Technologie internetowe

Paweł Rajba

pawel@ii.uni.wroc.pl
http://www.kursy24.eu/

Zawartość modułu

- Wprowadzenie do XML
- Składnia, znaczniki i atrybuty
- XML Schema, DTD
- XSL
  - XPath
  - XSLT
  - XSL-FO

Na podstawie kursów ze stron:
http://www.w3schools.com/schema/default.asp
http://www.w3schools.com/xpath/default.asp
http://www.w3schools.com/xsl/default.asp
http://www.w3schools.com/xslfo/default.asp

Kilka uwag wstępnych o XML

- XML to eXtensible Markup Language
- XML nie ma zastąpić HTML, oba języki służą do innych celów
  - XML służy do opisu danych
  - HTML skupia się na wyglądzie danych
- Do opisu struktury dokumentu XML używamy
  - DTD
  - XML Schema
  - RelaxNG

Kilka uwag wstępnych o XML

- XML niczego nie robi – służy tylko do opisu i składowania informacji
- XML, jak nazwa wskazuje jest rozszerzalny, czyli możemy tworzyć własne znaczniki
- XML nie ma żadnych predefiniowanych znaczników – wszystko tworzymy od zera

Przykłady

- Pierwszy prosty dokument

```
<?xml version="1.0" encoding="utf-8"?>
<mail>
<to>pawel@ii.uni.wroc.pl</to>
<from>zenon@ii.uni.wroc.pl</from>
<subject>Jak leci?</subject>
<body>Jak tam w Wielkim Świecie?</body>
</mail>
```


Kilka uwag dotyczących składni

- Każdy znacznik musi mieć znacznik zamykający, a jeśli jest nieparzysty, to jest to inaczej zaznaczane
  - akapit
- Nazwy znaczników są case sensitive
- Ważna jest kolejność zagnieżdżania
- Dokument XML ma jeden znacznik będący korzeniem
- Wartości atrybutów muszą być w cudzysłowach
- Wielokrotne spacje nie są zamieniane na jedną spację
- Komentarz: <!-- komentarz -->

Kilka uwag o znacznikach

- Zależności między znacznikami są takie jak te omówione przy okazji drzewa dokumentu
- Nazwy znaczników
  - mogą zawierać litery, cyfry, -, ., _, itp.
  - muszą się zaczynać od litery lub podkreślenia
  - nie mogą się zaczynać się od słowa xml
  - nazwy nie mogą zawierać spacji

Kilka uwag o znacznikach

- ,,Dobre zwyczaje" dotyczące nazw znaczników
  - nie zaleca się używania w nazwach znaków – i .
  - nazwy powinny opisywać zawartość znacznika, ale nie powinny być przesadnie długie
    - <book_title> a nie <the_title_of_the_book>
  - litery inne niż w alfabecie angielskim są dozwolone, ale nie zalecane
  - nie powinno się używać znaku : w nazwach

Kilka uwag o atrybutach

- Atrybuty dostarczają dodatkowej informacji o znacznikach
- Można używać zarówno cudzysłowie ' i "
  - zagnieżdżamy różne np. 'Ala "ma" kota'
- Czego używać: zagnieżdżonych znaczników czy atrybutów?
  - dane mogą być przechowywane w jednym i drugim
  - znaczniki przechowują również strukturę, atrybuty nie
    - <mail date="2005-10-22>...</mail>
    - <mail><date>2005-10-22</date>...</mail>
    - <mail><date><year>2005</year>
      <month>10</month><day>22</day></year>

Kilka uwag o atrybutach

- Czego używać: zagnieżdżonych znaczników czy atrybutów?
  - zalecenie speców: używać raczej znaczników zamiast atrybutów
  - problemy z atrybutami:
    - nie mogą mieć wielu wartości, znaczniki mogą
    - trudniej je modyfikować (w przyszłych zmianach)
    - nie opisują struktury
    - trudniej nimi operować z poziomu kodu programów
    - trudniej testować wartości z poziomu DTD
  - wyjątek: atrybut id, czyli identyfikator elementów

XML Schema

- Co daje XML Schema?
  - definiuje elementy i atrybuty
  - określa dzieci elementów i ich porządek
  - określa, czy element jest pusty, czy może zawierać tekst
  - określa typy danych dla elementów i atrybutów
  - definiuje domyślne i ustalone wartości dla elementów i atrybutów

