

# XPath. XSLT – część 1

## XPath – XML Path Language

- Problem:
  - jednoznaczne adresowanie fragmentów struktury dokumentu XML.
- Rozwiązanie:
  - abstrakcyjny drzewiasty model struktury dokumentu,
  - normalizacja zawartości dokumentu (ten sam byt, choć różnie zakodowany, jest tak samo reprezentowany w modelu),
  - język oparty na ścieżkach w drzewie struktury.
- Status:
  - wersja 1.0 – rekomendacja W3C z 16 listopada 1999 r.,
  - wersja 2.0 – *candidate recommendation* z 3 listopada 2005.
- Zastosowania:
  - XSL,
  - XPointer,
  - ...

## XPath data model

- Modelowanie dokumentu przy pomocy drzewa:
  - węzeł *root*,
  - węzły elementów,
  - węzły atrybutów,
  - węzły tekstowe,
  - węzły instrukcji przetwarzania,
  - węzły komentarzy,
  - węzły przestrzeni nazw.
- Własności węzłów:
  - *string-value*,
  - normalizacja odwołań do encji i sekcji CDATA,
  - *expanded-name*.

## Wyrażenia XPath

- Typy wyrażen:
  - *node-set*,
  - *boolean*,
  - *number*,
  - *string*.
- Wzrost bieżący (*context node*).
- Poruszanie się w hierarchii elementów:
  - /
  - /book/section
  - section/para

## Location paths

- Ścieżka XPath złożona z kroków.
- Opis kroku:
  - oś,
  - test węzła,
  - predykaty.
- Przykłady:
  - /child::book/child::section
  - child::para[attribute::type="warning"]

## Osie (axes)

- „Kierunki” poruszania się po modelu dokumentu:
  - *child*
  - *descendant*
  - *parent*
  - *ancestor*
  - *following-sibling*
  - *preceding-sibling*
  - *following*
  - *preceding*
  - *attribute*
  - *namespace*
  - *self*
  - *descendant-or-self*
  - *ancestor-or-self*

## Testy węzłów

- Podstawowy typ węzła:
  - dla osi `attribute`: atrybut,
  - dla osi `namespace`: przestrzeń nazw,
  - dla pozostałych osi: element.
- Testy:
  - nazwa węzła,
  - \* - wszystkie węzły podstawowego typu
  - `node()`
  - `text()`
  - `comment()`
  - `processing-instruction()`
  - `processing-instruction(target-name)`

## Zapis skrócony

- Skróty:
  - `child::` można pominąć,
  - `@` `attribute::`
  - `//` `/descendant-or-self::node()` /
  - `.` `self::node()`
  - `..` `parent::node()`
- Zapis pełny vs. skrócony – przykład:
  - `//para`
  - `self::node()/descendant-or-self::node()/child::para`

## Predykaty

- Dowolne wyrażenie.
- Interpretacja:
  - `number` – prawda, gdy równy pozycji węzła w kontekście,
  - `string` – prawda, gdy niepusty,
  - `node-set` – prawda, gdy niepusty.

## Przykłady

- `para`
- `*`
- `*/para`
- `@name`
- `@*`
- `/doc/chapter[5]/section[2]`
- `chapter//para`
- `chapter[title]`
- `chapter[title="Introduction"]`
- `employee[@secretary and @assistant]`
- `//country[not(.,=preceding::country)]`

## Ważniejsze funkcje

- Operator: `+` `-` `*` `/` `>` `>=` `<` `<=` `and` `or` ...
- `last()`
- `position()`
- `count(node-set)`
- `name(node-set?)`
- `string(object?)`
- `concat(string, string, string*)`
- `contains(string, string)`
- `not(boolean)`

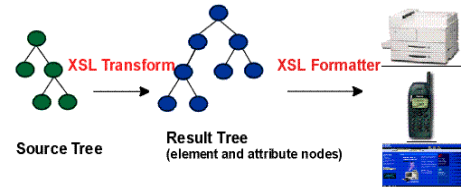
## Języki formatowania dokumentów strukturalnych

- SGML:
  - FOSI (Formatting Output Specification Instance):
    - specyfikacja MIL-PRF-28001,
    - zbyt małe możliwości dla ogólnych zastosowań.
  - DSSSL (Document Style Semantics and Specification Language):
    - ISO/IEC 10179:1996
    - oparty na podzbiorze języka Scheme bez efektów ubocznych.
- XML:
  - CSS (Cascading Style Sheets), stosowane m. in. w HTML-u,
  - XSL (Extensible Stylesheet Language):
    - język wysokopoziomowy,
    - deklaratywny, bez efektów ubocznych.

## XSL – części składowe

- XSLT (XSL Transformations):
  - język opisu przekształceń dokumentów XML,
  - składnia XML,
  - oparty na dopasowywaniu wzorców,
  - przestrzeń nazw: <http://www.w3.org/1999/XSL/Transform>,
  - wersja 1.0 – rekomendacja W3C z 16 listopada 1999 r.
- XPath (XML Path Language).
- XSL:FO (XSL Formatting Objects):
  - słownik XML-owy pozwalający definiować formatowanie,
  - przestrzeń nazw: <http://www.w3.org/1999/XSL/Format>,
  - opisany w rekomendacji XSL 1.0 z 15 października 2001 r.

