# XSLT
## transformácia XML dokumentov

## Transformácia XML dokumentov

eXtended Stylesheet Language - Transformations (XSLT)

- definovaná syntax a sémantika
- transformácia XML dokumentu na iný (XML dokument, textový formát, …)
- filtrovanie elementov, vkladanie nových elementov, preusporiadanie, výber dát



## Transformácia XML dokumentov (pokr.)

eXtended Stylesheet Language, XSL
- definuje štýl zobrazenia pre XML elementy
  (HTML - CSS)
- štandard W3C

dva jazyky: XSL-FO a XSL-T

**F**ormatting **O**bjects - definovanie layoutu
(zobrazenie v rôznych médiách)

**T**ransformations - transformácia dát
(XML dokument na …)



## XSLT - namespace

definovaná:
**http://www.w3.org/1999/XSL/Transform**

určuje **syntax a sémantiku** XML prvkom

pre XSLT v budúcnosti rozšíriteľná:
- množina inštrukcií
- množina preddefinovných funkcií

(ale v inej namespace)

## XSL - vloženie do XML dokumentu

processing instruction, vkladaná do XML dokumentu:

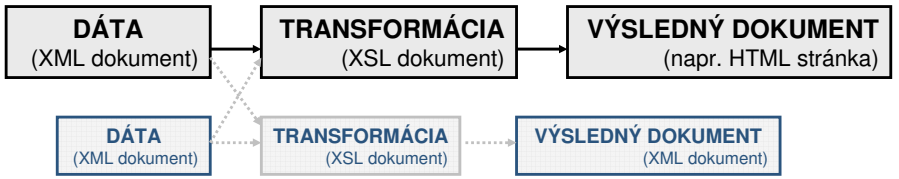
**<?xml-stylesheet type="text/xsl" href="*file-url*"?>**

výhoda:
- nie je potrebné externé riadenie

nevýhoda:
- zviazaná informácia priamo v dátach



## XSLT - Základné princípy, transformácia




- **zdojový strom** XML (source tree, document)
- definícia **transformačných pravidiel**
  dvojica **vzor** (pattern) a **predloha** (template)
- **výsledný strom** XML (result tree, document)
  odlišná štruktúra, preusporiadanie a filtrovanie dát, vkladanie nových štruktúr, …



## XSLT - Základné princípy, spracovanie

- **rekurzívne** prechádzaný zdojový strom
  (do hĺbky)
- **kontext** aktuálneho uzla
  (od koreňa stromu)
- aplikácia **pravidiel**
  (vyhľadané porovnávaním so vzormi)
  - ak nájde, aplikuje do výsledného stromu predlohu
  - ak nenájde, pokračuje na deťoch



## XSLT - Základné princípy, transformačné pravidlo

**vzor** je porovnávaný podľa kontextu v zdrojovom strome
pri úspešnom porovnaní (matched) sú
do výsledného stromu vložené údaje
podľa **predlohy**

vzor (pattern) - výraz v jazyku XPath
predloha (template) - inštrukcie jazyka XSLT
- výstupné dáta

## Príklad

```
<?xml version="1.0"?>

<xsl:transform  version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="/">
        <HTML>
            <BODY>
                <H1>Transformed XML</H1>
                <xsl:apply-templates/>
            </BODY>
        </HTML>
    </xsl:template>

    <!-- templates for XML elements -->

</xsl:transform>
```



## Štruktúra XSLT dokumentu

koreňový element **stylesheet** / **transform**

```
<?xml version="1.0"?>

<xsl:transform  version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <!-- top-level elementy -->

</xsl:transform>
```

povinný parameter **version**, určenie namespace
deti: top-level elementy (template, variable, output, …)

*Poznámka:* existuje aj skrátený zápis



## Štruktúra XSLT dokumentu (pokr.)

Transformačné pravidlá: vzor + predloha

**vzor** - XPath expression

**predloha** - element **template**

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

    <!-- štruktúra výsledného stromu,  XSLT inštrukcie-->

</xsl:template>
```



## Tvorba výsledného dokumentu

spracovanie zvolenej predlohy:

- element z XSLT namespace
  - **interpretované**
- element z inej namespace
  - **vložený do výsledného stromu**

implicitné / explicitné riadenie rekurzie

## Predloha XSLT

**<xsl:template**

**match** = *pattern* - XPath expression
**name** = *qname* - pomenovaná predloha
**priority** = *number* - priorita
**mode** = *qname* - mód práce

**>**

**match** atribút vyžadovaný, ak neexistuje **name**

mód práce - tá istá predloha pre rôzne scenáre (release, debug a pod.)(iba pre **match**)



## Pokračovanie spracovávania

po spracovaní predlohy rekurzívny proces nepokračuje, treba explicitne uviesť:

**<xsl:apply-templates**

**select** = *node-set-expression*
**mode** = *qname*

**>**

ak **select** neuvedený, default všetky deti (inak XPath)

určuje mód vykonávania



## Príklady