XML Schema

- Przewaga XML Schema nad DTD
  - XML Schema łatwiej rozszerzać
    - korzystać z jednych schematów w innych
    - tworzyć własne typy danych na podstawie już istniejących
  - XML Schema są bogatsze niż DTD
  - XML Schema są pisane w XML
  - XML Schema wspiera typy danych
    - można tworzyć wzorce, konwersje, ...
  - XML Schema wspiera przestrzenie nazw

Składnia XML Schema

- Proste elementy XSD
  - Opis
    - mogą zawierać tylko tekst
    - nie mogą zawierać innych elementów i atrybutów
  - Deklaracja
    - <xs:element name="xxx" type="yyy"/>
  - Wbudowane typy danych
    - xs:string,xs:decimal,xs:integer,xs:boolean,xs:date,xs:time
  - Wartości mogą być także domyślne lub ustalone
    - default="domyślna", fixed="ustalona"

Składnia XML Schema

- Proste elementy XSD, przykłady
  - <xs:element name="nazwisko" type="xs:string"/>
    <xs:element name="imie" type="xs:string"/>
    <xs:element name="wzrost" type="xs:integer"/>
    <xs:element name="data_urodzenia" type="xs:date"/>

    <nazwisko>Zabłocki</nazwisko>
    <imie>Ferdynand</imie>
    <wzrost>194</wzrost>
    <data_urodzenia>1968-03-27</data_urodzenia>
  - <xs:element name="color" type="xs:string" default="red"/>
  - <xs:element name="color" type="xs:string" fixed="red"/>

Składnia XML Schema

- Atrybuty
  - Opis
    - tylko złożone elementy mogą mieć atrybuty
    - atrybut zawsze jest ,,prostym elementem"
  - Deklaracja
    - <xs:attribute name="xxx" type="yyy"/>
  - Wartości mogą być także domyślne lub ustalone
    - default="domyślna", fixed="ustalona"
  - Wartości mogą być także wymagane lub opcjonalne
    - use="required", use="optional"

Składnia XML Schema

- Atrybuty, przykłady
  - <lastname lang="EN">Smith</lastname>

    <xs:attribute name="lang" type="xs:string"/>
  - <xs:attribute name="lang" type="xs:string" default="EN"/>
  - <xs:attribute name="lang" type="xs:string" fixed="EN"/>
  - <xs:attribute name="lang" type="xs:string" use="optional"/>
  - <xs:attribute name="lang" type="xs:string" use="required"/>

Składnia XML Schema

- Ograniczenia na wartości elementów i atrybutów
  - enumeration – określa listę wartości
  - fractionDigits – określa maksymalną ilość cyfr po przecinku
  - length – określa dokładną długość lub liczbę elementów
  - maxExclusive – określa wartość maksymalną na liczby (<)
  - maxInclusive – określa wartość maksymalną na liczby (<=)
  - maxLength – określa maksymalną długość lub liczbę elementów (<=)
  - odpowiednio maxExclusive, minExclusive, minLength
  - pattern – określa wzorzec
  - totalDigits – określa dokładną liczbę cyfr
  - whiteSpace – określa sposób traktowania białych znaków (LF, CR, tab, space)

Składnia XML Schema

- Ograniczenia na wartości, przykłady
  - <xs:element name="age">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="100"/>
      </xs:restriction>
    </xs:simpleType>
    </xs:element>
  - <xs:element name="letter">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="[a-z]"/>
      </xs:restriction>
    </xs:simpleType>
    </xs:element>

Składnia XML Schema

- Złożone elementy
  - Takie, które zawierają inne elementy i/lub atrybuty
  - Są cztery rodzaje złożonych elementów
    - puste elementy
    - elementy zawierające tylko inne elementy
    - elementy zawierające tylko tekst
    - elementy zawierające zarówno inne elementy jak i tekst
  - Każdy z powyższych rodzajów może zawierać atrybuty

Składnia XML Schema

- Przykłady złożonych elementów w XML z kolejnych kategorii
  - <product pid="1345"/>
  - <employee>
    <firstname>Jan</firstname>
    <lastname>Kowal</lastname>
    </employee>
  - <food type="przysmaki">Budyń</food>
  - <description>Kolizja miała miejsce
    <date lang="en">03.03.99</date> ....</description>

Składnia XML Schema

- Mamy dany kod XML
  - <employee>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
    </employee>
- Definicję w XML Schema możemy zrobić na kilka różnych sposobów.

