Lenguaje de Estilo XSL.

Presentación de documentos XML

- Ejemplo. Tenemos el fichero biblioteca.xml creado siguiendo las reglas de su DTD: biblioteca.dtd.

```
<?xml version="1.0" ENCODING="UTF-8"?>
<!DOCTYPE biblioteca SYSTEM "biblioteca.dtd">
                                        <!-- HASTA AQUi ES PROLOGO-->
<BIBLIOTECA> <!-- ELEMENTO DE DOCUMENTO-->
    <LIBRO>
        <TITULO>LA REGENTA</TITULO>
        <AUTOR>Leopoldo Alas "Clarin"</AUTOR>
        <EDITORIAL>RBA Editores</EDITORIAL>
        <CUBIERTA TIPO="DURA" />
        <CATEGORIA CLASE="FICCION" />
        <ISBN>ISBN-33490</ISBN>
        <NOTA CALIF="4"/>
        <COMENTARIOS>
        Es un clásico muy bueno aunque se puede hacer difícil de leer
        </COMENTARIOS>
    </LIBRO>
    <!-- Aquí podrían ir tantos libros como quisiésemos con el formato
        de este, donde se pueden quitar los comentarios -->
</BIBLIOTECA>
```

Presentación de documentos XML

- En la DTD habíamos definido una serie de elementos y sus atributos:

```
<!ELEMENT libro (titulo, autor, editorial, cubierta, categoria, isbn, nota,
    comentarios?)>
<!ELEMENT titulo (#PCDATA) >
<!ELEMENT autor (#PCDATA) >
<!ELEMENT editorial (#PCDATA) >
<!ELEMENT cubierta EMPTY>
<!ATTLIST cubierta TIPO (BLANDA|DURA) "BLANDA">
<!ELEMENT categoria EMPTY>
<!ATTLIST categoria CLASE (FICCION| FANTASIA| CFICCION| TERROR|
    HISTORICO | NOFICCION) "FICCION">
<!ELEMENT isbn (#PCDATA) >
<!ELEMENT nota EMPTY>
<!ATTLIST nota CALIF (1 |2|3|4|5) "3">
<!ELEMENT comentarios (#PCDATA) >
```

Presentación de documentos XML. XSL.

LENGUAJE DE ESTILO XSL- eXtensible Style Sheet

- El lenguaje de estilo extensible se desarrolló para ofrecer un modo de formateo más flexible y general que el que ofrecen las hojas de estilo en cascada. Estas tienen algunas limitaciones como por ejemplo que sólo nos permite acceder a elemento completos.

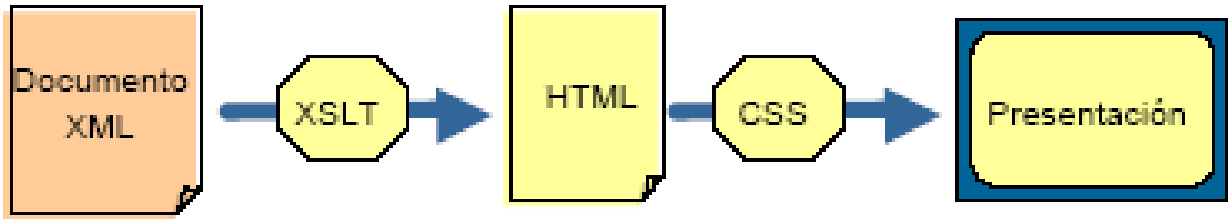
- XSL (Extensible Stylesheet Language) es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio específico.

Presentación de documentos XML. XSL.

- Una hoja de estilo XSL es también un documento XML que usa un DTD concreto, xsl:stylesheet.
- Esta familia está formada por tres lenguajes:
  - **XSLT** (siglas de Extensible Stylesheet Language Transformations, lenguaje de hojas extensibles de **transformación**), que permite convertir documentos XML de una sintaxis a otra (por ejemplo, de un XML a otro o a un documento HTML).
  - XSL-FO (lenguaje de hojas extensibles de **formateo** de objetos), que permite especificar el formato visual con el cual se quiere presentar un documento XML, es usado principalmente para generar documentos PDF.
  - XPath, o XML Path Language, es una sintaxis (no basada en XML) para acceder o referirse a porciones de un documento XML

Lenguaje de Estilo XSL.