```
<xsl:template match="author-group">
      <fo:inline-sequence>
            <xsl:apply-templates select="author/name"/>
      </fo:inline-sequence>
</xsl:template>

<xsl:template match="employee">
      <P>
      Employee <xsl:apply-templates select="name"/>
      in group <xsl:apply-templates select="ancestor::depart/group"/>.
      </P>
</xsl:template>
```

**!** nekonečné cykly



## Zabudované predlohy - implicitné riadenie rekurzie

nízka priorita

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

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

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

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

## Pomenované predlohy

výber predlohy priamo podľa mena

**<xsl:call-template**
**name =** *qname*
**>**

**nemení kontext**

*match*, *mode* a *priority* pre pomenovanú predlohu bez významu



## Tvorba textových uzlov

v predlohe je možné explicitne vytvoriť textový uzol:

**<xsl:text**
**disable-output-escaping =** *"yes" / "no"*
**>**
… a obsahuje #PCDATA

nahradzovanie špeciálnych znakov
- pri načítaní XSLT sa uchováva znak
- pri výstupe sa nahradzuje textovou entitou



## Získavanie dát zo zdojového stromu

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

povinný parameter **select** - XPath expression

doplnenie textového uzla

nahradzovanie špeciálnych znakov



## Získavanie dát zo zdojového stromu - príklady

XML dáta

```
<product type="car">
      <name>Skoda Filicia</name>
</product>
```

transformácia

```
<xsl:template match="product">
      <P>
            <xsl:value-of select="name"/>
            <xsl:text > is a </xsl:text>
            <xsl:value-of select="@type"/>
            <xsl:text >.</xsl:text>
      </P>
</xsl:template>
```

výsledok

```
<P>Skoda Felicia is a car.</P>
```

## Tvorba inštrukcií (processing instruction)

**<xsl:processing-instruction**
**name =** *{ qname }* - interpretované
**>**
… predloha pre obsah inštrukcie

```
<xsl:processing-instruction name="Lang">SK</xsl:processing-instruction>
<?Lang SK?>
```

*Poznámka:* hlavička XML dokumentu inak



## Tvorba komentárov

**<xsl:comment**
**>**
… predloha pre obsah komentáru

```
<xsl:comment>This file is generated. Do not edit!</xsl:comment>
<!--This file is generated. Do not edit!-->
```

predloha obsahuje text
nesmie obsahovať ‘ -- ’, alebo ‘ - ’ na konci



## Vytváranie elementov výsledného stromu

niektorá charakteristika elementu (meno, …) je známa až počas spracovávania

**<xsl:element**
**name =** *{ qname }* - interpretované
**namespace =** *{ uri-reference }* - interpretované
**use-attribute-sets =** *qnames*
**>**
…a predloha pre atribúty a deti elementu



## Pridávanie atribútov

atribút môže byť vytvorený pre novovytvorený element a tak isto aj pre elementy z XSLT (iná namespace)

**<xsl:attribute**
**name =** *{ qname }* - interpretované
**namespace =** *{ uri-reference }* - interpretované
**>**
…a predloha hodnoty atribútu

**chyby:** ak už vložené dieťa, v hodnote (predlohe) vytvárané iné než text

## Príklad

```
<?xml version="1.0"?>
<page>
        <ptitle>Welcome!</ptitle>
        <text>
                About me: <ref to="roman.html">Roman</ref>.
        </text>
</page>
```

```
<!--Generated page. Do not edit!-->
<HTML>
        <HEAD>
                <TITLE>Welcome!</TITLE>
        </HEAD>
        <BODY>
                About me: <A HREF="roman.html">Roman</A>.
        </BODY>
</HTML>
```



## Príklad