Składnia XML Schema

- Wersja 1
  - <xs:element name="employee">
    <xs:complexType>
    <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
    </xs:complexType>
    </xs:element>

Składnia XML Schema

- Wersja 2
  - <xs:element name="employee" type="personinfo"/>

    <xs:complexType name="personinfo">
     <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
  - Ten sposób jest bardziej elegancki, ponieważ można typ personinfo wykorzystać wielokrotnie

Składnia XML Schema

- Przykład wykorzystania jednego typu w drugim
  - ```
    <xs:element name="employee" type="fullpersoninfo"/>
    <xs:complexType name="personinfo">
      <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="fullpersoninfo">
      <xs:complexContent>
        <xs:extension base="personinfo">
          <xs:sequence>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    ```


    ```

# Składnia XML Schema

- Definiowanie pustego znacznika
  - ```
    <xs:element name="product">
      <xs:complexType>
        <xs:attribute name="prodid" type="xs:positiveInteger"/>
      </xs:complexType>
    </xs:element>
    ```
  - ```
    <xs:element name="product" type="prodtype"/>

    <xs:complexType name="prodtype">
      <xs:attribute name="prodid" type="xs:positiveInteger"/>
    </xs:complexType>
    ```


Składnia XML Schema

- Specyfikatory(?) (ang. indicators)
  - Porządkowe
    - All
    - Choice
    - Sequence
  - Ilości wystąpień
    - maxOccurs
    - minOccurs
  - Grupy
    - Group name
    - attributeGroup name

Składnia XML Schema

- Specyfikator all
  - określa, że jego dzieci mogą wystąpić w dowolnym porządku
  - każde jego dziecko musi wystąpić dokładnie raz
    - ```
      <xs:element name="person">
        <xs:complexType>
          <xs:all>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
          </xs:all>
        </xs:complexType>
      </xs:element>
      ```


      ```

# Składnia XML Schema

- Specyfikator choice
  - określa, że może wystąpić dokładnie jedno z jego dzieci
    - <xs:element name="person">
      <xs:complexType>
      <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
      </xs:choice>
      </xs:complexType>
      </xs:element>

 

# Składnia XML Schema

- Specyfikator sequence
  - określa, że jego dzieci muszą wystąpić w ustalonym w definicji porządku
    - ```
      <xs:element name="person">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      ```


      ```

# Składnia XML Schema

- Specyfikatory ilości wystąpień: minOccurs, maxOccurs
  - określają minimalną i maksymalną ilość wystąpień dzieci
  - jeśli nie chcemy dawać ograniczenia, możemy użyć
    - maxOccurs="unbounded"
  - domyślnie wartości obu specyfikatorów wynoszą 1
    - 
```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
                  maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Składnia XML Schema

- Specyfikator group
  - pozwala utworzyć grupę elementów, do której można się później odwoływać

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>
<xs:element name="person" type="personinfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```