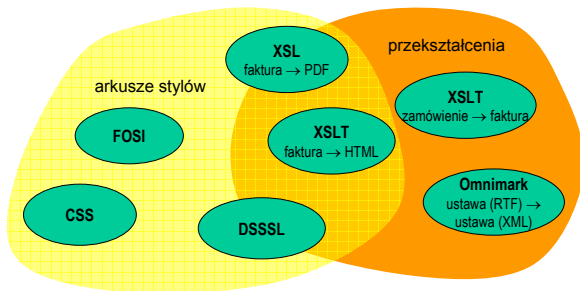
## XSL a XSLT



Result XML tree is the result of XSLT processing.

Źródło: *Extensible Stylesheet Language (XSL) Version 1.0*,  
W3C Recommendation 15 October 2001  
(<http://www.w3.org/TR/xsl/>)

## Arkusze stylów a przekształcenia



## Zasada działania przekształcenia XSLT

- Reguła XSLT:
  - wyrażenie XPath określające węzły, dla których reguła obowiązuje,
  - treść „wykonywana” w przypadku uruchomienia reguły:
    - tekst i elementy wypisywane na wyjście,
    - instrukcje XSLT.
- Sposób przetwarzania:
  - wykonaj regułę dla węzła /,
  - reguła może wywołać reguły dla innych węzłów.

## Arkusz stylów/przekształcenie XSLT

- Element główny:
 

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="..." />
  <xsl:output-method="html" />
  <xsl:param name="..." />
  ...
</xsl:stylesheet>
```
- Output methods: xml, html, text.
- Określanie arkusza stylów dla dokumentu:
 

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

## Podstawy składni – przykład

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="wiersz">
    <poemat>
      <tytul><xsl:value-of select="@tytul" /></tytul>
      <xsl:apply-templates />
    </poemat>
  </xsl:template>

  <xsl:template match="zwrotka">
    <xsl:apply-templates /><odstep />
  </xsl:template>

  <xsl:template match="wers">
    <p><xsl:apply-templates /></p>
  </xsl:template>
</xsl:stylesheet>
```

## Przekształcenie – przykład

```
<wiersz tytul="***"> <poemat><tytul>***</tytul>
  <zwrotka>
    <wers>aaa</wers> <p>aaa</p>
    <wers>bbb</wers> <p>bbb</p>
  </zwrotka> <odstep/>
  <zwrotka>
    <wers>ccc</wers> <p>ccc</p>
    <wers>ddd</wers> <p>ddd</p>
  </zwrotka> <odstep/>
</wiersz> </poemat>
```

## Wbudowane reguły

```
<xsl:template match="*/"/>
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="processing-instruction()|comment()"/>
```

## Generowanie dokumentu wyjściowego

- Elementy i tekst literalnie podane w przekształceniu.
- Instrukcje generujące:
  - `<xsl:value-of select="string-expression"/>`
  - `<xsl:element name="..." />`
  - `<xsl:attribute name="..." />`
  - `<xsl:text/>`
  - `<xsl:processing-instruction name="..." />`
  - `<xsl:comment/>`
  - `<xsl:copy/>`

## Szablony wartości atrybutów

- W określonych atrybutach można wstawiać wyrażenia XPath w nawiasach klamrowych {}.
- ```

<xsl:element name="h{count(ancestor::*)}"/>
```

| Element XSLT                            | Atrybuty                                                            |
|-----------------------------------------|---------------------------------------------------------------------|
| literalny element dokumentu wyjściowego | dowolny atrybut spoza przestrzeni nazw XSLT                         |
| element                                 | name, namespace                                                     |
| attribute                               | name, namespace                                                     |
| number                                  | level, count, from, format, lang, grouping-separator, grouping-size |
| sort                                    | order, lang, data-type, case-order                                  |
| processing-instruction                  | name                                                                |

## Przetwarzanie warunkowe: if

```
<xsl:template match="item">
  <tr>
    <xsl:if test="position() mod 2 = 0">
      <xsl:attribute name="bgcolor">yellow</xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
  </tr>
</xsl:template>
```

## Przetwarzanie warunkowe: choose

```
<xsl:template match="orderedlist/item">
  <xsl:variable name="level"
    select="count(ancestor::orderedlist) mod 3"/>
  <xsl:choose>
    <xsl:when test='$level=1'>
      <xsl:number format="i"/>
    </xsl:when>
    <xsl:when test='$level=2'>
      <xsl:number format="a"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:number format="1"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>. </xsl:text>
<xsl:apply-templates>
</xsl:template>
```

## Pętle

```
<xsl:template match="index">
  <h1>Index</h1>
  <xsl:for-each select="//keyword">
    <p><xsl:value-of select="text()" /></p>
  </xsl:for-each>
</xsl:template>
```

## Gdzie szukać dalej

- The Extensible Stylesheet Language Family:
  - 🔗 [www.w3.org/Style/XSL](http://www.w3.org/Style/XSL)
- XSLT Tutorial:
  - 🔗 [www.zvon.org/xxl/XSLTutorial/Output](http://www.zvon.org/xxl/XSLTutorial/Output)
- TopXML:
  - 🔗 [www.topxml.com/xsl](http://www.topxml.com/xsl)
  - 🔗 [www.topxml.com/xslstylesheets](http://www.topxml.com/xslstylesheets)

