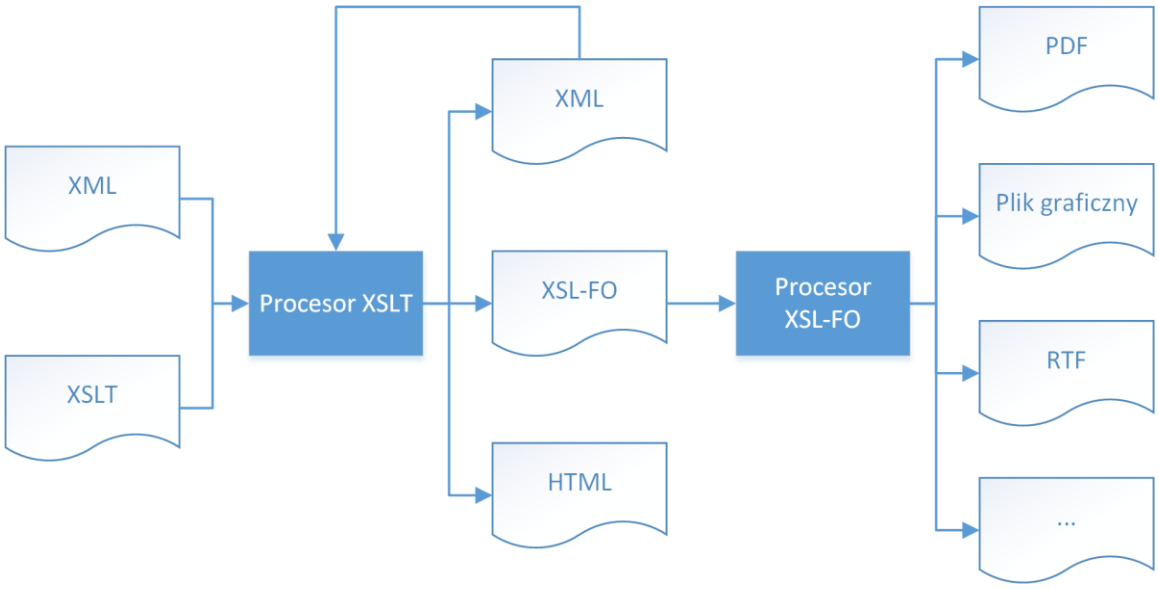
Laboratorium Hipertekst i Hipermedia
XSLT i XSL-FO

1. Wprowadzenie

XSL, czyli EXtensible Stylesheet Language jest językiem arkuszy stylów przeznaczonym dla plików XML. W ogólności, służy on do przetwarzania i formatowania danych zawartych w pliku XML w taki sposób, aby można je było przedstawić w postaci odpowiedniej dla dokumentu HTML, pliku PDF, pliku graficznego lub innych formatów. Język XSL składa się z trzech elementów:

- XSLT – transformacje XSL, służące do przekształcania plików XML do postaci dokumentów HTML lub innych dokumentów XML,
- XPath – umożliwiający odwoływanie się do elementów XML i ich atrybutów,
- XSL-FO (XSL – Formatting Objects, nazywany też po prostu XSL) – język formatowania obiektów, umożliwiający dostosowanie prezentacji danych XML do wyświetlenia w postaci np. pliku PDF.

Poniższy rysunek przedstawia przebieg przetwarzania pliku XML z użyciem XSL.



Jeśli wynikiem przetwarzania XSLT jest plik XML, może on zostać poddany kolejnym transformacjom XSL. Przykładowo, pierwsze przekształcenie realizuje modyfikację struktury drzewa dokumentu XML, a kolejne jest transformacją wynikowego XML do dokumentu HTML lub XSL-FO.

2. Przetwarzanie XSLT

Arkusze stylów XSL, podobnie jak pliki XML i HTML, są tworzone z użyciem znaczników. Co więcej, plik XSLT jest także rodzajem piku XML, rozpoczynającym się znacznikiem: `<?xml version="1.0" encoding="UTF-8"?>` , po którym rozpoczyna się znacznik: `<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">` ,
obejmujący cały arkusz stylów (znacznik `</xsl:stylesheet>` kończy plik XSLT).