Składnia XML Schema

- Specyfikator attributeGroup
  - Pozwala utworzyć grupę atrybutów

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="person">
  <xs:complexType>
    <xs:attributeGroup ref="personattrgroup"/>
  </xs:complexType>
</xs:element>
```


Składnia XML Schema

- Przykłady
  - osoby2.xml, osoby2.xsd
  - produkty1.xml, produkty1.xsd
  - produkty2.xml, produkty2.xsd

DTD

- Służy do definiowania schematu dokumentu XML
- Kłopot ze sprawdzaniem typów wartości w znacznikach i typów wartości atrybutów

DTD

- Deklaracja DTD w dokumencie
  - Wewnętrzne DTD:
    - <!DOCTYPE nazwa-DTD [podzbior-wewnętrzny]>
      - nazwa-DTD – nazwa DTD, powinna być taka sama jak nazwa głównego dokumentu
      - pozdbiór-wewnętrzny – zawiera lokalne deklaracje elementów, atrybutów i encji
    - Jeśli opis całej struktury jest wewnątrz dokumentu XML, to dokument taki jest autonomiczny
      - objawia się to w następujący sposób <?xml version="1.0" standalone="yes"?>

DTD

- Deklaracja DTD w dokumencie
  - Zewnętrzne DTD
    - Identyfikator systemowy
      - <!DOCTYPE nazwa SYSTEM "http://i.pl/plik.dtd">
      - <!DOCTYPE nazwa SYSTEM "../plik.dtd">

DTD

- Deklaracja DTD w dokumencie
  - Zewnętrzne DTD
    - Identyfikator publiczny – oficjalne DTD
      - <!DOCTYPE nazwa PUBLIC "identyfikator" "sciezka/dtd">
      - Identyfikator ma następującą budowę:
        rodzaj-rejestracji // właściciel // opis // język
        - rodzaj-rejestracji – to znak (+) jeśli właściciel został zarejestrowany zgodnie z ISO9070; zwykle tak nie jest i dajemy znak (-)
        - właściciel – nazwa właściciela, np. nazwisko, firma
        - opis – opis, można używać spacji
        - język – identyfikator języka zgodnie z ISO639 (2 znaki)
      - Dobrze jest dać potem adres do dokumentu DTD
      - Przykład:
        <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">

Składnia DTD

- Deklaracje elementów
  - <!ELEMENT nazwa kategoria>
  - <!ELEMENT nazwa (zawartosc)>
    - kategoria to:
      - EMPTY – pusty znacznik np.
      - ANY – zawartość znacznika może być dowolna (parsowalna); raczej niestosowane
    - zawartość to:
      - #PCDATA – oznacza dane tekstowe; nie może zawierać elementów podrzędnych
      - nazwy innych elementów

Składnia DTD

- Deklaracje elementów
  - Dodatkowe reguły przy tworzeniu definicji elementu
    - + – oznacza powtórzenie elementu jeden lub więcej razy
    - * – oznacza powtórzenie elementu zero lub więcej razy
    - ? – oznacza pojawienie się elementu raz lub nie pojawienie w ogóle
    - , – oznacza, że elementy muszą występować w określonej kolejności
    - | – oznacza alternatywę, czyli jeden z wielu

Składnia DTD

- Deklaracje atrybutów
  - <!ATTLIST element atrybut typ domyslna_wartosc>
    - element – nazwa elementu, dla którego definiujemy atrybut
    - atrybut – nazwa definiowanego atrybutu
    - typ – typ wartości; wymienione na następnej stronie
    - domyslna_wartosc – szczegóły na kolejnych stronach
  - Przykład:
    - <!ATTLIST konto typ CDATA "internetowe">

Składnia DTD

- Deklaracje atrybutów
  - Typy atrybutów mogą być następujące:
    - CDATA – oznacza tekst
    - (w1|w2|...) - wartością musi być jedna z pozycji
    - ID – wartością jest unikalny identyfikator
    - IDREF – wartością jest identyfikator innego elementu
    - IDREFS – wartością jest lista identyfikatorów innych elementów
    - NMTOKEN – wartością jest poprawna w XMLu nazwa
    - NMTOKENS – wartością jest lista poprawnych w XMLu nazw

Składnia DTD

- Deklaracje atrybutów
  - Typy atrybutów mogą być następujące:
    - ENTITY – wartością może być nazwa encji
    - ENTITIES – wartością może być list nazw encji
    - NOTATION – wartościa może być nazwa notacji
    - xml: – predefiniowana nazwa xml

Składnia DTD

- Deklaracje atrybutów
  - Wartość domyślna może być następująca:
    - value – jakiś tekst
    - #REQUIRED – atrybut w elemencie musi wystąpić
    - #IMPLIED – atrybut w elemencie może wystąpić; jeśli nie wystąpi wykorzystywana jest wartość domyślna
    - #FIXED – atrybut ustaloną na sztywno wartość
      - DTD: <!ATTLIST system name CDATA #FIXED "W2K">
Poprawny XML: <system name="W2K" />
Niepoprawny XML: <system name="Linux" />
      - DTD: <!ATTLIST cena waluta NMTOKEN #FIXED "pln">
Odczytanie <cena>100</cena> jest równoważne odczytaniu <cena waluta="pln">100</cena>

Składnia DTD

- Pojęcie encji
- Encje predefiniowane
  - & ' " > <
- Kategorie encji
  - ogólnego przeznaczenia lub parametrów
  - wewnętrzne lub zewnętrzne
  - zewnętrzne mogą być
    - analizowane lub nieanalizowane

Składnia DTD

- Encje ogólnego przeznaczenia
  - zmienne, które definiujemy w celu utworzenie skrótu na pewien tekst
  - deklarowane w DTD, używane w DTD lub XML
  - Przykład
    - <!ENTITY plik "nazwa_pliku">
      <!ENTITY link "&plik;

Składnia DTD

- Encje parametrów
  - deklarowane i używane w DTD
  - przykład
    - <!ENTITY % boolean "(true|false) 'false'">
      <!ELEMENT tel (#PCDATA)>
      <!ATTLIST tel ulubiony %boolean;>
    - <?xml version="1.0" encoding="iso-8859-2"?>
      <!ENTITY de "Germany"> <!ENTITY it "Italy">
      <!ENTITY us "United States"> <!ENTITY pl "Poland"> ...

      <?xml version="1.0" encoding="iso-8859-2">
      <!DOCTYPE address SYSTEM "address.dtd" [
      <!ENTITY % countries SYSTEM "countries.ent"> %countries;]>
      ... <country>&pl;</country> ...

Składnia DTD