- El **lenguaje de transformación** XSLT nos permite:
  - Partir de un documento XML conseguir otro documento XML perteneciente a otro vocabulario distinto.
  - El nuevo documento puede contener todo o parte de la información del primero.
  - Este tipo de transformaciones garantizan la compatibilidad entre sistemas que utilicen distintos vocabularios, simplemente hemos de añadir la hoja de estilo XSL adecuada
  - Permite transformar, por ejemplo, un documento XML en uno HTML



- El **lenguaje de formato** XSL-FO incluye una serie de objetos de presentación que asociados a los elementos del documento les asignan un aspecto determinado.

Lenguaje de Estilo XSL.

XSLT

- El lenguaje de transformación usa la estructura jerárquica del documento XML y básicamente transforma un árbol XML en otro (o en un documento HTML, o en texto).
- La hoja contiene:
  - Elementos de XSLT. Pertenecen al namespace xsl. Definidos por el estándar ye interpretados por cualquier procesador XSLT).
  - Elementos LRE (Literal Result Elements). Son elementos que no pertenecen a XSLT, sino que se repiten en la salida sin más.
  - Elementos de extensión. No-estándar, majenados por implementaciones específicas.
- Un documento XSL contiene una lista de plantilla (*templates)* y **reglas** que le permiten seleccionar elementos del árbol del documento y transcribirlos enteros o bien partes a otro vocabulario, al tiempo que se le añade nueva información, si ésta es necesaria.

Hojas de estilo: XSLT

- El procesador XSLT recorre el árbol desde la raíz (antes los elementos padre antes que los hijos).
- Para cada elemento, si existe una plantilla aplicable, se aplica y ya no se examinan más elementos descendientes (salvo que se solicite).
- Formato de un documento XSL (es un XML):

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

</xsl:stylesheet>
```

ELEMENTOS XSLT

- ELEMENTOS DE ALTO NIVEL
  - xsl:include que permite referenciar plantillas procedentes de una fuente externa. <xsl:include href="uri"/>
  - xsl:output que proporciona métodos para la salida (xml,html,text).
    - Ejem.: <xsl:output method="xml" encoding="ISO-8859-1"/>
  - xsl:strip-space que elimina antes del procesamiento todos los nodos consistentes en espacios en blanco.

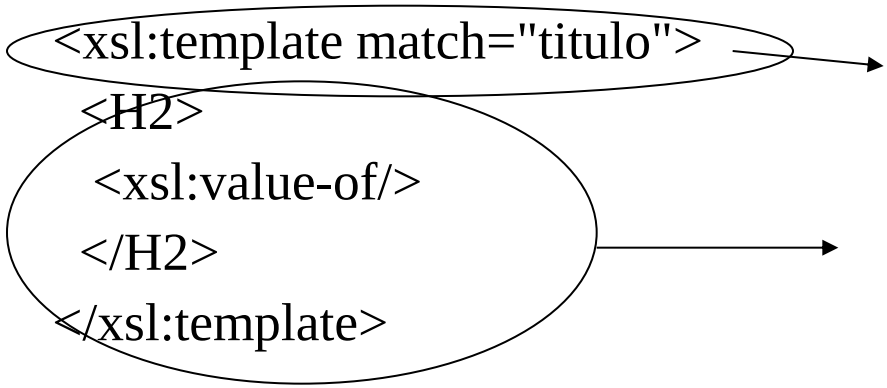
- INSTRUCCIONES XSLT

Hojas de estilo: XSLT

- Las **reglas** de construcción tienen dos partes:
  1. **Patrón**: parte de la regla que especifica la marca del documento origen a la que afecta la regla.
  2. **Acción**: parte de la regla que especifica como se traduce esa marca.
- Ejemplo:

  Fuente: <titulo> La historia de PI </titulo>

  Plantilla de la hoja de estilo:



Patrón: busca en el documento el elemento "titulo"

Acción: reemplazarlo por esto

Resultado: <H2> La historia de PI </H2>

Hojas de estilo: XSLT