Aby uniknąć niejednoznaczności w nazewnictwie, znaczniki transformacji XSL korzystają z przestrzeni nazw `xsl`, zatem mają one postać `<xsl:nazwa_znacznika>`. Użycie

odpowiednich elementów języka umożliwia realizację m.in. operacji przetwarzania warunkowego, sortowania, numerowania czy też modyfikacji oryginalnego drzewa elementów XML. Poniżej opisano podstawowe elementy transformacji XSL, które stanowią przedmiot zajęć laboratoryjnych.

2.1 Szablony

Szablon jest fragmentem arkusza XSLT objętym znacznikiem `<xsl:template>`. Można go przyrównać do funkcji, która zwraca, w miejscu jej wywołania, fragment wynikowego dokumentu XML lub HTML. Wyróżnia się dwa rodzaje szablonów: szablony nazwane i szablony z dopasowaniem.

2.1.1 Szablony nazwane

Definiowane są z atrybutem `name` (`<xsl:template name="nazwa_szablonu">`). Ich jednorazowe wywołanie jest realizowane przy użyciu znacznika

```
<xsl:call-template name="nazwa_szablonu"/>
```

Przykład – szablon wyświetlający na stronie tekst powitania w formie nagłówka:

```
<xsl:template name="powitanie">
      <h2>Witaj na stronie!</h2>
</xsl:template>
```

2.1.2 Szablony z dopasowaniem

Definiowane są z atrybutem `match` (`<xsl:template match="wyrażenie_xpath">`). Ich wywołanie jest realizowane przy użyciu znacznika

`<xsl:apply-templates select="wyrażenie_xpath"/>`.

W tym przypadku szablon odwołuje się do elementów dokumentu XML, określonych wyrażeniem zgodnym ze składnią XPath. Zazwyczaj główny szablon dokumentu XSLT zawiera dopasowanie do korzenia (root, /) drzewa elementów XML

`<xsl:template match="/">`.

W odróżnieniu od szablonów nazwanych, znacznik `apply-templates` może skutkować wielokrotnym wywołaniem szablonu, dla każdego elementu XML pasującego do wyrażenia `match`. Jeśli konieczne jest zdefiniowanie więcej niż jednego szablonu z dopasowaniem dla elementów o tej samej nazwie, należy te szablony rozróżnić poprzez przypisanie do nich odmiennych wartości parametru `mode`.

Przykład – wyświetlenie nazwisk wszystkich studentek z grupy:

```
<xsl:template match="/"> <!-- główny szablon arkusza-->
      <html>
            <body>
                  <xsl:apply-templates select="grupa/student[plec = 'K']">
                  <br/>
            </body>
      </html>
</xsl:template>
<xsl:template match="student"> <!--szablon dopasowany do elementu <student>-->
      <xsl:value-of select="nazwisko"/> <!--zwraca wartość elementu <nazwisko>-->
</xsl:template>
```

2.2 Kontrola przebiegu transformacji

Wywoływanie procedur transformacji XSL może być kontrolowane poprzez instrukcje warunkowe i pętle.

2.2.1 Warunkowe wykonanie

Jeśli szablon albo jego fragment ma zostać zastosowany jedynie w przypadku, gdy spełniony jest określony warunek, należy ten fragment objąć znacznikiem
`<xsl:if test="warunek">`.
W przypadku gdy wynik transformacji ma być uzależniony od spełnienia jednego z wielu możliwych warunków, należy użyć znacznika `<xsl:choose>`, zawierającego, dla każdego z warunków, odrębny znacznik `<xsl:when test="warunek_nrX">`. Dla elementów nie spełniających żadnego z warunków wynik transformacji można zdefiniować wewnątrz znacznika `<xsl:otherwise>`. Do formułowania warunków można użyć następujących operatorów:

`And`	logiczna koniunkcja
`Or`	logiczna suma
`not()`	negacja
`=`	równość
`!=`	nierówność
`<`	mniejszy niż
`<=`	nie większy niż
`>`	większy niż
`>=`	nie mniejszy niż

Przykład – inna możliwość realizacji transformacji z punktu 2.1.2; sprawdzenie warunku przeniesione z `<xsl:apply-templates/>` do wnętrza szablonu z dopasowaniem:

```
<xsl:template match="student">
      <xsl:if test="plec = 'K'">
            <xsl:value-of select="nazwisko"/> <br/>
      </xsl:if>
</xsl:template>
```