```
<?xml version="1.0"?>
<xsl:transform  version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
                <xsl:comment>Generated page. Do not edit!</xsl:comment>
                <HTML>
                        <HEAD>
                                <TITLE><xsl:value-of select="/page/ptitle"/></TITLE>
                        </HEAD>
                        <BODY>
                                <xsl:apply-templates/>
                        </BODY>
                </HTML>
        </xsl:template>
        <xsl:template match="page">
                <xsl:apply-templates select="text"/>
        </xsl:template>
        <xsl:template match="ref">
                <A>
                        <xsl:attribute name="HREF">
                                <xsl:value-of select="@to"/>
                        </xsl:attribute>
                        <xsl:value-of select="."/>
                </A>
        </xsl:template>
</xsl:transform>
```



## Úloha

```
<pridelenia>
        <oznacenie>počítač - pracovník</oznacenie>
        <skratka>PC</skratka>
        <dvojice>
                <dvojica>
                        <cislo>1</cislo>
                        <pracovnik>šéf Jožo</pracovnik>
                </dvojica>
                <dvojica>
                        <cislo>2</cislo>
                        <pracovnik>asistentka Aňa</pracovnik>
                </dvojica>
                <dvojica>
                        <cislo>100</cislo>
                        <pracovnik>vývojár Jano</pracovnik>
                </dvojica>
        </dvojice>
        <popis>
                Číslo počítača a pracovník, ktorému je pridelený.
        </popis>
</pridelenia>
```

```
<HTML>
<BODY>
        <H1>Pridelenia (počítač - pracovník)</H1>
        <P>
                Číslo počítača a pracovník, ktorému je pridelený.
                <TABLE>
                        <TR>
                                <TD>PC1</TD>
                                <TD>šéf Jožo</TD>
                        </TR>
                        <TR>
                                <TD>PC2</TD>
                                <TD>asistentka Aňa</TD>
                        </TR>
                        <TR>
                                <TD>PC100</TD>
                                <TD>vývojár Jano</TD>
                        </TR>
                </TABLE>
        </P>
</BODY>
</HTML>
```



## Riadiace konštrukcie jazyka XSLT

spracovanie XSLT dokumentu:

- začína od /
- rekurzívne vyhľadávanie vzoru
- implicitný alebo explicitný krok

opakovanie (iterácia, cyklus)

**for-each**

podmienené spracovanie (vetvenie, výber)

**if** , **choose**

## Iteračný cyklus

užitočný ak je jasná výsledná štruktúra
(definuje anonymnú predlohu)

**<xsl:for-each**
**select =** *node-set-expression*
**>**
… a predloha pre každý vybraný uzol

kontext sa mení (kontextový uzol)



## Iteračný cyklus - príklad (rekurzia)

```
<xsl:template match="dvojice">
    <TABLE>
        <xsl:apply-templates/>
    </TABLE>
</xsl:template>
<xsl:template match="dvojica">
    <TR>
        <TD>
            <xsl:value-of select="/pridelenia/skratka"/>
            <xsl:value-of select="cislo"/>
        </TD>
        <TD>
            <xsl:value-of select="pracovnik"/>
        </TD>
    </TR>
</xsl:template>
```



## Iteračný cyklus - príklad

```
<xsl:template match="dvojice">
    <TABLE>
    <xsl:for-each select="dvojica">
        <TR>
            <TD>
                <xsl:value-of select="/pridelenia/skratka"/>
                <xsl:value-of select="cislo"/>
            </TD>
            <TD>
                <xsl:value-of select="pracovnik"/>
            </TD>
        </TR>
    </xsl:for-each>
    </TABLE>
</xsl:template>
```



## Podmienené spracovanie - if

jednoduché jednovetvové spracovanie
(definuje predlohu iba pre splnenú testovanú podmienku)

**<xsl:if**
**test =** *boolean-expression*
**>**
… a predloha pri splnení testu

## Podmienené spracovanie - if (príklady)

mená poodeľované čiarkou

```
<xsl:template match="namelist/name">
    <xsl:apply-templates/>
    <xsl:if test="not (position() = last())">, </xsl:if>
</xsl:template>
```

každý druhý riadok zvýraznený modrou farbou

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



## Podmienené spracovanie - choose

viacvetvové podmienené spracovanie

(vykoná sa jediná (prvá platná) vetva / alebo "inak")

**<xsl:choose**

**>** obsah: (xsl:when**+**, xsl:otherwise**?**)

**<xsl:when**

**test =** *boolean-expression*

**>** … a predloha pri splnení testu

**<xsl:otherwise**

**>** … a predloha pri splnení testu



## Podmienené spracovanie - choose (príklad)

