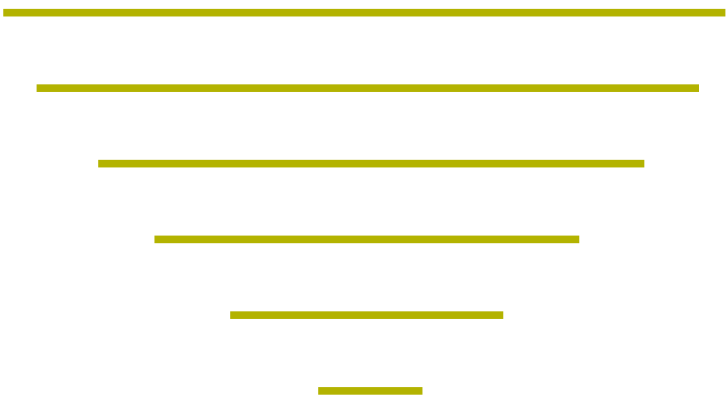
XSLT

Apuntes de guerrilla



Index

Aplicar un XSLT a un XML

En documento XSLT externo

En el documento xml introduciremos la siguiente linea diciéndole donde se encuentra el archivo.

```
<?xml version="1.0" encoding="UTF-8""?>
<?xml-stylesheet href="ejemplo1.xsl" type="text/xsl"?>
<elemento_raiz>
</elemento_raiz>
```

Como se forma el archivo xsl

Definiremos las etiquetas stylessheet. Dentro de ellas irá el código.

```
<?xml version="1.0" encoding="UTF-8""?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
                    .
                    .
</xsl:stylesheet
```

Definición de transformación

Podemos transformar el archivo resultante en xml o html, lo indicaremos de esta forma colocando la etiqueta dentro de stylesheet.

```
<xsl:output method="html" />
```

Formatear el sangrado del documento

En el caso de que queramos que los nodos hijos salgan con tabulaciones respecto a los nodos padres le añadiremos el valor “yes” al atributo indent.

```
<xsl:output method="html" indent=”yes”/>
```

Plantillas

Plantilla principal

Al aplicar la plantilla principal le indicaremos una dirección xpath para que nos muestre su contenido.

```
<xsl:template match='/'>
<html>
<head>
<title>Plantillas</title></head>
<body>
<xsl:apply-templates />
</body>
</html>
</xsl:template>
```

Plantillas externas

Las plantillas externas se definen independiente de la principal y se aplican en la etiqueta <xsl:apply-templates />.

```
<xsl:template match='/'>
<html>
<head>
<title>Plantillas</title></head>
<body>
<xsl:apply-templates />
</body>
</html>
</xsl:template>
<xsl:template match='nombre_elemento'>
</xsl:template>
```

Mostrar contenido de elementos

Usaremos la función value-of para mostrar la información que tiene un elemento.

```
<xsl:template match='/'>
<xsl:value-of select='elemento_ejemplo”/>
</xsl:template>
```

Insertar texto

En el caso de que queramos insertar texto a mano podemos hacerlo pero si queremos podemos usar la función text.

```
<xsl:template match='/'>
<p><xsl:text> Introducimos aquí el texto </xsl:text></p>
</xsl:template>
```

```
<xsl:template match='/'>
<p> Introducimos aquí el texto </p>
</xsl:template>
```

Podemos indicarle un atributo que sirve para que los editores que no permiten los caracteres que no son PCDATA.

```
<xsl:text disable-output-escaping="yes"> &quot; texto &quot;: </xsl:text>
```

Elementos

Podremos crear elementos en la transformación usando la etiqueta correspondiente o bien escribiendo el elemento directamente.

```
<xsl:template match='/'>
<xsl:element name=”ejemplo”> Introducimos aquí el texto </xsl:element>
</xsl:template>
```

```
<xsl:template match='/'>
<elmento>Introducimos aquí el texto </elemento>
</xsl:template>
```

Si queremos generar un xml con etiquetas que tengan como nombre el nombre del contenido del elemento elegido, así podemos hacerlo.

```
<xsl:template match="nombre">
<xsl:element name="{.}">texto de la etiqueta</xsl:element>
</xsl:template>
```

Atributos

Siempre acompañado de elemento podemos insertar atributos utilizando la etiqueta correspondiente o bien escribiéndolos directamente sobre el código.

```
<xsl:template match='/'>
<p><xsl:attribute name=”class”> ejemplo </xsl:attribute>
</p>
</xsl:template>
```

```
<xsl:template match='/'>
<p class=”ejemplo”>
</p>
</xsl:template>
```

También podemos crear atributos con los nombres de los atributos del documento xml.

```
<xsl:template match='/'>
<xsl:element name=”ejemplo”>
<xsl:attribute name="{.}">valor del atributo</xsl:attribute>
</xsl:element>
</xsl:template>
```

Copia de elementos y contenido

Podemos copiar parte del código origen en el caso de que el código destino sea igual al del origen, lo haremos d ella siguiente forma.

```
<xsl:template match='/'>
   <xsl:copy-of select=”.”/>
</xsl:template>
```

Copia de elementos y contenido

A diferencia del “copy-of” está etiqueta solo copia el elemento, sin su contenido.

```
<xsl:template match='/'>
   <xsl:copy/>
</xsl:template>
```

Elementos de control

For-each

Sirve para que se escriba la expresión en caso de que se cumpla la ruta xpath.

```
<xsl:template match='/'>
   <xsl:for-each select=”nodo1/nodo2”>
      <h1> <xsl:value-of select=”.”/></h1>
   </xsl:for-each>
</xsl:template>
```