Przykład – opisowe określenie wzrostu osoby:

```
<xsl:template match="osoba">
      <xsl:choose>
            <xsl:when test="wzrost &lt; 150"> niski </xsl:when>
            <xsl:when test="wzrost &gt; 180"> wysoki </xsl:when>
            <xsl:otherwise> średniego wzrostu </xsl:otherwise>
      </xsl:choose>
</xsl:template>
```

2.2.2 Wykonanie w pętli

Wielokrotne wykonanie określonego fragmentu transformacji jest najczęściej realizowane poprzez odpowiednie użycie szablonów. Jednakże, w przypadku niektórych operacji nie jest to możliwe. Wtedy należy użyć znacznika `<xsl:for-each select="wyrażenie_xpath">`.

Kod zawarty w tym znaczniku zostanie wykonany tyle razy, ile jest wystąpień elementów XML pasujących do wyrażenia.

2.3 Sortowanie

Czasami pożądane jest wykonanie transformacji w kolejności uzależnionej od wartości elementu potomnego lub wartości atrybutu. W takim przypadku pomocny jest znacznik `<xsl:sort select="wyrażenie_xpath"/>`. Domyślnie sortowanie odbywa się rosnąco. Aby odwrócić kolejność należy użyć atrybutu `order="descending"`. **Uwaga**: sortowanie wymaga użycia pętli `<xsl:for-each>`.

Przykład – sortowanie produktów w sklepie po cenie (wywołanie z szablonu głównego):

```
<xsl:for-each select="sklep/produkt">
      <xsl:sort select="cena"/>
      Produkt: <xsl:value-of select="nazwa">
      Cena: <xsl:value-of select="cena"> zł
      <br/>
</xsl:for-each>
```

2.4 Numerowanie

Gdy zachodzi potrzeba ponumerowania fragmentów dokumentu, pomocne jest użycie elementu `<xsl:number />`. Przy każdorazowym wystąpieniu tego elementu, jego wartość jest zwiększana. Domyślnie zwracana jest wartość liczbowa wyrażona cyframi arabskimi, jednakże, ustawiając odpowiednio atrybut format, można użyć np. numeracji liczbami rzymskimi czy też literami z alfabetu łacińskiego. Możliwe jest także tworzenie numeracji wielopoziomowej. Element `<xsl:number>` może zwracać, oprócz numerów porządkowych, również inne wartości, określone atrybutem `value`.

Przykład – ponumerowana lista studentów:

```
<xsl:template match="student">
      <xsl:number format="1. "/>
      <xsl:value-of select="imie"/> <xsl:value-of select="nazwisko"/>
</xsl:template>
```

Przykład – liczba studentów w grupie:

```
<xsl:template match="grupa">
      W grupie <xsl:number/> jest <xsl:number value="count(student)"/> studentów.
      <br/>
</xsl:template>
```

2.5 Korzystanie z kaskadowych arkuszy stylów

Pliki XML służą do przechowywania danych i nie zawierają informacji dotyczących ich prezentacji przez przeglądarkę WWW. Informacja dotycząca wyglądu strony jest zawarta w pliku HTML. Zatem informacje dotyczące np. układu, kolorów czy czcionek na stronie można zdefiniować w pliku XSLT. Jest to jednak mało wygodne rozwiązanie, ponieważ każda zmiana wyglądu strony wymaga ingerencji w arkusz transformacji. W pliku wynikowym transformacji XML do HTML można, podobnie jak w zwykłym pliku HTML, zawrzeć odwołanie do arkusza stylów CSS. Odwołanie to należy załączyć w sekcji nagłówkowej dokumentu:

```
<head>
      <link rel="stylesheet" type="text/css" href="ścieżka_do_pliku_css" />
</head>
```

2.6 Dodawanie nowych elementów i atrybutów XML

Wykonując transformację XML do XML można dodać do istniejącego drzewa nowe elementy oraz atrybuty elementów. Aby dodać element należy użyć znacznika