- Encje wewnętrzne
  - definiowane wewnątrz dokumentu
  - przykład
    - <!ENTITY jednostka "Instytut Informatyki">
      <!ENTITY adres "ul. Przesmyckiego 20">
    - Użycie: <author>&jednostka;&adres;</author>
- Encje zewnętrzne
  - definiowane w zewnętrznych plikach
  - przykład
    - <!ENTITY jednostka SYSTEM "http://i.pl/dtd/entities.dtd">
      <!ENTITY adres SYSTEM "http://i.pl/dtd/entities.dtd">
    - Użycie: <author>&jednostka;&adres;</author>

Przykłady

- osoby1.xml, osoby1.css
- pracownicy.xml, pracownicy-encje.ent

Kilka uwag wstępnych o XSL

- XSL – pochodzi od
  - eXtensible Stylesheet Language
- CSS – style dla HTML-a
- XSL – style dla XML-a
- XSL składa się z trzech części
  - XPath – język nawigacji dokumentów XML
  - XSLT – transformacja dokumentów XML
  - XSL-FO – język formatowania dokumentów XML

XPath

- Terminologia
  - Node – węzłem są elementy, atrybuty, tekst, itp.
    - np. <znacznik>, akapit , lang="en"
  - Wartości atomowe – węzły, które nie mają potomków
    - np. Akapit, "en",
  - Item – node lub wartość atomowa
- Terminologia zależności (jak w DOM)
  - Parent, Children, Siblings, Ancestors, Descendants

XPath

- Składnia wzorców, przykłady
  - para – dopasuje się do elementu para
  - *, @* – dopasuje się do każdego elementu/atrybutu
  - chap|appen – dopasuje się do chap i do appen
  - ol/item – dopasuje się do item, którego rodzicem jest ol
  - ol//item – dopasuje się do item, którego przodkiem jest ol
  - . – bieżący węzeł
  - .. – rodzic danego węzła

XPath

- Składnia wzorców, przykłady
  - / – root
  - @class – dopasuje się do atrybutu class
  - id("W11") – dopasuje się do elementu o id=W11
  - div[@class="appendix"]//p – dopasuje sie do p, który jest potomkiem div z atrybutem class=”appendix”
  - items/item[position()>1] – dopasuje się do item-ów, które są innych niż pierwszym dzieckiem items
  - item[position() mod 2 = 1] – analogicznie
  - para[last()=1] – dopasuje się do para, które jest jedynakiem (jedynym znacznikiem para)

XPath

- Przykłady
  - /bookstore
  - bookstore/book
  - //book
  - bookstore//book
  - //@lang
  - /bookstore/book[1]
  - /bookstore/book[price>35]/title

XPath

- Dokładne określenie lokalizacji
  - Relacje zależności: ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self
  - Ścieżka ma postać
    - /step/step/... lub step/step/...
    - każdy krok ścieżki ma postać
      - axisname::nodetest[predykat]
  - Przykłady: child::book, attribute::lang, child::*, attribute::*, ancestor::book, child::*/child::price

XPath

- Kilka przykładowych funkcji
  - compare(s1,s2) – zwraca -1, jeśli s1<s2; 0, jeśli s1=s2 i 1 jeśli, s1>s2
  - concat(s1,s2,...) - zwraca połączone napisy
    - np. concat('XPath ','is ','FUN!') // XPath is FUN
  - codepoints-to-string(int,int,...) - zwraca napis złożony z kodów będących argumentami
    - np. string-to-codepoints("Thérèse")
    - Wynik: 84, 104, 233, 114, 232, 115, 101
  - substring(string,start,len) – wycina fragment napisu
    - np. substring('Beatles',1,4) Wynik: 'Beat'

XPath

- Kilka przykładowych funkcji
  - string-length(napis) – zwraca długość napisu
  - name() - zwraca nazwę bieżącego węzła
  - index-of((item,item,...),searchitem) – zwraca numer szukanego elementu; numeracja od 1
  - count((item,item,...)) - zwraca liczbę węzłów
  - position() - zwraca numer aktualnie przetwarzanego elementu
  - last() - zwraca liczbę elementów na liście

XSLT

- Wprowadzenie
  - Korzysta z XPath
  - Służy do transformacji dokumentów XML na inne dokumenty XML, dokumenty HTML, itd.
  - Przeglądarki obsługujące XSLT:
    - Firefox 1.0.2
    - Mozilla 1.7.8
    - Netscape 8
    - Opera 8
    - Internet Explorer 6

XSLT

- Deklaracja arkusza transformacji
  - <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  - <xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
- Odwołanie do arkusza XSL w dokumencie XML
  - <?xml-stylesheet type="text/xsl" href="arkusz.xsl"?>

XSLT

- Arkusz XSL może zawierać pewną liczb reguł (template)
- Do utworzenia reguły używamy elementu
  - <xsl:template match="wartość">
    - atrybut match określa element, dla którego reguła będzie obowiązywać
    - wartość atrybutu match to wyrażenie XPath

XSLT

- Element value-of
  - Służy do pobrania wartości elementu z dokumentu XML i przekazanie jej do wyniku
  - Składnia
    - <xsl:value-of select="element">
      - select określa wyrażenie XPath
  - Przykład
    - <xsl:value-of select="/dostawa/produkt/nazwa">

XSLT

- Element for-each
  - Służy do iterowania elementów z dokumentu XML
  - Składnia
    - <xsl:for-each select="element">
      - select określa wyrażenie XPath
  - Przykład
    - <xsl:for-each select="nazwa_pliku">
      <xsl:value-of select="."/>
      </xsl:foreach>

XSLT

- Element sort
  - Umożliwia sortowanie wyniku uzyskanego przez foreach
  - Składnia
    - <xsl:sort order="ascending | descending"
      case-order="upper-first | lower-first">
    - Znaczenie atrybutów zgodne z opisem
  - Podanie kilku elementów sort realizuje sortowanie zagnieżdżone

XSLT

- Element if
  - Umieści treść w wyniku, jeśli będzie spełniony określony warunek
  - Składnia
    - <xsl:if test="warunek"> treść </xsl:if>
  - Przykład
    - <xsl:if test=". > 10">Robimy na zielono</xsl:if>

XSLT

- Element choose
  - Podobny do instrukcji switch z języka C
  - Składnia
    -
```
<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```