If test

Para decidir si se cumplirá la expresión o no según las condiciones que queramos usaremos la siguiente etiqueta. < and > with < and >

```
<xsl:template match='/'>
   <xsl:for-each select="nodo1/nodo2">
      <xsl:if test=".='contenido_nodo'">
            <p> encontrado el conenido_nodo </p>
      </xsl:if>
   </xsl:for-each>
</xsl:template>
```

En el caso de que la condición tenga que ser mayor o igual y/o menor o igual tendremos que usar el código especial para esos caracteres.
< es < y > es > .

```
<xsl:template match='/'>
   <xsl:for-each select="nodo1/nodo2">
      <xsl:if test="numero_nodo &lt; 4">
            <p> encontrado el conenido_nodo </p>
      </xsl:if>
   </xsl:for-each>
</xsl:template>
```

Choose

Podemos crear una estructura de elección eligiendo el tipo de condición.

```
<xsl:template match='/'>
   <xsl:for-each select="nodo1/nodo2">
      <xsl:choose>
            <xsl:when test =".='contenido_nodo1'">
            <p> encontrado el conenido_nodo1 </p>
            </xsl:when>
            <xsl:when test =".='contenido_nodo2'">
            <p> encontrado el conenido_nodo2 </p>
            </xsl:when>
      </xsl:choose>
   </xsl:for-each>
</xsl:template>
```

En el caso de que no se cumpla ninguna podemos utilizar una opción alternativa que siempre se cumplirá en el caso de que no se cumplan las anteriores.

```
      <xsl:choose>
            <xsl:otherwise">
            <p> encontrado el conenido_nodo1 </p>
            </xsl:otherwise>
      </xsl:choose>
```

Ordenar

Para que los elementos dentro de un “for-each” salgan de forma alfabéticamente ordenada usaremos esta etiqueta.

```
<xsl:template match='/'>
   <xsl:for-each select=”nodo1/nodo2”>
      <xsl:sort select=”.”/>
      <h1> <xsl:value-of select=”.”/></h1>
   </xsl:for-each>
</xsl:template>
```

Códigos de ejemplo

Uso de html junto con xslt

Código original xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="horarios.xsl" type="text/xsl"?>
<horario>
  <dia>
    <numdia>1</numdia>
    <tarea prioridad="media">
      <hora-ini>12</hora-ini>
      <hora-fin>14</hora-fin>
      <asignatura>Tutorías</asignatura>
    </tarea>
  </dia>
  <dia>
    <numdia>2</numdia>
    <tarea prioridad="alta">
      <hora-ini>12</hora-ini>
      <hora-fin>14</hora-fin>
      <asignatura>Fol</asignatura>
    </tarea>
  </dia>
  <dia>
    <numdia>4</numdia>
    <tarea prioridad="alta">
      <hora-ini>9</hora-ini>
      <hora-fin>11</hora-fin>
      <asignatura>Lenguajes de Marcas</asignatura>
    </tarea>
    <tarea prioridad="alta">
      <hora-ini>16</hora-ini>
      <hora-fin>17</hora-fin>
      <asignatura>Inglés</asignatura>
    </tarea>
  </dia>
  <dia>
    <numdia>3</numdia>
    <tarea prioridad="alta">
      <hora-ini>9</hora-ini>
      <hora-fin>11</hora-fin>
      <asignatura>Procesadores de lenguajes</asignatura>
    </tarea>
  </dia>
  <dia>
    <numdia>5</numdia>
    <tarea prioridad="baja">
      <hora-ini>17</hora-ini>
      <hora-fin>18</hora-fin>
      <asignatura>Ver la tele</asignatura>
    </tarea>
  </dia>
</horario>
```

Código xsl.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <head>
        <title>Ejercicio</title>
      </head>
      <body>
      <p> Las asignaturas que termina después de 15h. y su prioridad no es alta son::</p>
        <xsl:for-each select="horario/dia/tarea">
        <xsl:if test="hora-ini &amp;gt; 15">
          <xsl:if test="./@prioridad='baja'">
           <p>-<xsl:value-of select="asignatura/." /></p>
          </xsl:if>
        </xsl:if>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Html resultante.

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <head>
    <title>Ejercicio</title>
  </head>
  <body>
    <p> Las asignaturas que se inicia siempre antes de la 13h, son:</p>
    <p>-Tutorías</p>
    <p>-Fol</p>
    <p>-Lenguajes de Marcas</p>
    <p>-Procesadores de lenguajes</p>
  </body>
</html>
```

Al igual que el mapa en el bolsillo, la cantimplora en el cinturón y el machete cruzado en la espalda es esencial para la vida de un guerrillero estos manuales facilitan la lucha constante contra los trabajos de programación.

Los informáticos tenemos la suerte de tener internet cerca de nosotros, pero cuando esa suerte no existe entonces tenemos que recurrir al papel o pdf. Estos pequeños manuales no atienden a explicaciones para principiantes ni avanzados, simplemente sacian las consultas de las dudas que pueden surgir programando en cualquier sitio, en el día a día.

Cuando estas lejos de tu puesto de trabajo, internet no está ahí o simplemente la red no funciona el guerrillero informático tiene el manual en el bolsillo, la botella de agua en la mochila y el portátil cruzado en la espalda.

Autor: Jesús Benages Sales

Contacto:jobinary@hotmail.com