`<xsl:element name=”nazwa_elementu”>`.
Wewnątrz tego znacznika można zdefiniować jego wartość, elementy potomne lub atrybuty. W przypadku tych ostatnich używa się elementu
`<xsl:attribute name=”nazwa_atrybutu”>`.
Wartość atrybutu określa się wewnątrz tego znacznika.
Przykład:

```
<xsl:element name=”nowy_element”>
      <xsl:attribute name=”atrybut_nowego_elementu”>
            wartość_atrybutu
      </xsl:attribute>
      <xsl:element name=”nowy_element_potomny”>
            wartość_elementu_potomnego
      </xsl:element>
</xsl:element>
```

2.7 Wykonywanie transformacji XSL

Transformacje XSL mogą być wykonywane zarówno po stronie serwera, jak i po stronie klienta. W drugim przypadku procesorem XSLT jest zazwyczaj przeglądarka WWW. Ponadto, istnieją dedykowane aplikacje (np. jSimpleX), które dokonują transformacji i umożliwiają zapis wynikowego dokumentu do pliku.

Aby powiązać pliki XML i XSL należy w pierwszym z nich dołączyć, przed częścią zawierającą drzewo elementów, odwołanie do arkusza stylów w postaci:

```
<?xml-stylesheet type="text/xsl" href="ścieżka_do_pliku_xsl"?>
```

Po otwarciu takiego pliku XML w przeglądarce WWW, wyświetlony zostanie dokument stanowiący wynik transformacji. Takie rozwiązanie jest proste, jednakże niesie ze sobą pewne ograniczenia. Przede wszystkim, plik XML jest w tym przypadku powiązany tylko z jednym określonym arkuszem transformacji. Wykonanie transformacji wieloetapowej jest w związku z tym utrudnione. Ponadto, sposób wczytywania i przetwarzania arkusza XSLT jest z góry określony przez użytą przeglądarkę.

Nieco bardziej skomplikowanym, jednakże dającym większe możliwości, rozwiązaniem jest utworzenie dokumentu HTML, zawierającego funkcje napisane w języku JavaScript, realizujące odczyt plików wejściowych i transformacje XSL. Wykonanie sekwencji kilku transformacji nie stanowi w tym przypadku problemu. Niedogodnością jest jedynie konieczność napisania różnych postaci funkcji dla przeglądarki Internet Explorer (korzysta z ActiveX) i pozostałych przeglądarek.

Uwaga: Przeglądarka Google Chrome domyślnie blokuje dostęp do plików zapisanych na dysku lokalnym. Aby umożliwić realizację transformacji XSL przez tę przeglądarkę, należy uruchomić program z odpowiednią opcją (np. z linii poleceń):

```
>chrome.exe --allow-file-access-from-files
```

3. Przetwarzanie XSL-FO

Pliki XSL-FO są plikami XML zawierającymi znaczniki interpretowane przez procesor XSL-FO. Stanowią one informację dotyczącą m.in. formatu strony pliku wynikowego oraz rozmieszczenia poszczególnych elementów na stronie. Elementy XSL-FO korzystają z przestrzeni nazw `fo`.

3.1 Dokument XSL-FO

Przykładowa struktura prostego dokumentu FO wygląda następująco:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
    <fo:region-body  margin="2cm"/>
  </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="A4">
 <fo:flow flow-name="xsl-region-body">
      <fo:block>
            Treść dokumentu
      </fo:block>
 </fo:flow>
</fo:page-sequence>

