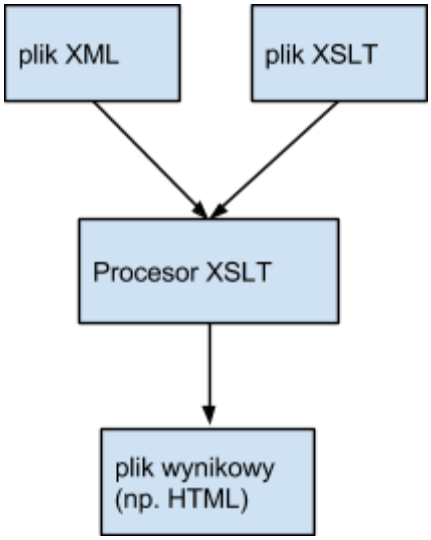
Przetwarzanie dokumentów XML i zaawansowane techniki WWW

"Przetwarzanie dokumentów XML za pomocą procesora XSLT "

(Zajęcia 06 - 11.04.2016 r.)

Przetwarzanie dokumentów XML oznacza zwykle, wyłuskanie z nich danych oraz przetwarzanie ich na inny format np HTML lub PDF. Jedną z metod przekształcania i przetwarzania dokumentów XML jest technika XSLT (Extensible Stylesheet Language Transformation).

Diagram modelu przekształcenia jest następujący:



Pliki opisujące transformację XSLT to również pliki w formacie XML w których nalezy odwoływać się do przestrzeni nazw: xmlns:xsl="http://www.w3.org/1999/XSL/Transform".

Procesor XSLT umożliwiający wykonywanie przekształceń w systemie Linux jest zainstalowany w lini komend:

```
> xsltproc  plik.xsl plik.xml
```

w wyniku działania tego polecenia na ekranie otrzymamy gotowy kod przekształcenia. Czasem jednak wygodniej jest od razu zapisać przekształcenie do pliku co można osiągnąć przez wykonanie polecenia:

```
> xsltproc  -o wynik.html plik.xsl plik.xml
```

W związku z tym że pliki do przekształceń są plikami XML należy rozpocząć je od prologu XML, a następnie określamy dokument XSLT. W tym miejscu musimy również określić co ma być wynikiem przekształcenia przez element <xsl:output />. Minimalna postać pliku z przekształceniem wygląda następująco:

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

<!-- ty znajdują się polecenia przekształcające -->

</xsl:stylesheet>
```

Kolejnym podstawowym elementem pliku z przekształceniem jest tzw. "template" dla elementu głównego który dokonuje identyfikacji głównego węzła dokumentu XML poszukując go za pomocą odniesienia w formacie XPath: "/".

```
 <xsl:template match="/">
   <!-- ty znajdują się polecenia przekształcające lub tekst -->
 </xsl:template>
```

W zasadzie każda reguła <template> posiada atrybut "match" którego wartością jest wyrażaenie XPath. Procesor XSLT analizuje drzewo XML od węzła do węzła i jeżeli na trafi na pasujące wyrażenie XPath

to stosuje dla niego regułę zapisaną we wzorcu. Należy również zwrócić uwagę że wszystkie elementy nie pochodzące z przestrzni nazw z prefixem “xsl” są kopiowane przez procesor XSLT do wynikowego dokumentu. Dlatego w łątwy sposób możemy dodoawać znaczniki html niezbędne do utworzenia strony na bazie danych przkształconych z plików XML.

Przykładowy plik XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<nobeldata-v1>
  <prizes continent="america" prizes="0" decade="1900s"/>
  <prizes continent="america" prizes="0" decade="1910s"/>
  <prizes continent="america" prizes="0" decade="1920s"/>
  <prizes continent="america" prizes="3" decade="1930s"/>
  <prizes continent="america" prizes="2" decade="1940s"/>
  <prizes continent="america" prizes="1" decade="1950s"/>
  <prizes continent="america" prizes="2" decade="1960s"/>
  <prizes continent="america" prizes="3" decade="1970s"/>
  <prizes continent="america" prizes="3" decade="1980s"/>
  <prizes continent="america" prizes="2" decade="1990s"/>
</nobeldata-v1>
```

Chcemy przkształcić dane zawarte w pliku XML do postaci tabeli osadzonej na prostej stronie internetowej. Do tego celu wykorzystujemy przekształcenie XSLT w następującej postaci:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="UTF-8" />
   <xsl:template match="/">
      <html>
      <head>
      <title>Nagrody Nobla w dziedzinie literatury</title>
      </head>
      <h1>Nagrody Nobla w dziedzinie literatury</h1>
      <p>Liczba nagrod Nobla przyznanych Amerykanom w poszczegolnych
            dekadach dwudziestego wieku</p>
      <table border="1">
      <tr>
            <th>Dziesieciolecie</th>
            <th>Nagrody</th>
      </tr>

      <xsl:apply-templates />

      </table>
      </html>

   </xsl:template>

   <xsl:template match="prizes">
      <tr>
      <td>
```

```
      <xsl:value-of select="@decade"/>
      </td>
      <td>
      <xsl:value-of select="@prizes"/>
      </td>
      </tr>
   </xsl:template>