```
<xsl:template match="/">
      <html>
            <head>
            </head>
            <body>
            </body>
            </html>
</xsl:template>
```

Patrón:busca el nodo raiz

Acción: transforma el elemento de documento en un documento html vacío.

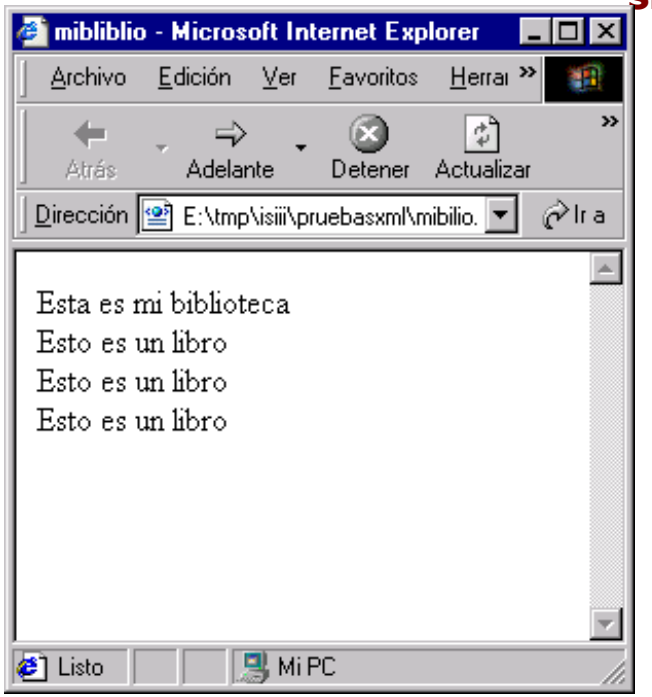
- Con esto el resto del árbol del documento queda sin procesar. Para comenzar a procesarlo necesitamos especificar el elemento de proceso, este se llama xsl:apply-templates

```
<xsl:template match="/">
      <html>
            <head>
            </head>
            <body>
                  <xsl:apply-templates />
            </body>
            </html>
</xsl:template>
```

Con esto se desciende los hijos del raíz pero habrá que definir reglas para ellos

Hojas de estilo: XSLT

- En el ejemplo biblioteca, si queremos evaluar cada libro:




```
<?xml version="1.0" ?>
<xsl:stylesheet type="text/xsl"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <HTML >
  <HEAD>
    <TITLE>
     mibliblio
    </TITLE>
   </HEAD>
   <BODY>
          <xsl:apply-templates/>
  </BODY>
</HTML>
</xsl:template >
<xsl:template match="biblioteca">
          Esta es mi biblioteca <BR/>
          <xsl:apply-templates />
</xsl:template >
<xsl:template match="libro">
          Esto es un libro <BR/>
 </xsl:template >
</xsl:stylesheet>
```

Regla para el documento

Regla para el elemento biblioteca

Hojas de estilo: XSLT

- Cuando evaluamos un nodo con xsl:apply-templates podemos añadir un atributo para seleccionar solo una parte de los hijos del nodo a la hora de continuar con el recorrido del árbol. Este atributo es select.

```
<xsl:template match="libro">
        <xsl:apply-templates select="titulo" >
</xsl:template >
```

En este caso solo los elementos titulo dentro de libro serian recorridos

Hojas de estilo: XSLT

- Lo que hemos visto hasta ahora nos permite controlar el recorrido del árbol, pero no tenemos acceso a la copia de valores de los nodos origen. Para ello utilizamos la marca xsl:value-of con el atributo select, esto copiará el contenido del nodo seleccionado en la salida.