</fo:root>
```

Elementem głównym jest `<fo:root>`, który zawiera deklarację przestrzeni nazw `fo`. Następnie, w obrębie znacznika `<fo:layout-master-set>`, może być zdefiniowany jeden lub więcej formatów stron `<fo:simple-page-master>` dokumentu wyjściowego, w którym m.in. ustala się jaki obszar strony mają zajmować poszczególne regiony i jakie mają być ich marginesy. Zdefiniowano pięć regionów: `region-body`, `region-before`, `region-after`, `region-start` i `region-end`, odpowiadające: obszarowi centralnemu strony, nagłówkowi, stopce, lewemu i prawemu marginesowi. **Uwaga:** cztery ostatnie z wymienionych fragmentów nie są samodzielne i stanowią podobszary regionu `body`.

Po definicji formatów stron umieszczana jest właściwa zawartość dokumentu, ujęta wewnątrz sekwencji stron `<fo:page-sequence>`, odwołującej się, poprzez atrybut `master-reference`, do zdefiniowanego uprzednio formatu strony. Treść zasadnicza dokumentu, która może obejmować wiele stron (np. tekst książki), jest umieszczona wewnątrz elementu `<fo:flow>`, którego atrybut `flow-name` określa docelowy region strony. Wewnątrz znacznika `<fo:page-sequence>` może występować tylko jeden taki element. Zawartość, która pozostaje niezmienna na wszystkich stronach, np. tekst nagłówka, umieszcza się w granicach znacznika `<fo:static-content flow-name="nazwa_regionu">`. Zawartość regionów podzielona jest na bloki `<fo:block>`. Wewnątrz bloków można definiować także linie tekstu oraz fragmenty linii tekstu.

Dokumenty XSL-FO mogą być napisane ręcznie, jednakże zazwyczaj do ich tworzenia stosuje się transformacje XSLT, które m.in. umożliwiają formatowanie danych zawartych w plikach XML.

3.2 Procesor XSL-FO

Obecnie jest dostępny jest szeroki wybór oprogramowania służącego do przetwarzania dokumentów XSL-FO na wydruki PDF i wiele innych formatów plików. Jednym z darmowych, a jednocześnie oferującym szerokie możliwości, formaterów wydruku, jest Apache FOP. Został on napisany w języku Java, co sprawia, że jest niezależny od systemu operacyjnego. Program jest uruchamiany z linii poleceń. Konwersja pliku XSL-FO do PDF jest wywoływana następująco:

```
>fop plik_wejściowy.fo plik_wyjściowy.pdf
```

Apache FOP ma także zintegrowany procesor XSLT, zatem możliwe jest dokonanie transformacji XSLT i XSL-FO przy użyciu jednego polecenia:

```
>fop -xml plik_z_danymi.xml -xsl plik_transformacji.xsl -pdf plik_wyjściowy.pdf
```

Uwaga: Należy upewnić się, że zmienna środowiskowa PATH zawiera ścieżkę do folderu z instalacją środowiska Javy, np. „C:\Program Files\Java\jre7\bin”

Domyślny zestaw czcionek Apache FOP jest niewielki i nie umożliwia poprawnego wyświetlania polskich znaków w plikach PDF. Jednakże, po dodaniu do pliku konfiguracyjnego fop.xconf następującej treści:

