Hypertext and hypermedia
Project (2021)

Theme:

MY HOBBY

Stages:

Stage	Score [pts]
HTML, XML document, XML Schema, DTD	25
XSLT	15

XSLT:

Requirements:

Using created during first stage XML file and XSLT put the data on the screen in browser.

- Two different transformations and two different output HTML files should be created.
  - Output HTML files should differ in both graphical form (fonts, colours, etc.) and content.
- The templates should be used. The **for-each** instruction may be used **no more than twice**.
- Dodatkowo stworzyć trzeci plik z transformacjami, pozwalający przekształcić plik XML, do nowego pliku XML o innej strukturze.

- In the first XSLT file declare and use (**10,5 pkt**):
  - at least 8 not trivial templates (except the root template) matching to element (**1,6pts**)
  - at least 1 template matching to attribute (**0,5pts**)
  - at least 1 template with 'name' attribute (**0,5pts**)
  - choose and if instructions (in total at least 3 times) (**0,9pts**)
  - loop instruction including sort element (**0,8pts**)
  - at least 2 '*number*' tag with different parameters, one for sorted elements (**1pkt**)
  - XPath expressions: paths, at least 3 predicates (**1pts**)
  - at least 5 different XPath and/or XSLT functions (**1pts**)
  - variable at least 3 times (simple and complex) (**0,7pkt**)
  - stylesheet (css) (**0,3pts**)
  - twice format-number function with different parameters (**0,6pts**)
  - photos in output HTML file (**0,8pts**)
  - active links in output HTML file (**0,8pts**)
- The second transformation may be very simply (for example reduced first transformation). The aim of its creation is to show the possibility of extracting various information from an XML file and presenting it on a website in a different form. (**0,5 pts**)
- The third XSLT file (**4pts**) transforms XML file into new XML file with different structure
  - at least 4 levels in output XML file (excluding root element) (**0,4pkt**)
  - at least 4 tags with names different from tag names in the input file (**0,4pkt**)
  - in the output XML file, one of the tag's name should be the value of any tag or attribute from the XML file. For example:
    in input XML file <name>John</name>
    in output XML file < John>value</John >. (**0,8pts**)
  - similarly, in the output XML file, the name of one of the attributes should be the value of any tag or attribute from the XML file (**0,8pts**)
  - use
    - copy (**0,4pts**)
    - copy-of (**0,4pts**)
    - element (**0,4pts**)

- attribute (**0,4pts**)

Note:

- The final number of points for the project depends on the answers given during project evaluation, grasp of the project and of the theory.
- During the evaluation the project **code** must be **free** of any **comments.**

Selected sample errors:

- more than double use of the loop instruction **(-2pts)**
- incorrect use of templates (**-1pts**)
- incorrect numbering of the sorted list (**-0.5pts**)
- the same templates with matching to different elements / attributes (**-1pts**)

XSLT useful information

1. **Templates** describe what is to be done
   <xsl:template match=" XPath_expression" name="name" mode="mode">
   ...
   </ xsl:template>
2. **Calling the template with the 'match' attribute**
   <xsl:apply-templates select=" XPath_expression"/>
3. **Calling the template with the 'name' attribute**
   <xsl:call-template name="name"/>
4. **extracting the value of a selected node**
   <xsl:value-of select="XPath_expression"/>
   "/" root
   "name_of_an_element" element
   "@ name_of_an_attribute" attribute
   "." current node
5. **conditional instruction**
   <xsl:if test="expression">
   ...
   </xsl:if>
6. **multiple conditional tests**
   <xsl:choose>
   <xsl:when test=" expression 1">
   ...
   </xsl:when>
   <xsl:when test=" expression 2">
   ...
   </xsl:when>
   ...
   <xsl:otherwise>
   ...
   </xsl:otherwise >
   </xsl:choose>
7. **loop**
   <xsl:for-each select="XPath_expression">
   ...
   </xsl:for-each>

8. **sorting**
   <xsl:sort select=" XPath_expression" order="sort_order"/>
   sort_order
   ascending
   descending
9. **numbering**
   <xsl:number format="format"/>
   Format e.g. 1) a) A. etc.
10. **creation of new elements and attributes**
   <xsl:element name=" the_name_of_the_element"></xsl:element>
   <xsl:attribute name=" the_name_of_the_attribute"></xsl:attribute>