XML Path Language (XPath)

- **Cel - adresowanie elementów /części dokumentu XML**

- składnia podobna do URI
- wyszukiwanie elementów bądź grup elementów
- dokument jako drzewo
- typy węzłów: element, attribute, text
- wyrażenie (expression) jako podstawowa konstrukcja składniowa

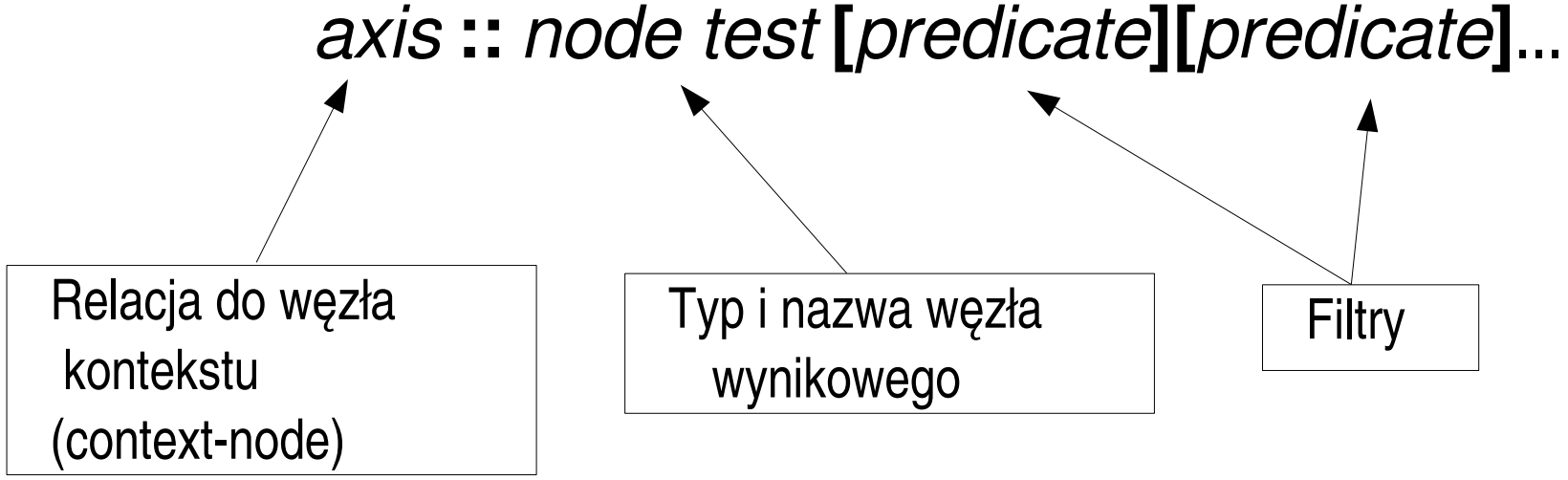
Wynik ewaluacji wyrażenia (obiekt)

Kontekst ewaluacji wyrażenia = *węzeł (context node) + pozycja (context position) + rozmiar (context size) + dowiązania zmiennych (variable bindings) + biblioteka funkcji (function library) + deklaracje przestrzeni nazewniczych (namespace declarations)*

Location Paths - szczególny przypadek wyrażeń

Element ścieżki składa się z:

- osi (axis)
- testu węzła (node test)
- predykatów (predicates)




```
child::para[position()=1]
```

```
/child::doc/child::chapter[position()=5]/child::section
```

```
/descendant::olist/child::item
```

Osie w Location Paths

- child
- descendant
- parent
- ancestor (zawsze zawiera element-korzeń)
- following-sibling, preceding-sibling
- following, preceding (bez potomków, atrybutów i przestrzeni nazw)
- attribute (principal node - atrybut)
- namespace (principal node - namespace)
- self
- descendant-or-self
- ancestor-or-self

Testy węzłów

- Element pasuje, jeśli jego typ się zgadza z podstawowym typem osi oraz nazwa jest równa podanej
- Symbol * oznacza “każdy węzeł”
- text(), comment(), processing-instruction(), node()

Składnia skrócona

- **child** jest osią domyślną
- **attribute::** może być przedstawiony jako **@**
- // jest równoważne /**descendant-or-self::node()**/
- **.** oznacza **self::node()**
- **..** oznacza **parent::node()**
- **[x]** jest równe **[position()=x]**