```
<xsl:template match="/">
<HTML >
 <HEAD>
  <TITLE>
   mibliblio
  </TITLE>
 </HEAD>
 <BODY>
          <xsl:apply-templates/>
 </BODY>
</HTML>
</xsl:template >
<xsl:template match="biblioteca">
          Esta es mi biblioteca <BR/>
          <xsl:apply-templates />
</xsl:template >
<xsl:template match="libro">
          <xsl:value-of select="titulo" />
</xsl:template >
```

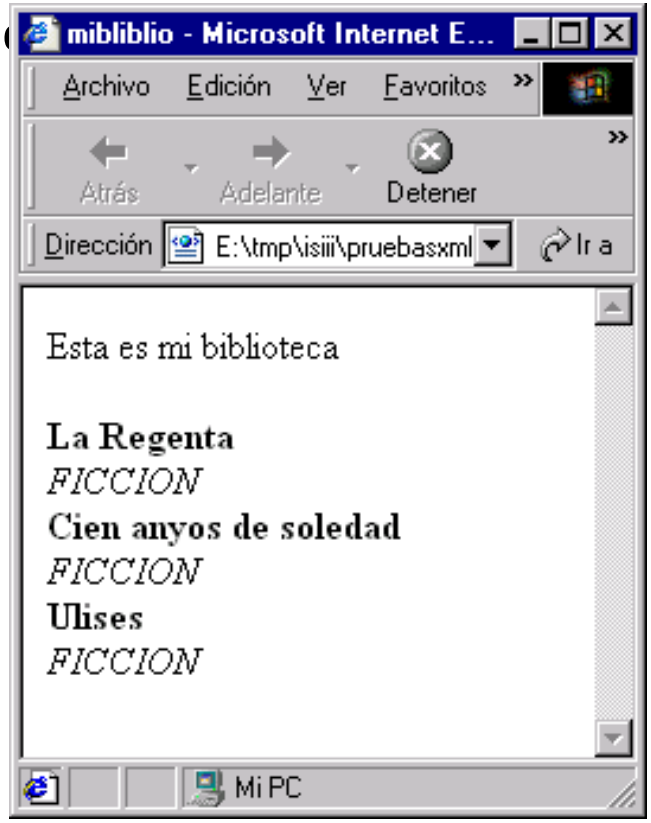
Hojas de estilo: XSLT

- También podemos acceder a los elementos a través de su identificador único (si lo tienen)

```
<xsl:template match=”id(‘e25’)">
        <i><xsl:value-of select=“.”/><i>
</xsl:template >
```

- El otro punto importante es localizar los atributos (empleamos el simbolo @ junto con el nombre del

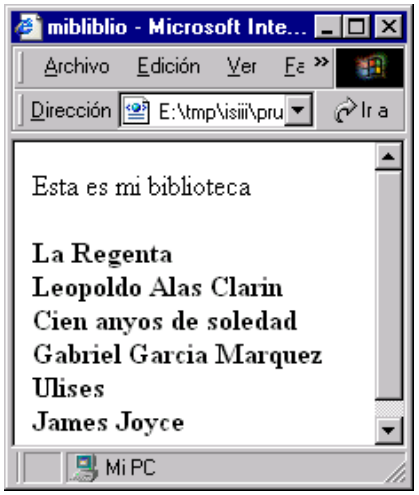
```
<xsl:template match="biblioteca">
        Esta es mi biblioteca <BR/>
        <xsl:apply-templates/>
</xsl:template >
<xsl:template match="libro">
        <xsl:apply-templates/>
</xsl:template >
<xsl:template match="titulo">