```
<xsl:template match="orderedlist/listitem">
    <fo:list-item indent-start='2pi'>
        <fo:list-item-label>
            <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>
        </fo:list-item-label>
        <fo:list-item-body>
            <xsl:apply-templates/>
        </fo:list-item-body>
    </fo:list-item>
</xsl:template>
```



## Usporiadanie uzlov - sort

zoradenie vybratých uzlov inak ako document order

platí pre **xsl:apply-templates** a **xsl:for-each**

viacúrovňové usporiadanie je možné podľa zadaného poradia uzlov **sort**

**<xsl:sort**

**select =** *string-expression*
**data-type =** *{ "text" | "number" }* - interpretované
**order =** *{ "ascending" | "descending"}* - interpretované
**lang =** *{ nmtoken}* - interpretované
**case-order =** *{ "upper-first"|"lower-first"}* - interpretované

**>**

## Usporiadanie uzlov - sort (pokr.)

**<xsl:sort**
**select =** *string-expression*
**data-type =** *{ "text" | "number" }*
**order =** *{ "ascending" | "descending"}*
**lang =** *{ nmtoken}*
**case-order =** *{ "upper-first" | "lower-first"}*
**>**

kľúč usporiadania - **select**

vyhodnotené pre každý kontextový uzol (default výraz ' . ')

typ usporiadania - **data-type**

lexikograficky - *text* (default), alfanumericky - *number*

poradie usporiadania - **order**

vzostupne - *ascending* (default), zostupne - *descending*



## Usporiadanie uzlov - sort (pokr.)

**<xsl:sort**
**select =** *string-expression*
**data-type =** *{ "text" | "number" }*
**order =** *{ "ascending" | "descending"}*
**lang =** *{ nmtoken}*
**case-order =** *{ "upper-first" | "lower-first"}*
**>**

lexikografické (reťazcové) usporiadanie

- **lang** - jazyk (zadané / podľa prostredia)
- **case-order** - A a B b … / a A b B … (default lang)

pri rovnakých kľúčoch **zachováva** document order



## Usporiadanie uzlov - sort (príklad)

```
<xsl:template match="employees">
    <ul>
        <xsl:apply-templates select="employee">
            <xsl:sort select="name/family"/>
            <xsl:sort select="name/given"/>
        </xsl:apply-templates>
    </ul>
</xsl:template>
<xsl:template match="employee">
    <li>
        <xsl:value-of select="name/given"/>
        <xsl:text > </xsl:text>
        <xsl:value-of select="name/family"/>
    </li>
</xsl:template>
```



## Interpretovaná hodnota atribútov

atribúty označené ' { ' a ' } '

výraz je nahradený hodnotou (vrátane zátvoriek)

**nevnárané** (jediná úroveň)

znaky ' { ' a ' } ' nahradzovať dvojitým výskytom

## Interpretovaná hodnota atribútov - príklad

transformácia

```
<xsl:variable name="image-dir">/images</xsl:variable>

<xsl:template match="photograph">
      <img src="{$image-dir}/{href}" width="{size/@width}"/>
</xsl:template>
```

XML dáta

```
<photograph>
  <href>headquarters.jpg</href>
  <size width="300"/>
</photograph>
```

výsledok

```
<img src="/images/headquarters.jpg" width="300"/>
```



## Premenné a parametre

definujú **meno** a jemu priradenú **hodnotu**

**<xsl:variable**
**name =** *qname*
**select =** *expression*
**>** … predloha

**<xsl:param**
**name =** *qname*
**select =** *expression*
**>** … predloha



## Premenné a parametre (pokr.)

**param** definuje default hodnotu
(môže byť zmenená pri použití **transform**ácie alebo predlohy)

hodnota môže byť ľubovoľného typu
- obsah elementu (fragment výsledného stromu)
- výsledok vyhodnotenia výrazu **select**

referenecia použitím znaku ‘ **$** ’

pri použití je definovaná platnosť (viditeľnosť)



## Premenné a parametre - na najvyššej úrovni stromu

definované ako deti **transform** / **stylesheet**
- platné pre celý strom transformácie
- hodnota získaná v kontexte koreňa dokumentu

**param** určuje default hodnoty pre transformáciu
(zmena hodnoty je definovaná požitým softvérom)