Przykłady:

- *para*
- *text()*
- *@name*
- *@**
- **/para*
- */doc/chapter[5]/section[2]*
- *para[@type="warning"]*
- *employee[@secretary and @assistant]*
- *../@lang*
- *chapter//para*
- *para[last()]*
- *.//para*

Core Function Library (wybrane funkcje)

- Zbiory węzłów

last() - zwraca context-size,
position() - zwraca context-position,
count(node-set) - liczebność zbioru węzłów
local-name(node-set), name(node-set), namespace-uri(node-set)
id(object)

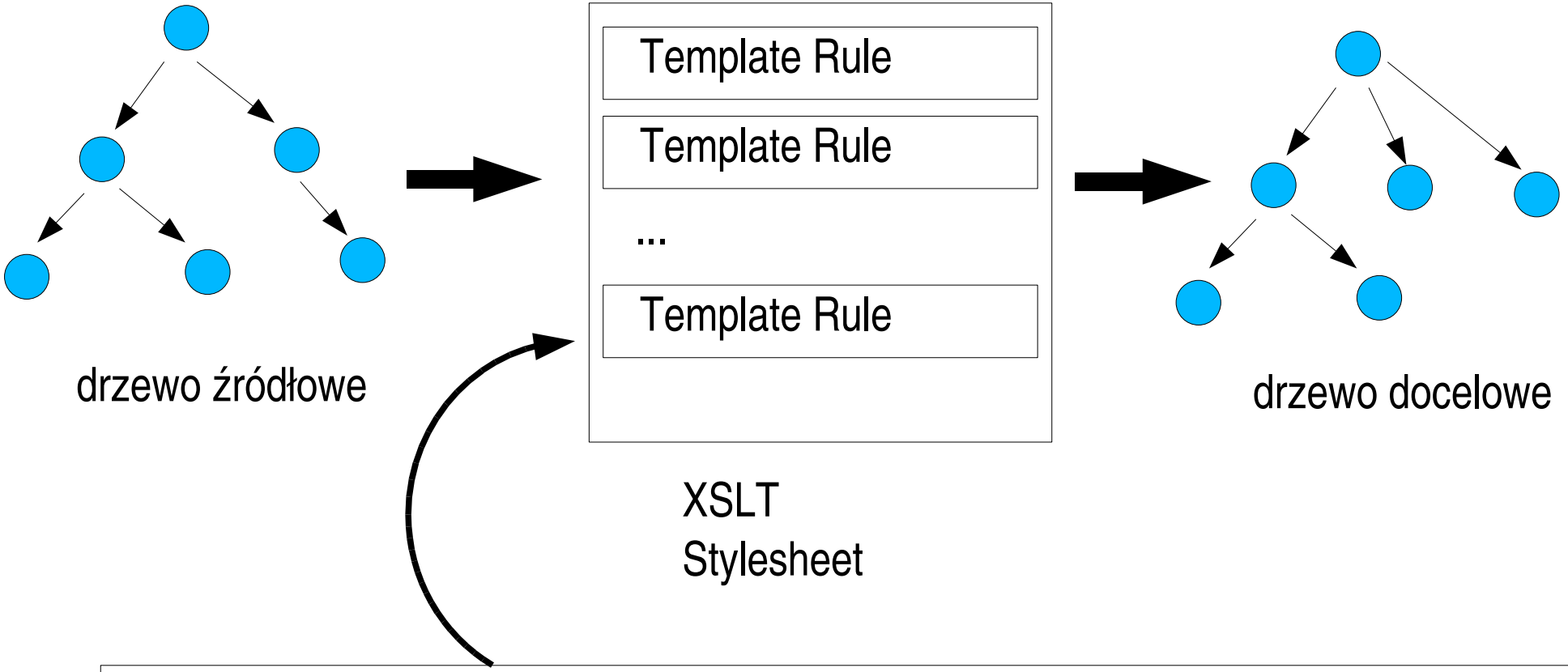
id("ala")
id("ala")/child::para[position()=5]

- Napisy

string(object), concat(string,string,string*), starts-with(string,string),
contains(string,string), substring(string,number,number?),
string-length(string?)

