INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Realizováno za finanční podpory ESF a státního rozpočtu ČR v rámci v projektu *Zkvalitnění a rozšíření možností studia na TUL pro studenty se SVP* reg. č. CZ.1.07/2.2.00/29.0011

Definice vzhledu

Prezentace XML

- XML popisuje strukturu dat, neřeší vzhled
- definice vzhledu:
  - **CSS** – jednoduchá varianta
  - **XSL** – vyvinuto pro XML, možnosti výrazně přesahují oblast prezentace

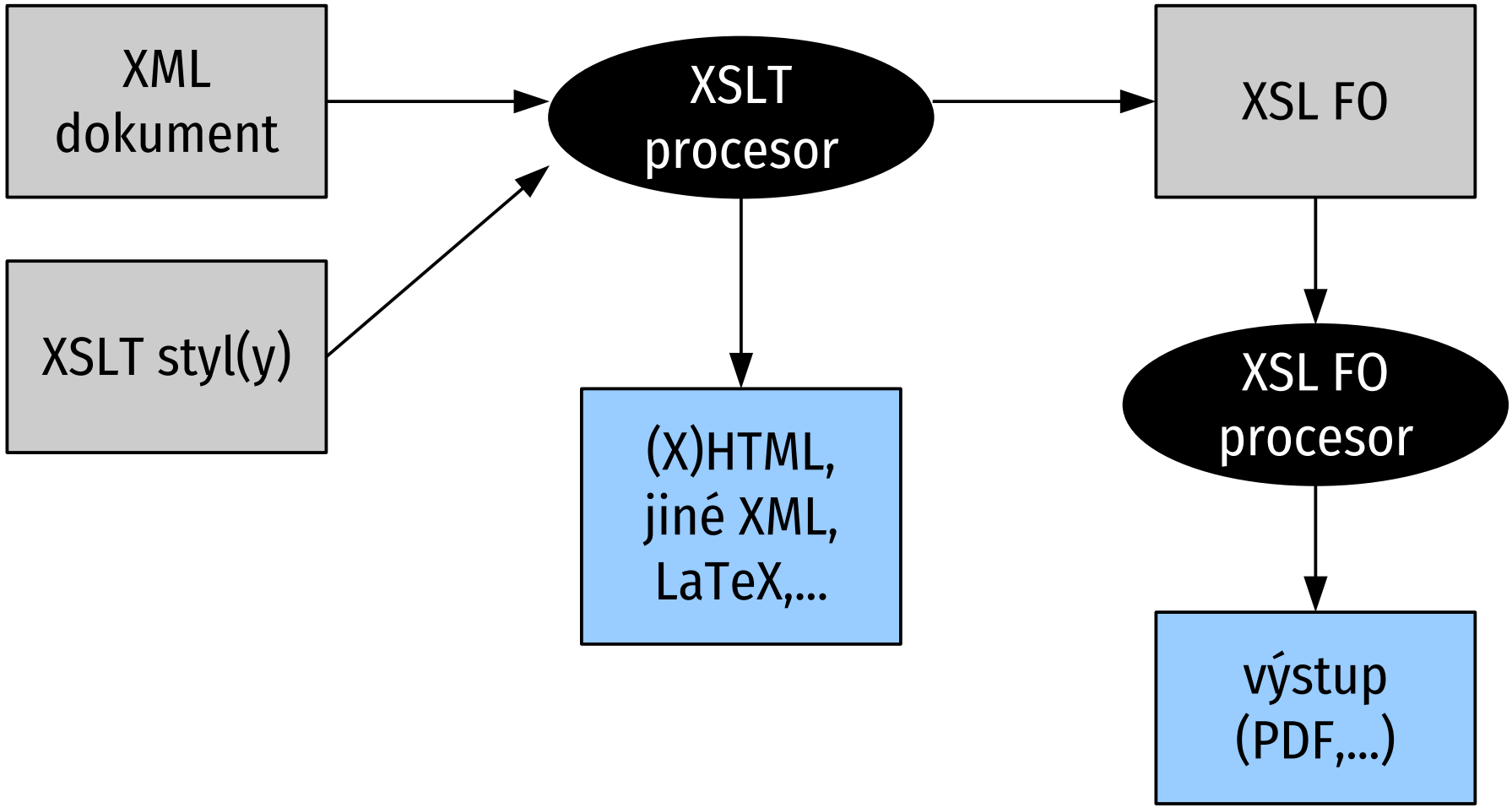
CSS

- Cascading Style Sheets
- vyvinuto pro definici vzhledu HTML stránek
- použitelné i pro XML
- vložení do XML dokumentu – instrukce pro zpracování xml-stylesheet:
  <?xml-stylesheet type="text/css" href="*styl.css*"?>
  na začátku (za identifikací verze XML)