```
<directory>ścieżka_do_folderu_czcionek_systemowych (np. C:\Windows\Fonts)
</directory>
<auto-detect/>
```

wewnątrz znacznika <fonts>, umożliwia korzystanie z większości czcionek dostępnych dla systemu operacyjnego. Program należy uruchamiać z dodatkowym parametrem –c :

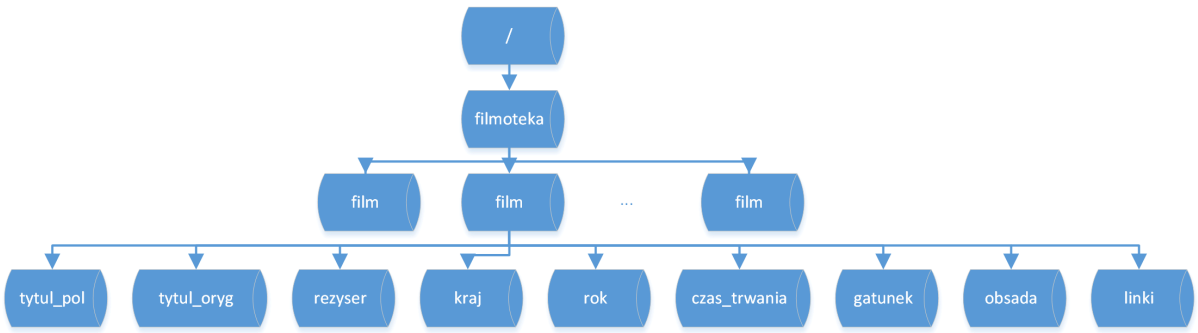
```
>fop -c ścieżka_do_pliku_fop.xconf
```

Podstawowe komendy wiersza poleceń Windows (cmd.exe):
- przejście na inny dysk lub partycję po wpisaniu **`x:`** gdzie x jest literą przypisaną do dysku
- przejście do folderu wyższego poziomu **`cd ..`**
- przejście do innego folderu podrzędnego **`cd nazwa_folderu`**

4. Ćwiczenia

Na potrzeby wykonania ćwiczeń laboratoryjnych, przygotowany został zestaw plików, w którego skład wchodzi m.in.:
- filmoteka.xml – plik XML z bazą danych zawierającą informacje o kilkunastu filmach fabularnych i odwołaniem do arkusza XSLT,
- filmoteka.xsl – plik transformacji XSLT rozwijany w trakcie zajęć laboratoryjnych,
- filmoteka_fo.xsl – arkusz transformacji z XML do XSL-FO,
- style.css – prosty arkusz stylów CSS,
- folder fop-1.1 zawierający odpowiednio skonfigurowany program Apache FOP.

Struktura danych zawartych w pliku filmoteka.xml wygląda następująco:



Zadania do wykonania

Poniżej wyszczególniono zadania, które należy zrealizować w ramach zajęć laboratoryjnych. Punktacja jest podana przy treści zadania i została uzależniona od jego złożoności. Całkowita liczba punktów, możliwa do uzyskania ze wszystkich zadań, wynosi 10.
Przed przystąpieniem do wykonywania zadań należy otworzyć pliki XML, XSL i HTML w edytorze, aby zapoznać się z ich zawartością. Wszystkie pliki powinny być zapisane w folderze D:\Imie_Nazwisko (bez spacji i polskich znaków).

XSLT

Po otwarciu w przeglądarce WWW pliku filmoteka.xml, zostanie wyświetlona strona zawierająca napis „Filmoteka" oraz nagłówek tabeli, zawierający nazwy wszystkich ośmiu elementów zagnieżdżonych w elemencie `<film>`. Poniżej znajdują się polecenia, których wykonanie zmodyfikuje pierwotny wygląd strony. Wygląd strony wynikowej w każdym punkcie można obejrzeć w pliku „Wyniki_transformacji.pdf"
Uwaga: **O ile nie jest to powiedziane wprost, nie należy używać pętli `<xsl:for-each>`.**
Uwaga 2: Pliki wynikowe po każdym kroku zapisz pod inną nazwą, np. po wykonaniu trzeciego kroku zapisz plik filmoteka3.xsl. Aby wyświetlić wynik tej transformacji w przeglądarce, zmień odwołanie w pliku XML `<?xml-stylesheet type="text/xsl" href="filmoteka3.xsl"?>`.

1. Utworzyć w pliku filmoteka.xsl nowy szablon nazwany o nazwie `separator`, którego wywołanie spowoduje wstawienie do tabeli nowego wiersza o ośmiu komórkach zawierających poziomą linię (`<hr/>`). Wywołać utworzony szablon w szablonie głównym `<template match="/">`. (0.5 pkt)
2. Utworzyć szablon z dopasowaniem do elementu `<film>`, wstawiający w tabeli wiersz o ośmiu komórkach, w których zostaną wyświetlone wartości elementów potomnych elementu `<film>`. Wywołać utworzony szablon w szablonie głównym, przed wstawionym wcześniej wywołaniem szablonu `separator`. (1 pkt)
3. Zastosować szablon z dopasowaniem w taki sposób, aby najpierw zostały wyświetlone w tabeli filmy polskie, a następnie wiersz separatora i filmy zagraniczne. *Wskazówka: zakres dopasowanych elementów* `<film>` *ograniczyć poprzez odpowiednie sformułowanie atrybutu* `select`. (0.5 pkt)
4. Zmodyfikować szablon z dopasowaniem tak, aby były wyświetlane tylko informacje o filmach dłuższych niż 120 minut. Należy użyć `<xsl:if>`. (0.25 pkt)
5. Usunąć warunek wprowadzony w poprzednim punkcie. Stosując element `<xsl:choose>` zmodyfikować szablon z dopasowaniem tak, aby czas trwania filmu był zapisany czcionką o kolorze uzależnionym od jego wartości:
   - krótszy niż 120 minut – kolor #00cc00 (zielony),
   - dłuższy niż 150 minut – kolor #cc0000 (czerwony),
   - pozostałe – kolor #e6b800 (żółty).
   Wskazówka: zmodyfikować atrybut „style" znacznika <td>, style="color:#_____"
   (1.5 pkt)
6. Wyświetlić w tabeli wszystkie filmy z filmoteki, bez podziału na polskie i zagraniczne, posortowane rosnąco po roku produkcji. Należy użyć `<xsl:sort>` w pętli `<xsl:for-each>`. **Uwaga**: W tym przypadku nie są używane szablony utworzone w punktach 1 i 2. Wyświetlanie informacji o filmie należy przenieść do szablonu głównego. (1 pkt)

7. Otworzyć ponownie plik wynikowy z piątego kroku (filmoteka5.xsl). Dodać do tabeli pierwszą kolumnę z nagłówkiem „Lp.”. Używając elementu `<xsl:number>` wyświetlić listę wszystkich filmów, bez podziału na polskie i zagraniczne, opatrzoną numerami porządkowymi umieszczonymi w dodanej pierwszej kolumnie. (1 pkt)
8. Dołączyć w sekcji nagłówka HTML odwołanie do arkusza CSS „style.css”. Zwrócić uwagę na zmiany w obramowaniu tabeli. Zmodyfikować plik CSS tak, aby tło strony miało kolor #abcdef. (0.5 pkt)
9. Przy napisie "Filmoteka" wyświetlić obrazek, którego adres znajduje się w atrybucie *grafika* elementu `<filmoteka>`. W tym celu utworzyć szablon dla atrybutu grafika. W szablonie, aby wyświetlić grafikę należy utworzyć znacznik HTML-owy `<img>` i w jego atrybucie `src` podać zawartość atrybutu grafika z pliku XML. Można np. wykorzystać `xsl:element` do utworzenia znacznika HTML `<img>` oraz `xsl:attribute` do utworzenia jego odpowiednich atrybutów (`src, width, height`):
```
<xsl:element name="nazwa_tworzonego_elementu">
  <xsl:attribute name="nazwa_1_atrybutu_dla_elementu">
    wartość atrybutu 1    (np. <xsl:value-of select="wezel-
_w_ktorym_jest_adres_grafiki"/>)
  </xsl:attribute>
  <xsl:attribute name="nazwa_2_atrybutu_dla_elementu">
    wartość atrybutu 2
  </xsl:attribute>
wartość elementu (jeśli potrzebna)
</xsl:element>
```
lub można wykorzystać poniższą konstrukcję:
```
<nazwa_tworzonego_elementu nazwa_atrybutu="{wezel_w_drzewie_dokumentu}"/>
```
(1 pkt)

10. Dodać do tabeli kolumnę „Linki”, w której należy wyświetlić adresy stron związanych z filmem. Linki mają być aktywne (po najechaniu kursorem umożliwiają wejście na stronę). W tym celu należy stworzyć szablon dla elementu *link*. W szablonie utworzyć znacznik HTML <a> z atrybutem `href`, którego wartością będzie zawartość atrybutu *adres* znacznika `link` z pliku XML. Utworzenie znacznika `<a>` z atrybutami odbywa się analogicznie jak utworzenie znacznika `<img>` (patrz pkt.9). (1 pkt)

XSL-FO

11. Wygenerować plik PDF przy użyciu programu Apache FOP z linii poleceń:
```
ścieżka_do_folderu_fop\fop-1.1>fop -c fop.xconf -xml ..\filmoteka.xml -xsl
..\filmoteka_fo.xsl -pdf ..\filmoteka.pdf
```
i wyświetlić utworzony plik PDF. (0.5 pkt)
12. Zmodyfikować plik filmoteka_fo.xsl w taki sposób, aby u dołu strony dokumentu PDF drukowany był tekst „Wygenerowano: *data_czas*”, gdzie *data_czas* oznacza aktualną datę i czas systemowy. Stopka powinna mieć wysokość 0.5cm a tekst być pisany czcionką Calibri 10 punktów. Do wygenerowania informacji o czasie należy użyć przygotowanego szablonu nazwanego `dataCzas`. (1 pkt)
13. W pliku filmoteka.xml skopiować pięciokrotnie zawartość elementu `<filmoteka>` i zaobserwować jak wygląda wygenerowany dokument PDF o kilku stronach. (0.25 pkt)

Etapy oceniania: p. 1-4, p. 5-7, p. 8-10, p. 11-13. Po dokonaniu końcowej oceny przez prowadzącego, usuń wszystkie pobrane i utworzone przez Ciebie pliki!