- Wyrażenia logiczne
- Wyrażenia arytmetyczne

XSL Transformations



Template Rule

Wzorzec (Pattern)
Definiowany przy użyciu XPath

Szablon (Template)
Opisuje fragment drzewa docelowego

Stylesheet

- Dokument taki zawiera “zwykłe” elementy XML oraz elementy z przestrzeni nazewniczej *http://www.w3.org/1999/XSL/Transform*

- *stylesheet*

```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:import href="..."/>
 <xsl:include href="..."/>
 <xsl:strip-space elements="..."/>
 <xsl:preserve-space elements="..."/>
 <xsl:output method="..."/>
 <xsl:key name="..." match="..." use="..."/>
 <xsl:decimal-format name="..."/>
 <xsl:namespace-alias stylesheet-prefix="..." result-
 prefix="..."/>
 <xsl:attribute-set name="..."> ... </xsl:attribute-set>
 <xsl:variable name="...">...</xsl:variable>
 <xsl:param name="...">...</xsl:param>
 <xsl:template match="..."> ...  </xsl:template>
 <xsl:template name="...">    ...  </xsl:template>
</xsl:stylesheet>
```

Składnia uproszczona - Literal Result Element


```
<html xsl:version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns="http://www.w3.org/TR/xhtml1/strict">
  <head>
    <title>Tutaj tytuł</title>
  </head>
  <body>
    <p>Podsumowanie: <xsl:value-of select="wydatki/razem"/></p>
  </body>
</html>
```

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:template match="/">
<html>
  <head>
    <title>Tutaj tytuł</title>
  </head>
  <body>
    <p>Podsumowanie: <xsl:value-of select="wydatki/razem"/></p>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Wybrane pomocnicze elementy XSLT

- xsl:include (top-level)

```
<xsl:include  href = uri-reference />
```

- xsl:import (top-level)

```
<xsl:import  href = uri-reference />
```

- xsl:strip-space / xsl:preserve-space (top-level)

```
<xsl:strip-space  elements = tokens />
<xsl:preserve-space  elements = tokens />
```

- xsl:variable / xsl:param

```
<xsl:variable name="para-font-size">12pt</xsl:variable>
<xsl:template match="para">
 <fo:block font-size="{$para-font-size}">
   <xsl:apply-templates/>
 </fo:block>
</xsl:template>
```

Przetwarzanie dokumentu

- Model przetwarzania
- Szablon

```
<xsl:template  match = pattern  name = qname  priority = number
  mode = qname>
  <!-- Content: (xsl:param*, template) -->
</xsl:template>

<xsl:apply-templates  select = node-set-expression
  mode = qname>
  <!-- Content: (xsl:sort | xsl:with-param)* -->
</xsl:apply-templates>
```

Przykłady:

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

```
<xsl:template match="ksiazka">
  <p align=”center”>
  <xsl:apply-templates
          select=".//rozdzial"/>
  </p>
</xsl:template>
```

Wstawianie nowych elementów

- Tag

```
<xsl:element  name = { qname }  namespace = { uri-reference }
  use-attribute-sets = qnames>
  <!-- Content: template -->
</xsl:element>
```

- Atrybut

```
<xsl:attribute  name = { qname }  namespace = { uri-reference }>
  <!-- Content: template -->
</xsl:attribute>
```

- Tekst

```
<xsl:text  disable-output-escaping = "yes" | "no">
  <!-- Content: #PCDATA -->
</xsl:text>
```

- Komentarz

```
<xsl:comment>
  <!-- Content: template -->
</xsl:comment>
```

Wstawianie nowych elementów cd

- Wartość wyliczona

```
<xsl:value-of select = string-expression
  disable-output-escaping = "yes" | "no" />
```

```
<xsl:template match="osoba">
 <p>
  <xsl:value-of select="@imie"/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="@nazwisko"/>
 </p>
</xsl:template>
```

```
<xsl:template match="ksiazka">
 <p>
  Ilość rozdziałów:
  <xsl:value-of
        select="count(rozdzial)"/>
 </p>
</xsl:template>
```

- Pętla

```
<xsl:for-each  select = node-set-expression>
  <!-- Content: (xsl:sort*, template) -->
</xsl:for-each>
```

- Instrukcja warunkowa (if,choose,when,otherwise)

```
<xsl:if  test = boolean-expression>
  <!-- Content: template -->
</xsl:if>
```