XSL

- eXtensible Stylesheet Language
- původně určeno pro prezentaci XML dokumentů
- později rozděleno na dvě části:
  - **XSL FO (XSL Formatting Objects)** – jazyk definující formátovanou podobu dokumentu; po zpracování XSL FO procesorem vede k finální podobě
  - **XSLT (XSL Transformations)** – jazyk pro transformaci XML dokumentů, do XML FO, ale i jiných formátů

Formátování XML



XSL FO (1)

- popisuje formátování XML dokumentu, zejména pro stránkovaná média
- jmenný prostor http://www.w3.org/1999/XSL/Format
- dva klíčové prvky:
  - **<fo:layout-master-set>** definuje předlohy stránek
  - **<fo:page-sequence>** určuje obsahovou náplň stránek (s odkazy na předlohy)

XSL FO (2)

- stále poměrně abstraktní:

```
<fo:page-sequence>
 <fo:flow flow-name="xsl-region-body">
  <fo:block font-size="14pt" color="red">
   Tady bude vlastní text bloku.
  </fo:block>
 </fo:flow>
</fo:page-sequence>
```

- malý zájem – WWW Consortium dále nerozvíjí, nahrazeno CSS3-paged media

XSLT

Charakteristika XSLT

- XML jazyk
- slouží k transformaci dokumentu
  - změna struktury a/nebo prvků
  - přeuspořádání, přidávání, výběr informací
- výstupní formáty
  - XML (např. XHTML, XSL FO, ale i XML data pro jinou aplikaci)
  - HTML
  - text

XSLT procesor

- program implementující XSLT
- na základě stylového předpisu (XSLT stylu) transformuje vstupní dokument do výstupního
- interně transformuje stromy – uplatňováním šablon na vstupní strom vytváří výstupní strom

XSLT styl

- sada **šablon** definujících transformaci
- společný styl umožňuje jednotnou prezentaci řady dokumentů
- stejný dokument zpracovaný různými styly vede k diametrálně odlišným výsledkům
- příklad: DocBook poskytuje jednotný XML jazyk pro tvorbu dokumentace, XSLT styly (a doprovodné nástroje) z něj generují HTML, PDF,...

Vazba stylu a dokumentu

- teoreticky stejně jako CSS (xml-stylesheet), jen typ je jiný:
  <?xml-stylesheet type="text/xsl" href="*styl.xsl*"?>
- podporují jen některé nástroje
- procesory spouštěné z příkazového řádku někdy vyžadují uvést styl jako parametr

Jmenný prostor a obal stylu

- jmenný prostor XSLT je http://www.w3.org/1999/XSL/Transform
- celý styl je obalen prvkem stylesheet (nebo transform – synonymum)
- <xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  vlastní styl
  </xsl:stylesheet>

Šablony

XSLT šablona

- <xsl:template match="*vzor*">
  obsah_šablony
  </xsl:template>
- **vzor** určuje, na které prvky výchozího dokumentu bude použit
  - vychází z XPath, ale připouští jen omezený rozsah konstrukcí – lze child::, attribute:: a //
- **obsah_šablony** definuje, co se vloží do výstupního stromu

Příklad vstupních dat

```
<?xml version="1.0"?>
<cenik>
    <zbozi id="zb001" druh=“potravina“>
        <nazev>Houska</nazev>
        <cena>1.70</cena>
    </zbozi>
    <zbozi id=“zb004“ druh=“potravina“>
        <nazev>Voda</nazev>
        <cena>7.50</cena>
    </zbozi>
</cenik>
```

Jednoduché šablony

- **implicitní šablona**
  - provedena pro prvky, jež nevyhovují žádné šabloně
  - opíše do výstupu textový obsah prvku
  - rekurzivně prochází a transformuje jejich obsah
- **konverze prvku na jiný**
  - <xsl:template match="nazev">
    <td><xsl:apply-templates/>**</td>**
    </xsl:template>
  - **<xsl:apply-templates/>** způsobí rekurzivní procházení

Jednoduchý styl (1)

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
<xsl:template match="/">
    <html>
    <head><title>Ceník</title></head>
    <body>
    <h1>Ceník</h1>
    <xsl:apply-templates/>
    </body>
    </html>
</xsl:template>
```

Jednoduchý styl (2)

```
<xsl:template match="cenik">
    <table><xsl:apply-templates/></table>
</xsl:template>
<xsl:template match="zbozi">
    <tr><xsl:apply-templates/></tr>
</xsl:template>
<xsl:template match="nazev">
    <td><xsl:apply-templates/></td>
</xsl:template>
<xsl:template match="cena">
    <td align=“right“><xsl:apply-templates/></td>
</xsl:template>
</xsl:stylesheet>
```

apply-templates

- způsobí vyhledání a použití šablon na vyhovující uzly vstupního stromu
- řídí zpracování dokumentu, rekurzivní přístup
- atributem **select** lze omezit, jakých uzlů se bude týkat (implicitně všech dětí aktuálního uzlu)

