </para>
   </legalnotice>
  </info>
  <chapter>bla, bla, bla... </chapter>
</book>
```

Bloques

- Las etiquetas de bloque son aquellas que, en la presentación, generalmente requieren un salto de lineas antes y/o después del bloque. Por ejemplo:
  - Listas
  - Ejemplos, figuras, tablas
  - Parrafos
  - Ecuaciones

Listas (I)

- <itemizedlist> y <orderedlist>
  - Hay más tipos de listas
- Tienen que contener etiquetas <listitem> que, a su vez, deben contener elementos de bloque (p.ej. parrafos)

Listas (II)

```
<itemizedlist>
     <listitem>
          <para>Primer item</para>
     </listitem>
     <listitem>
          <para>Segundo item (1)</para>
          <para>Segundo item (2)</para>
     </listitem>
</itemizedlist>
```

Ejemplos, Figuras, Tablas

- <example>, <figure>, <table>
  - Contienen ejemplos, figuras, y tablas.
  - Siempre tienen titulo (<title>)
  - Al convertir DocBook a un formato de presentación, estos bloques pueden flotar o permanecer donde están (dependerá de la conversión, no de DocBook).
- <informalexample>, <informalfigure>, <informaltable>
  - No tienen título y generalmente no flotan.

Figuras (I)

- Para insertar datos multimedia: <mediaobject>
- Puede contener: <imageobject>, <audioobject>, <videoobject>
- <imageobject> contiene <imagedata>
- Atributos de <imagedata>
  - **fileref**: URL de la imagen
  - Atributos para controlar tamaño de la imagen

Figuras (II)

```
<figure>
     <title>Una imagen</title>
     <mediaobject>
          <imageobject>
               <imagedata fileref="A.PNG"/>
          </imageobject>
     </mediaobject>
</figure>
```

Tablas (I)

- Las tablas en DocBook son muy versátiles, y están indicadas únicamente para datos que se presten a una presentación tabular.
  - No podemos utilizarlas para maquetar, como en HTML.
- La sintaxis de una tabla sencilla en DocBook es muy similar a la de HTML
  - Tabla ⇦ <tbody>
  - Fila ⇦ <row>
  - Celda ⇦ <entry>

Tablas (II)

```
<table>
     <title>Una tabla</title>
     <tbody>
          <row>
               <entry>A</entry>
               <entry>B</entry>
          <row>
          <row>
               <entry>C</entry>
               <entry>D</entry>
          <row>
     </tbody>
</table>
```

Miscelanea

- <qandaset> ➡ Lista de preguntas y respuestas
- <programlisting> ➡ Código fuente
- <screen> ➡ Contenidos de una consola de texto
- <blockquote> ➡ Citas textuales
- <equation> ➡ Ecuación matemática
- <procedure> ➡ Para describir procedimientos que consten de varios pasos (<step>)

Elementos en Linea (I)

- Las etiquetas *inline* se utilizan para anotar ciertas partes del texto, pero sin provocar un salto de linea antes o despues del texto etiquetado (a diferencia de los bloques).

- Las etiquetas *inline* generalmente sirven para añadir metadatos al propio texto del documento, aunque muchas veces también afectan a la presentación.

Elementos en Linea (II)

- Los típicos...
  - <emphasis> ⇦ énfasis (cursiva)
  - <footnote> ⇦ nota al pie de página
  - <quote> ⇦ cita textual (no bloque)

Elementos en Linea (III)

- Enlaces
  - <ulink url="..."> ⇦ Enlace con una URL
  - <link linkend="..."> ⇦ Enlace con otra parte del documento
    - El valor de "linkend" debe ser el valor del atributo "id" de la etiqueta con la que queremos enlazar
  - <xref linkend="..."> ⇦ Referencia a otra parte del documento.

Elementos en Linea (IV)

```
<chapter>
      <title>Primer capitulo</title>
      <section id="sec">
            <para>Primer parrafo</para>
            <para>Segundo parrafo</para>
      </section>
</chapter>
<chapter>
      <title>Segundo capitulo</title>
      <section>
            <para>
            <link linkend="sec">Enlace</link>
            </para>
      </section>
</chapter>
```

Elementos en Linea (V)

- Tenemos una *gran* cantidad de elementos inline para anotar el texto. Por ejemplo:
  - <foreignphrase> ⇦ Una frase en otro idioma
  - <returnvalue> ⇦ El valor devuelto por una función
  - <filename> ⇦ Denota el nombre de un fichero
  - <email> ⇦ Denota una dirección de e-mail
  - Muchos, muchos más.
- La mayoría de estos elementos no afectan a la presentación, pero aportan muchos metadatos al documento, facilitanto búsquedas, indexación, etc.

Índice

Generación Avanzada

- Podemos generar más tipos de documentos:
  - XHTML 'en cachos'
  - XSL-FO
    - LaTeX, PDF, RTF
- Podemos parametrizar la generación:
  - Pasando parámetros al procesador XSLT
  - Creando una hoja de transformación XSLT como extensión de una existente

Varios Documentos XHTML (I)

- Para generar varios documentos XHTML...
  - En Debian:

```
xsltproc \
/usr/share/xml/docbook/stylesheet/nwalsh/xhtml/chunk.xsl \
fichero_DocBook
```

  - En general:

```
xsltproc \
http://docbook.sourceforge.net/release/xsl/current/xhtml/chunk.xsl \
fichero_DocBook
```

Varios Documentos XHTML (II)

- El fichero “index.html” es el índice del documento generado.
- Por defecto:
  - Un índice
  - Una página por cada capítulo con el índice del capítulo y la primera sección.
  - Una página por cada sección.
  - Este comportamiento puede modificarse.

XSL-FO y PDF (I)

- Para generar XSL-FO...
  - En Debian:

```
xsltproc \
--output fichero_salida \
/usr/share/xml/docbook/stylesheet/nwalsh/fo/docbook.xsl \
fichero_DocBook
```

  - En general:

```
xsltproc \
--output fichero_salida \
http://docbook.sourceforge.net/release/xsl/current/fo/docbook.xsl \
fichero_DocBook
```

XSL-FO y PDF (II)

- Aunque existen visualizadores de XSL-FO, es más habitual convertirlo a otro formato más extendido.
  - XSL-FO ⇦ PDF
  - XSL-FO ⇦ LaTeX ⇦ PDF

XSL-FO y PDF (III)

- XSL-FO ⇒ PDF
  - Utilizando Apache FOP
    http://xml.apache.org/fop/
  - Programado en Java
  - El proyecto está congelado y todavía no soporta todas las características de XSL- FO.
  - El resultado es aceptable, pero puede dar problemas con documentos grandes y complejos.
  - Es aconsejable pasar el siguiente parámetro a xsltproc:

```
--string-param fop.extensions 1
```

XSL-FO y PDF (IV)

- XSL-FO ⇒ LaTeX ⇒ PDF
  - Utilizando PassiveTeX. Suele venir incluido en las distribuciones de LaTeX, pero como *componente opcional*.
  - En Debian: paquetes “passivetex” y “xmltex”
  - Para convertir de XSL-FO a PDF:
    `pdfxmltex` *`fichero_xslfo`*
  - Es aconsejable pasar el siguiente parámetro a xsltproc:

```
--string-param passivetex.extensions 1
```

XSL-FO y PDF (V)

- Calidad tipográfica de LaTeX. Suele producir mejores resultados que FOP.
- Sin embargo, si nos encontramos con errores o problemas, nos costará entenderlos si no sabemos LaTeX

Parametrización (I)

- Podemos parametrizar la generación de XHTML, XSL-FO, etc.
  - Pasando parámetros al procesador XSLT
  - Creando una hoja de transformación XSLT como extensión de una existente
- Los parámetros son simplemente parejas (nombre,valor)
- La lista completa de parámetros está disponible en:
  http://docbook.sourceforge.net/release/xsl/current/doc/reference.html

Parametrización (II)

- Pasando parámetros al procesador XSLT:
  - Pasando a xsltproc el siguiente parámetro desde la linea de comandos:

```
--string-param nombre_param valor_param
```

  - Por ejemplo, para eliminar los enlaces de navegación en el pie de las páginas (al generar varios documentos XHTML):

```
--string-param suppress.footer.navigation 1
```

Parametrización (III)

- Creando una nueva hoja XSLT:
  - Si utilizamos muchos parámetros, la invocación de XSLT puede resultar bastante engorrosa.
  - Podemos definir una hoja de transformación en función de otra.
  - En nuestra nueva hoja solo tenemos que incluir nuestros parámetros.
  - Indicaremos al procesador XSLT que queremos utilizar la nueva hoja.

Parametrización (IV)

```
<?xml version='1.0'?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">


<xsl:import href="Hoja en la que nos basamos"/>

<xsl:param name="param1">valor1</xsl:param>
<xsl:param name="param2">valor2</xsl:param>
<xsl:param name="param3">valor3</xsl:param>

</xsl:stylesheet>
```

Parametrización (V)

```
<?xml version='1.0'?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">


<xsl:import
href="http://docbook.sourceforge.net/release/xsl/current/xh
tml/chunk.xsl"/>

<xsl:param name="suppress.footer.navigation">1</xsl:param>

</xsl:stylesheet>
```

Parametrización (VI)

- Si no encontramos un parámetro que nos permita ajustar algún aspecto concreto de la generación, podemos añadir nuestro propio código XSLT.
  - Las modificaciones más habituales que requieren escribir código XSLT ya están bien documentadas.

DocBook

Unai Aguilera
gkalgan @ gmail .com

Créditos:
Nando Quintana
Pablo Pérez
Borja Sotomayor

Cursillos de Julio 2006