XSLT

- Element apply-templates
  - Przetwarza dzieci znacznika
  - Składnia
    - <xsl:apply-templates select="element">
      - Podanie opcjonalnego atrybutu select spowoduje przetwarzanie tylko określonych elementów

XSLT

- Do utworzenia elementu wraz atrybutami mamy znaczniki element i attribute
  - Przykład
    - XSL:
      ```
      <xsl:element name="napis">
       <xsl:attribute name="kolor">czerwony</xsl:attribute>
       Wielki napis
      </xsl:element>
      ```
    - Wynik:
      ```
      <napis kolor="czerwony">Wielki napis</napis>
      ```


XSLT

- Do utworzenia komentarza mamy znacznik comment
  - Przykład
    - XSL:
      <xsl:comment>To jest komentarz</xsl:comment>
    - Wynik:
      <!--To jest komentarz-->
- Do uworzenia tekstu mamy znacznik text
  - Przykład
    - <xsl:text>To będzie zwykły kawałek tekstu</xsl:text>

XPath i XSLT

- Przykłady
  - produkty3.xml, produkty3.xsl
  - osoby3.xml, osoby3.xsl
  - osoby4.xml, osoby4.xsl

XSL-FO

- Służy do formatowania elementów w XML
- Jest pod pewnym względem podobne do CSS

XSL-FO

- Szablon dokumentu
  - <?xml version="1.0" encoding="ISO-8859-2"?>

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

<fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
    <!-- Page template goes here -->
  </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="A4">
  <!-- Page content goes here -->
</fo:page-sequence>