        <BR/><B><xsl:value-of select="."/></B>

</xsl:template >
<xsl:template match="categoria">
        <BR/><i><xsl:value-of select="@CLASE"/></i>
</xsl:template >
```



Hojas de estilo: XSLT

- Podemos hacer que una regla sea activada por varios tipos de elementos para ello basta separarlos por la barra vertical '|'. En el ejemplo siguiente se seleccionan los elementos titulo y autor y se les aplica negrita:

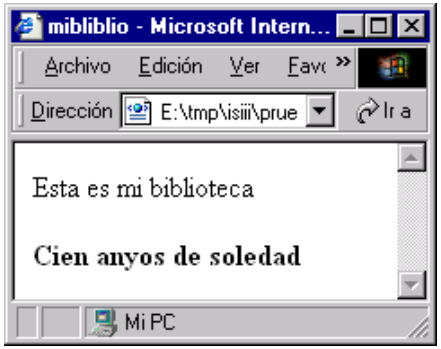
```
<xsl:template match="biblioteca">
        Esta es mi biblioteca <BR/>
        <xsl:apply-templates/>
</xsl:template >
<xsl:template match="libro">
        <xsl:apply-templates/>
</xsl:template >
<xsl:template match="titulo | autor">

        <BR/><B><xsl:value-of select="."/></B>
</xsl:template >
```



Hojas de estilo: XSLT

- También podemos hacer test de contenido para activar solo aquellos nodos que tengan dentro algo que nos interesa, para ello se utilizan corchetes '[ ]'. Ese algo pueden ser atributos, elementos, valores de atributos, contenido de elementos, posiciones en la jerarquía.
- En el ejemplo solo se sacan los libros con nota 5

```
<xsl:template match="biblioteca">
        Esta es mi biblioteca <BR/>
        <xsl:apply-templates select="libro[nota/@CALIF='5']" />
</xsl:template >
<xsl:template match="libro">
        <xsl:value-of select="titulo"/>
</xsl:template >
```



Ordenar. xsl:sort

- xsl:sort
  - Se especifica dentro de xsl:apply-templates o xsl:for-each
  - ¿Podría haber sido un atributo?
  - Su atributo es select
  - Indica cómo se establece el orden (ascending|descending)
- Ejercicio: hacer que los títulos salgan en orden alfabético

Condicional. xsl:if

- xsl:if
  – Atributo: test
  – El valor del atributo es una expresión booleana
  – Las instrucciones que contiene se ejecutan sólo si la condición se cumple

EJEMPLO

```
<?xml version = "1.0"?>
<agenda>
  <anyo value = "2000">
     <fecha mes = "7" dia = "15">
        <nota hora = "1430">Visita al m&#233;dico</nota>
        <nota hora = "1620">Clase de F&#237;sica en la  BH291C</nota>
     </fecha>
     <fecha mes = "7" dia = "4">
        ....
     <fecha mes = "7" dia = "20">
     .......
    <fecha mes = "7" dia = "20">
     ......
     <fecha mes = "7" dia = "20">
     ....
  </anyo>
</agenda>
```

Condicional: choose, when, otherwise

