

## Definiowanie wzorców

W każdym pliku XSL możemy definiować szablony dla różnych elementów.

### Szablony z atrybutem match:

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

[szablon dla elementu root - należy pamiętać aby w każdym pliku XSL umieścić szablon obejmujący całe drzewo XML]

]

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

[szablon dla elementu pozycja zawartego wewnątrz znacznika zestawienie]

Wywoływanie tak zdefiniowanych szablonów:

```
<xsl:apply-templates select="zestawienie/pozycja"/>
```

[szablon zostanie zastosowany do elementów pozycja]

```
<xsl:apply-templates select="zestawienie/pozycja[@id='polskie']"/>
```

[szablon zostanie zastosowany dla elementów pozycja posiadających atrybut id równy 'polskie']

```
<xsl:apply-templates select="zestawienie/pozycja[2]"/>
```

[szablon zostanie zastosowany dla drugiego wystąpienia elementu pozycja]

### Szablony z atrybutem name:

```
<xsl:template name="pozycja">
...
</xsl:template>
```

[szablon o nazwie "pozycja"]

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

[call-template nie zmienia pozycji w drzewie XML]

**Tworzenie węzłów w dokumencie wyjściowym**

```
<xsl:element name="a">
  <xsl:attribute name="href">
    zdjecie.jpg
  </xsl:attribute>
</xsl:element>
```

[zapis ten tworzy element "a" z atrybutem "href" równym "zdjecie.jpg"]

```
<xsl:element name="a">
  <xsl:attribute name="href">
    <xsl:value-of select="pozycja/zdjecie"/>
  </xsl:attribute>
</xsl:element>
```

[element "a" z atrybutem "href" równym wartości elementu "zdjecie" zawartego w elemencie "pozycja"]

**Sortowanie**

```
<xsl:sort select="pozycja/numer"/>
```

[jako wartość atrybutu "select" podajemy wzorec wg którego mają zostać posortowane wyniki]

```
<xsl:sort select="pozycja/numer" data-type="number"/>
```

[atrybut data-type wskazuje na format wartości wg której następuje sortowanie – możliwe opcje "number" oraz "text"]

```
<xsl:for-each select="zestawienie/pozycja[@id='polskie']">
  <xsl:sort select="numer" data-type="number"/>
  ...
</xsl:for-each>
```

[jeżeli sortowania chcemy użyć w pętli for-each, element sort należy umieścić w pierwszej linii kodu wewnątrz pętli]

Sortowanie jest również możliwe przy przetwarzaniu listy z wykorzystaniem szablonu (apply-templates). Znacznik <xsl:sort /> należy wtedy zagnieździć wewnątrz <xsl:apply-templates />

## Numerowanie

```
<xsl:number/>
```

[numerowanie kolejnych węzłów – numery przypisane przed transformacją]

```
<xsl:number format="I"/>
```

[atrybut "format" decyduje o formacie liczb użytych w procesie numerowania – tu liczby rzymskie]

```
<xsl:value-of select="position()"/>
```

[numerowanie elementów w dokumencie wyjściowym (np. po zastosowaniu sortowania)]

## Funkcje (wybór)

contains(łańcuch, wzorzec)	Sprawdza czy dany łańcuch zawiera określony wzorzec
concat(string, string ...)	Łączy podane łańcuchy
string-length(string)	Zwraca długość podanego łańcucha
round(liczba)	Zaokrągla liczbę do najbliższej wartości całkowitej
sum(elementy)	Sumuje wartości zawarte w podanych elementach
count(element)	Zwraca ilość wystąpień danego elementu
position()	Zwraca pozycję elementu
current()	Zwraca wartość aktualnego elementu

## Operatory (wybór)

Operatory porównania oraz logiczne:

=	!=	&gt;	&gt;=	&lt;	&lt;=	and	or	not
---	----	------	-------	------	-------	-----	----	-----

Operatory matematyczne:

dodawanie: +	Odejmowanie: -	mnożenie: *	dzielenie: <b>div</b>
--------------	----------------	-------------	-----------------------