</fo:root>
```


XSL-FO

- W XSL FO również występuje tzw. box model
- Kolejne poziomy obszarów
  - strony (page)
  - regiony (region)
  - bloki (block)
  - wiersze (line)
  - zawartość wierszy (inline)

XSL-FO

- Dokument dzieli się na strony
  - w wydruku jest to jedna lub więcej stron
  - na ekranie jest to jedna wielka strona
- Każda strona dzieli się na regiony
  - region-body (treść strony)
  - region-before (nagłówek)
  - region-after (stopka)
  - region-start (lewy panel nawigacyjny)
  - region-end (prawy panel nawigacyjny)

XSL-FO

- Każdy region zawiera bloki
- Każdy blok może zawierać inne bloki lub wiersze
- Wiersze zawierają jakąś zawartość, zwykle po prostu tekst.

XSL-FO

- Jak to dokładnie wygląda?
  - XSL-FO korzysta z <fo:page-sequence> do zdefiniowania stron wyniku
  - Każda strona wynikowa odwołuje się do ,,page master", który definiuje układ strony
  - Każda strona wynikowa ma elementy <fo:flow> określające jej zawartość
  - Kolejne strony są drukowane lub wyświetlane sekwencyjnie

XSL-FO

- Strony XSL-FO są wypełnione zawartością elementu <xsl:flow>
- Element ten ma atrybut "flow-name", który może przyjmować następujące wartości
  - xsl-region-body (odpowiednik region-body)
  - xsl-region-before (odpowiednik region-before)
  - xsl-region-after (odpowiednik region-after)
  - xsl-region-start (odpowiednik region-start)
  - xsl-region-end (odpowiednik region-end)

XSL-FO

- Wzorce stron (tzw. page masters) definiują układ stron
- Przykłady
  - <fo:simple-page-master master-name="intro">
    <fo:region-body margin="5in" />
    </fo:simple-page-master>
  - <fo:simple-page-master master-name="left">
    <fo:region-body margin-left="2in" margin-right="3in" />
    </fo:simple-page-master>
  - <fo:simple-page-master master-name="right">
    <fo:region-body margin-left="3in" margin-right="2in" />
    </fo:simple-page-master>
    - pierwszy może być używany jako wstęp, dwa kolejne do stron nieparzystych i parzystych

XSL-FO

- Do określenia rozmiaru strony mamy atrybuty
  - page-width
  - page-height
- Do określenia marginesów mamy
  - margin-top
  - margin-bottom
  - margin-left
  - margin-right
  - margin (wszystkie cztery marginesy)

XSL-FO

- Przykład układu strony A4
  - <fo:simple-page-master master-name="A4"
    page-width="297mm" page-height="210mm"
    margin-top="1cm" margin-bottom="1cm"
    margin-left="1cm" margin-right="1cm">
    <fo:region-body margin="3cm"/>
    <fo:region-before extent="2cm"/>
    <fo:region-after extent="2cm"/>
    <fo:region-start extent="2cm"/>
    <fo:region-end extent="2cm"/>
    </fo:simple-page-master>

XSL-FO

- Umieszczanie bloków, przykład
  - ```
    <fo:page-sequence>
      <fo:flow flow-name="xsl-region-body">
        <fo:block>
          <!-- Treść -->
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
    ```
- Formatowanie bloków
  - Mamy szereg właściwości znanych już z CSSa


    ```

# XSL-FO

- Tworzenie list, przykład

```
<fo:list-block>
 <fo:list-item>
  <fo:list-item-label>
    <fo:block>*</fo:block>
  </fo:list-item-label>
  <fo:list-item-body>
    <fo:block>Volvo</fo:block>
  </fo:list-item-body>
 </fo:list-item>
 <fo:list-item>
  <fo:list-item-label>
    <fo:block>*</fo:block>
  </fo:list-item-label>
  <fo:list-item-body>
    <fo:block>Saab</fo:block>
  </fo:list-item-body>
 </fo:list-item>
</fo:list-block>
```

XSL-FO

- Do tworzenia tabel mamy następujące obiekty
  - fo:table-and-caption
  - fo:table
  - fo:table-caption
  - fo:table-column
  - fo:table-header
  - fo:table-footer
  - fo:table-body
  - fo:table-row
  - fo:table-cell

XSL-FO

- Przykłady
  - xslfo-w3c.xml
  - osoby5.xml, osoby5.xsl