```
<xsl:choose>
        <xsl:when test = "@hora &gt;= '0500' and @hora &lt; '1200'">
          Ma&#241;ana (<xsl:value-of select = "@hora"/>):
        </xsl:when>
        <xsl:when test ="@hora &gt;= '1200' and @time &lt; '1700'">
          Tarde (<xsl:value-of select = "@hora"/>):
        </xsl:when>
        <xsl:when test = "@hora &gt;= '1700' and @hora &lt;= '2359'">
          Tarde-Noche (<xsl:value-of select = "@hora"/>):
        </xsl:when>
        <xsl:when test = "@hora &gt;= '0100' and @hora &lt; '0500'">
          Noche (<xsl:value-of select = "@hora"/>):
        </xsl:when>
        <xsl:otherwise>
          Todo el d&#237;a.
        </xsl:otherwise>
    </xsl:choose>
```

Iterando. xsl:for-each

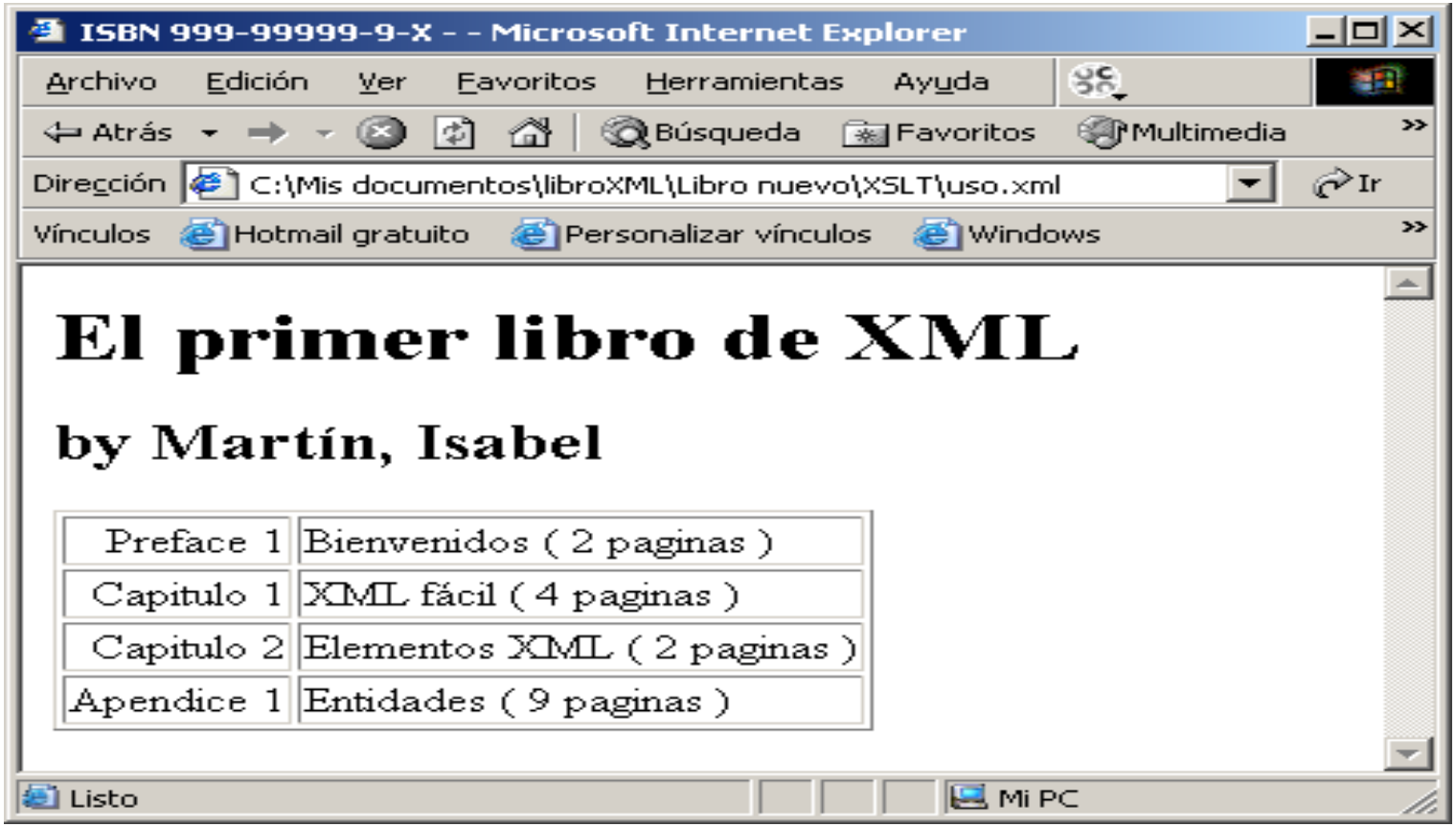
<xsl:for-each select=“ … “>

```
<?xml version = "1.0"?>
 <?xml:stylesheet type = "text/xsl" href = "uso.xsl"?>
  <libro isbn = "999-99999-9-X">
  <titulo> El primer  libro de XML </titulo>
    <autor>
       <nombre>Isabel </nombre>
       <apellido>Martín</apellido>
   </autor>
    <capitulos>
       <prefacio num = "1" paginas = "2">Bienvenidos</prefacio>
       <capitulo num = "1" paginas = "4">XML f&#225;cil</capitulo>
       <capitulo num = "2" paginas = "2">Elementos XML</capitulo>
        <apendice num = "1" paginas = "9">Entidades</apendice>
   </capitulos>
  </libro>
```

Iterando. xsl:for-each