```
<xsl:appy-templates
                select="zbozi[@druh='potravina']" />
```

Výběr šablon

- vlastní mají přednost před importovanými
- konkrétnější vzory mají přednost před obecnými
  - **obecné** – *, text(), ...
  - **obecné ve jmenném prostoru** – html:*, ...
  - **konkrétní jména** – zbozi, @druh, ...
  - **obsahující /, // nebo []** – cenik/zbozi, zbozi[@druh='potravina'], ...
- více šablon na stejné úrovni je chyba

Prvky a atributy

Generování prvků

- **opisováním**
  - obsahuje-li šablona prvky z jiného jmenného prostoru (než prostor XSLT), opíší se do výstupního stromu
- **pomocí xsl:element**
  - <xsl:element name="*jméno*">*obsah*</xsl:element>
  - zdlouhavější, ale mocnější – jméno prvku je hodnotou atributu a tu lze vytvořit podle obsahu dokumentu

Příklad obou přístupů

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

dělá totéž co

```
<xsl:template match="zbozi">
    <xsl:element name="tr">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Využívání hodnot z dokumentu

- **v atributech**
  - {*výraz*}
  - vyhodnotí výraz a přiřadí výsledek jako hodnotu atributu
- **v těle prvků**
  - <xsl:**value-of** select="*výraz*"/>
  - vyhodnotí výraz a výsledek vloží na místo svého použití

Příklad xsl:value-of

```
<zbozi id="zb001" druh="potravina">
    <nazev>Houska</nazev>
    <cena>1.70</cena>
</zbozi>
```

```
<xsl:template match="zbozi">
    <tr>
        <td><xsl:value-of select="@id"/></td>
        <xsl:apply-templates/>
    </tr>
</xsl:template>
```

```
<tr>
    <td>zb001</td>
    <td>Houska</td>
    <td align="right">1.70</td>
</tr>
```

Odvození hodnoty atributu

```
<zbozi id="zb001" druh="potravina">
    <nazev>Houska</nazev>
    <cena>1.70</cena>
</zbozi>
```

```
<xsl:template match="zbozi">
    <xsl:element name="{@druh}">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

```
<potravina>
    ...
</potravina>
```

Generování atributů

- **opisováním**
  - atributy opisovaných prvků lze zapsat přímo (viz align)
  - hodnotu lze vytvořit ze vstupních dat pomocí {}
- **pomocí xsl:attribute**
  - <xsl:attribute name="*jméno*">*hodnota*</xsl:attribute>
  - vhodné uvnitř xsl:element
  - umožňuje vytvořit i jméno atributu

Příklad

```
<xsl:template match="zbozi">
    <xsl:element name="{@druh}">
        <xsl:attribute name="ident">
            <xsl:value-of select="@id"/>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

```
<zbozi id="zb001" druh="potravina">
    <nazev>Houska</nazev>
    <cena>1.70</cena>
</zbozi>
```

```
<potravina ident="zb001">
    ...
</potravina>
```

Sady atributů

- lze si připravit sady atributů – skupiny přiřazované více prvkům
  - <xsl:attribute-set name="*jméno*">
    definice_atributů (prvky xsl:attribute)
    </xsl:attribute-set>
- lze se pak na ně opakovaně odvolávat atributem use-attribute-sets
  - hodnotou je seznam jmen sad oddělovaných mezerami

Příklad sady atributů

```
<xsl:attribute-set name="zboziAttr">
    <xsl:attribute name="ident">
        <xsl:value-of select="@id"/>
    </xsl:attribute>
    <xsl:attribute name="prodejna">Husova</xsl:attribute>
</xsl:attribute-set>

<xsl:template match="zbozi">
    <xsl:element name="{@druh}" use-attribute-sets="zboziAttr">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Kopírování částí (1)

- vhodné, pokud chcete do výsledku převzít část původního dokumentu
- **xsl:copy**
  - zkopíruje do výstupního stromu aktuální uzel (který vyhověl atributu match šablony)
  - příklad: zkopírovat zboží a zpracovat jeho obsah:

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

Kopírování částí (2)

- **xsl:copy-of**
  - prázdný prvek
  - zkopíruje vše, co vyhoví jeho atributu select
  - příklad: kompletní kopie libovolného prvku

```
<xsl:template match="*">
    <xsl:copy>
        <xsl:copy-of select="@*"/>
        <xsl:copy-of select="*"/>
    </xsl:copy>
</xsl:template>
```

Komentáře

- **generování**
  - <xsl:comment> text komentáře </xsl:comment>
  - text lze i generovat – xsl:apply-templates uvnitř
- **čtení**
  - match="comment()" v podmínce šablony
  - např. zkopírování do výstupu

```
<xsl:template match="comment()">
    <xsl:comment>
        <xsl:value-of select="."/>
    </xsl.comment>
</xsl:template>
```

Instrukce pro zpracování

- analogicky komentářům
- **generování**
  - <xsl:processing-instruction name="*cíl*">
    obsah instrukce
    </xsl:processing-instruction>
- **čtení**
  - podmínka match="processing-instruction()"
  - případně match="processing-instruction(*cíl*)"