</xsl:stylesheet>
```

Jak widać w głownym wzorcu znajdując się elemeny budujące strone html oraz polecenie dla procesora xslt nakazujące zastosowanie wzorców. Jako że w naszym przekształceniu mamy tylko jeden wzorzec odnoszący się do elementu potomnego <prizes> to on zostanie zastosowany. We wzorcu przekształcenia dla węzła <prizes> budowana jest struktura tabeli w html, a wartości do wypełnienia tebali są agregowane dzięki zastosowaniu polecenia <xsl:value-of select=”XPath” /> gdzie wartość atrybutu select jest wyrażeniem XPath wskazującym na żadane dane. Wyrażenie wzorca będzie wykonane tyle razy ile występuje pasujący element do wzorca.

Kolejny przykład bardzo prosty przykład:

```
<?xml version="1.0" encoding="UTF-8"?>
<invited>
    <speaker>
      <name>(TBC) Giuseppe Battistoni</name>
      <institute>National Institute of Nuclear Physics, Milano,
Italy</institute>
    </speaker>
    <speaker>
      <name>Mikhail Bashkanov</name>
      <institute>University of Tuebingen, Germany</institute>
    </speaker>
    <speaker>
      <name>Piotr Białas</name>
      <institute>Jagiellonian University, Cracow, Poland</institute>
    </speaker>
    <speaker>
      <name>Caterina Bloise</name>
      <institute>National Institute of Nuclear Physics, LNF Frascati,
Italy</institute>
    </speaker>
</invited>
```

oraz przekształcenie:

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

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

```
    <ul style="font-size:17px;line-height:150%">
     <xsl:apply-templates />
      </ul>
      </body>
      </html>
    </xsl:template>

      <xsl:template match="speaker">
      <li>
      <b><xsl:value-of select="name"/></b>
      -
      <i><xsl:value-of select="institute"/></i>
      </li>
      </xsl:template>

</xsl:stylesheet>
```

Dalsze przykłady zastosowania <xsl:apply-templates>

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <cd>
       <title>Empire Burlesque</title>
       <artist>Bob Dylan</artist>
       <country>USA</country>
       <company>Columbia</company>
       <price>10.90</price>
       <year>1985</year>
    </cd>
    <cd>
       <title>Hide your heart</title>
       <artist>Bonnie Tyler</artist>
       <country>UK</country>
       <company>CBS Records</company>
       <price>9.90</price>
       <year>1988</year>
    </cd>
</catalog>
```

Korespondujący plik z przekształceniem:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>
```

```
<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>
<xsl:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>
<xsl:template match="artist">
  Artist: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>
</xsl:stylesheet>
```

3) Dodatkowe funkcje XSLT (cały czas korzystamy z pliku XML z poprzednich zajęć o nazwie katalog.xml):

- instrukcja warunkowa “if”

```
<xsl:if test="warunek">
  instrukcje w przypadku spełnienia warunku
</xsl:if>
```

przykład:

```
<xsl:template match="cd">
    <xsl:if test="price &gt; 5">
      <p>
       <xsl:apply-templates select="title"/>
       <xsl:apply-templates select="artist"/>
      </p>
    </xsl:if>
</xsl:template>
```

- pętla “for-each”

```
<xsl:for-each select="wyrazenie-XPath">
      instrukcje
</xsl:for-each>
```

przykład (w template match=”/”)

```
<xsl:for-each select="catalog/cd">
  <p>
     Artist: <span style="color:#ff0000">
     <xsl:value-of select="artist"/></span>
     Title: <span style="color:#00ff00">
     <xsl:value-of select="title"/></span>
   </p>
  </xsl:for-each>
```

- sortowanie za pomoca funkcji “sort” elementów wypisywanych w pętli for-each:
`<xsl:sort select="co-ma-być-sortowane"/>` -- instrukcja nie posiadająca struktury!!!

przykład:

```
<xsl:for-each select="catalog/cd">
  <xsl:sort select="artist"/>
  <p>
     Artist: <span style="color:#ff0000">
     <xsl:value-of select="artist"/></span>
     Title: <span style="color:#00ff00">
     <xsl:value-of select="title"/></span>
   </p>
  </xsl:for-each>
```

- instrukcja warunkowa wyboru “choose”:

```
<xsl:choose>
    <xsl:when test="warunek-1">
            instrukcja kiedy warunek 1 spełniony
    </xsl:when>
    <xsl:when test="warunek-2">
            instrukcja kiedy warunek 2 spełniony
    </xsl:when>
    <xsl:otherwise>
            instrukcja kiedy żaden z waunków nie jest spełniony
    </xsl:otherwise>
</xsl:choose>
```

przykład:

```
<xsl:choose>
  <xsl:when test="price &gt; 10">
       Price:<span style="color:#ff00ff">
       <xsl:value-of select="price"/></span>
  </xsl:when>
  <xsl:when test="price &gt; 9">
       Price:<span style="color:#222999">
       <xsl:value-of select="price"/></span>
  </xsl:when>
  <xsl:otherwise>
      Price:<span style="color:#555aaa">
      <xsl:value-of select="price"/></span>
  </xsl:otherwise>
</xsl:choose>
```

Zadanie na ćwiczenia:

Proszę przygotować plik XML zawierający dowolne dane tabelaryczne np . bazę adresów, spis książek, spis smachodów, tak aby zawierały kilka elementów zagnieżdzonych. Następnie proszę przygotować przekształcenie XSLT bazujące na podanych przykładach umożliwiające otrzymanie w wyniku przekształcenia prostej strony prezentującej dane zawarte w przygotowanym pliku XML.