```
<xsl:for-each select = "capitulos/prefacio">
            <xsl:sort select = "@num" order = "ascending"/>
            <tr>
              <td align = "right">
                Preface <xsl:value-of select = "@num"/>
              </td>
              <td>
                <xsl:value-of select = "."/> (
                <xsl:value-of select = "@paginas"/> paginas )
              </td>
            </tr>
</xsl:for-each>
<xsl:for-each select = "capitulos/capitulo">
          …..
 </xsl:for-each>
 <xsl:for-each select = "capitulos/apendice">
     …..
 </xsl:for-each>
```

Resultado


ISBN 999-99999-9-X - - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos Multimedia
Dirección C:\Mis documentos\libroXML\Libro nuevo\XSLT\uso.xml
Ir
Vínculos Hotmail gratuito Personalizar vínculos Windows
El primer libro de XML
by Martín, Isabel
Preface 1 Bienvenidos (2 paginas)
Capitulo 1 XML fácil (4 paginas)
Capitulo 2 Elementos XML (2 paginas)
Apendice 1 Entidades (9 paginas)
Listo
Mi PC

xsl:copy xsl:copy-of select

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "intro.xsl"?>
<miMensaje>
  <mensaje> Bienvenidos a  XSLT</mensaje>
</miMensaje>
```

```
<?xml version = "1.0"?>
<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match = "miMensaje">
    <xsl:copy>
      <xsl:apply-templates/>
    </xsl:copy>
  </xsl:template>
  <xsl:template match = "mensaje">
    <xsl:copy>
      &apos;Hola Mundo&apos; para variar
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Instrucciones de diseño

- <xsl:element name=“ … “> y <xsl:atrribute name=“ … “

Cuando definimos en tiempo de ejecución el nombre de elementos o atributos a partir del documento XML origen.

Ejercicio:

```
<?xml version = "1.0"?>
<deportes>
  <juego titulo = "baloncesto">  <id>243</id>
    <para>       Más popular en algunos Paises      </para>
  </juego>
  <juego titulo = "baseball"> <id>431</id>
    <para>        El más conocido en América      </para>
  </juego>
  <juego titulo = "futbol"> <id>123</id>
    <para>       El deporte más conocido en todo el mundo      </para>
  </juego>
</deportes>
```

Ejercicio

- XML resultado:

```
<?xml version = "1.0" encoding = UTF-8?>
<deportes>
    <baloncesto id =”243”>
    <comentario> Más popular en algunos Estados.</comentario>
    </baloncesto>
    <baseball id=431”>
        <comentario> El más conocido en América. </comentario>
    </baseball>
    <futbol id=”123”>
        <comentario>  El deporte más conocido en todo el mundo
    </comentario>
    </futbol>
</deportes>
```

Ejercicio XSL

```
<?xml version = "1.0"?>
<xsl:stylesheet version = "1.0"
        xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match = "/">
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match = "deportes">
    <deportes>
      <xsl:apply-templates/>
    </deportes>
  </xsl:template>
  <xsl:template match = "juego">
    <xsl:element name = "{@titulo}">
      <xsl:attribute name = "id">
        <xsl:value-of select = "id"/>
      </xsl:attribute>
      <comentario>
        <xsl:value-of select = "para"/>
      </comentario>
    </xsl:element>
  </xsl:template>
```

Definición de tipos

- xsl:variable xsl:text xsl:number

```
<?xml version = "1.0"?>
<xsl:stylesheet version = "1.0"
      xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<xsl:template match = "/">
  <total>
    Numero de paginas =
    <xsl:variable name = "suma"
      select = "sum(libro/capitulo/*/@paginas)"/>
    <xsl:value-of select = "$suma"/>
  </total>
</xsl:template>
</xsl:stylesheet>
```