```
<xsl:param name="font-size">12pt</xsl:param>

<xsl:template match="paragraph">
      <P><FONT SIZE="{$font-size}">
            <xsl:apply-templates/>
      </FONT></P>
</xsl:template>
```

## Premenné a parametre - súčasť predlohy

platnosť pre nasledujúcich súrodencov a ich potomkov
- **variable** ľubovoľne v rámci **template**
- **param** len ako dieťa **template** (pred inými deťmi)

prekrytie predchádzajúcich definícií
(z vyšších úrovní, nesmú sa prekrývať mená na tej istej úrovni)
- lokálne premenné



## Odovzdávanie parametrov predlohám

pre **apply-templates**, **call-template**

**<xsl:with-param**
**name =** *qname*
**select =** *expression*
**>** … predloha

vyhodnocované v kontexte vyššej inštrukcie



## Odovzdávanie parametrov predlohám - príklad

```
<xsl:template name="numbered-block">
    <xsl:param name="format">1. </xsl:param>
    <fo:block>
        <xsl:number format="{$format}"/>
        <xsl:apply-templates/>
    </fo:block>
</xsl:template>
<xsl:template match="ol//ol/li">
    <xsl:call-template name="numbered-block">
        <xsl:with-param name="format">a. </xsl:with-param>
    </xsl:call-template>
</xsl:template>
```



## Doplnkové funkcie - prehľad vybraných

*node-set* **document(***object*, *node-set?***)**
ak prvý parameter ako URI, načíta externý XML dokument (vhodné uložiť do premennej)

*node-set* **current()**
aktuálny uzol (pri vyhodnocovaní výrazov)

```
<xsl:apply-templates select="//glossary/item[@name=current()/@ref]"/>
```

*string* **generate-id(***node-set?***)**
vráti unikátny reťazec pre prvý uzol z argumentov (implementačne závislé)

## Výstup

nastavenie parametrov výstupného dokumentu

**<xsl:output**
**method =** *“xml” | “html” | “text”*
**indent =** *“yes” | “no”*
**version =** *nmtoken*
**encoding =** *string*
**standalone =** *“yes” | “no”*
**omit-xml-declaration =** *“yes” | “no”*
**>**

element iba na najvyššej úrovni (dieťa koreňa)



## Výstup (pokr.)

výstup z transformácie mimo výsledného stromu
hlásenie (testovanie, kontrola práce a pod.)

**<xsl:message**
**terminate =** *“yes” | “no”*
**>** … predloha

implementačne závislé (logfile, dialógové okno, …)

**terminate** - XSLT procesor by mal ukončiť transformáciu
(default “*no*”)



## Výstup - hlásenia (príklad)

XML resources/*L*.xml

```
<messages>
  <message name="problem">A problem was detected.</message>
  <message name="error">An error was detected.</message>
</messages>
```

transformácia

```
<xsl:param name="lang" select="en"/>
<xsl:variable name="messages"
              select="document(concat('resources/', $lang, '.xml'))/messages"/>

<xsl:template name="localized-message">
    <xsl:param name="name"/>
    <xsl:message >
        <xsl:value-of select="$messages/message[@name=$name]"/>
    </xsl:message>
</xsl:template>

<xsl:template name="problem">
    <xsl:call-template name="localized-message"/>
        <xsl:with-param name="name">problem</xsl:with-param>
    </xsl:call-template>
</xsl:template>
```



## XSLT - opakovanie

**transformácia** zdrojového (**dátového**) stromu na **výstupný** strom

(úloha pre XSLT procesor)

**transformačné pravidlá**: vzor - predloha

**rekurzívne spracovanie**
- rozšírené o riadiace štruktúry **iterácie** a **podmieneného spracovania**
- aktuálny **kontext**

## XSLT - opakovanie (pokr.)

**document order**, usporiadanie (**sort**)

**premenné** a **parametre**

doplnkové **funkcie**

rozšírenia (extensions)



# Ďakujem za pozornosť



## Použitá (a študijná) literatúra

www.w3.org
oficiálne stránky WWW Consortia
(odporúčania sú na adresách namespace technológie)

www.w3schools.com
tutoriály k väčšine web-technológií

www.zvon.org
český server - veľa materiálov, tutoriály