GETsnippets

Java

Intger value of String

```
int type = Integer.parseInt(String)
```

String return ""

```
        private String getParameter(String input)
         {
           if(input == null || input.equals(""))
                 return "";
           else return input;
         }

           private String getNull(String input)
         {
           if(input == null || input.equals(""))
                 return "0";
           else return input;
         }
```

esql

esql-query loop
{{{<esql:execute-query>
<esql:query>select * from TABLE</esql:query>
<esql:results>
<esql:row-results>
<esql:get-columns/>
</esql:row-results>
</esql:results>
<esql:no-results>
<no-results/>
</esql:no-results>
<esql:error-results/>
</esql:execute-query>
}}}
_If your <body> element just contains text, the following templates will transform it and escape any apostrophes present, replacing them with
' character entities:

```
<xsl:template match="body">
        <xsl:text>&lt;body&gt;</xsl:text>
        <xsl:call-template name="escape-apos">
                <xsl:with-param name="text" select="."/>
        </xsl:call-template>
        <xsl:text>&lt;/body&gt;</xsl:text>
</xsl:template>

<xsl:template name="escape-apos">
<xsl:param name="text"/>
<xsl:variable name="apos">'</xsl:variable>
 <xsl:choose>
  <xsl:when test="contains($text, $apos)">
   <xsl:value-of select="substring-before($text, $apos)"/>
   <xsl:text disable-output-escaping="yes">&amp;apos;</xsl:text>
   <xsl:call-template name="escape-apos">
    <xsl:with-param name="text" select="substring-after($text, $apos)"/>
   </xsl:call-template>
 </xsl:when>
 <xsl:otherwise>
  <xsl:value-of select="$text"/>
 </xsl:otherwise>
</xsl:choose>
</xsl:template>
```

xhtml_

Forms original by W3schools_
{{{<form action="form_action.xsp" method="get">
First name:
<input type="text" name="fname" value="Mickey" />

Last name:
<input type="text" name="lname" value="Mouse" />

<input type="submit" value="Submit" />
</form>
}}}_
table - more "Optional Attributes" W3schools_
{{{<table border = "1">
<tr>
<td>Cell A</td>
<td>Cell B</td>
</tr>
</table>
}}}

xsl_

Match everything and apply all templates

```
<xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()" />
    </xsl:copy>
  </xsl:template>
```

_Define a variable

```
<xsl:variable name="name" select="expression">
<!-- Content:template -->
</xsl:variable>
```

_
Request paramteres in xsl

The "best" way (easier) is to use :

```
<xsl:param name="my_req_param" />
```

at the beginning of your xsl, then use it like a variable.

And in the sitemap, add to the xslt <map:transform ...> :

```
<map:transform ...>
<map:parameter name="use-request-parameters" value="true"/>
</map:transform>
```

Testing whether certain criteria matches then apply template

```
<xsl:for-each select="paragraph">
 <xsl:if test="visible = 1">
  <xsl:apply-templates/>
 </xsl:if>
</xsl:for-each>
```

attribute testing
`[@name()!='name']`
_with xsl:if

```
<xsl:if test="expression">
<!-- Content: template -->
</xsl:if>
```

_
with xsl:choose - block_

```
<xsl:choose>
        <xsl:when test=""></xsl:when>
        <xsl:otherwise></xsl:otherwise>
</xsl:choose>
```

_
If you test a string sometimes it is important that you don't have whitespaces!

```
normalize-space(.)
```

The following XPath expression selects all the cd elements of the catalog element that have a price element:

```
/catalog/cd[price]
```

Uppercase to lower case and vice versa

```
  <xsl:template match="upper-case">
    <xsl:value-of select="translate(current(),
'abcdefghijklmnopqrstuvwxyz',
'ABCDEFGHIJKLMNOPQRSTUVWXYZ')" />
  </xsl:template>

  <xsl:template match="lower-case">
    <xsl:value-of select="translate(current(),
'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'abcdefghijklmnopqrstuvwxyz')" />
  </xsl:template>
```

format Percent

```
<xsl:template match="formatPercent">
  <xsl:value-of select="concat('(',current(),'%)')" />
</xsl:template>
```

format Currency

```
<xsl:template match="formatCurrency">
  <xsl:value-of select="concat(current(),'$')" />
</xsl:template>
```

King regards

Thorsten Scherler