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

Pojmenované šablony

Pojmenované šablony

- pro opakované konstrukce, často parametrizovány
- definice:
  <xsl:template name="*jméno*">
  šablona
  </xsl:template>
- použití:
  <xsl:call-template name="*jméno*" />

Parametry

- definovány konstrukcí xsl:param, definovaná hodnota se chová jako implicitní
- lze používat stejně jako proměnné, ale při volání stylu či šablony jim lze nastavit hodnotu
- nastavení globálních parametrů závisí na procesoru
- nastavení lokálních parametrů šablony:

```
<xsl:with-param name="jméno">
    hodnota
</xsl:with-param>
```

Příklad – buňka tabulky

```
<xsl:template name="bunkaTab">
   <xsl:param name="align">left</xsl:param>
   <td align="{$align}"><xsl:apply-templates/></td>
</xsl:template>
<xsl:template match="nazev">
   <xsl:call-template name="bunkaTab"/>
</xsl:template>
<xsl:template match="cena">
   <xsl:call-template name="bunkaTab">
      <xsl:with-param name="align">right</xsl:with-param>
   </xsl:call-template>
</xsl:template>
```

Krok stranou: current()

- často z dokumentu vybíráme prvky, v nichž se určitý údaj shoduje s jiným údajem aktuálního uzlu
- s výhodou lze využít XPath funkci **current()**, která vždy vrátí uzel, pro nějž byla volána šablona
- např: <utvar> obsahuje @zkratka a <nazev>, <osoba> obsahuje <pracoviste> se zkratkou útvaru, v šabloně pro osobu chceme jméno útvaru:

```
<xsl:value-of select=
"//utvar[@zkratka = current()/pracoviste]/nazev"/>
```

Číslování

Automatické číslování (1)

- čísla generuje šablona
  <xsl:number value="*hodnota*" format="*formát*"/>
- bez atributů: generuje číslo podle pozice kontextového uzlu nebo pořadí v xsl:for-each

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

Automatické číslování (2)

- hodnotou value je XPath výraz
  <xsl:number/>/<xsl:number value="count(../zbozi)"/>
- format určuje intuitivně podobu hodnoty
  - 1, 001 – arabské číslice
  - I, i – římské číslice
  - a, A – písmena
  - format="I. " – přidá k římskému číslu tečku a mezeru
- grouping-separator, grouping-size umožňují oddělovat řády (jakým znakem, po kolika)

Víceúrovňové číslování (1)

- level="multiple" čísluje hierarchicky
- count="*XPath výraz*" které uzly počítat (implicitně jen uzly stejného jména a typu jako kontextový)
-
```
<xsl:template match="section">
    <xsl:number format="1. " level="multiple"
                                count="chapter|section"/>
    <xsl:apply-templates/>
</xsl:template>
```

Víceúrovňové číslování (2)

- level=“any“ čísluje všechny prvky průběžně
- atribut from umožňuje určit, jaké prvky restartují číslování
- příklad: číslování obrázků průběžně v jednotlivých kapitolách

```
<xsl:template match="figure">
    <xsl:number format="1. " level="any" from="chapter"/>
    <xsl:apply-templates/>
</xsl:template>
```

Další vývoj

XSLT 2.0

- používá XPath 2.0
- více výstupů (více výstupních stromů):

```
<xsl:result-document href="{jmeno}.html"
                     format="xhtml">
    <xsl:value-of select="popis"/>
</xsl:result-document>
```

  - href typicky určuje jméno souboru
  - format odpovídá nějaké <output> definici

Seskupování (1)

- **<xsl:for-each-group select="*XPath výraz*" group-by="*XPath výraz*">**

  tělo

  </xsl:for-each-group>
- vybere uzly vyhovující select a seskupí je do skupin se stejnou hodnotou group-by
- pro každou skupinu jednou uplatní *tělo*

Seskupování (2)

- uvnitř <xsl:for-each-group> je k dispozici:
  - **current-group()** – členové aktuální skupiny
  - **current-grouping-key()** – (společná) hodnota group-by této skupiny
- příklad: chceme zaměstnance uspořádat po útvarech

```
<osoba id="zam1234">
    <jmeno>Satrapa Pavel</jmeno>
    <utvar>NTI</utvar>
</osoba>
```

Seskupování (3)

```
<xsl:for-each-group select="osoba" group-by="utvar">
    <h2>
        <xsl:value-of select="current-grouping-key()"/>
    </h2>
    <ul>
    <xsl:for-each select="current-group()">
        <li><xsl:value-of select="jmeno"/></li>
    </xsl:for-each>
    </ul>
</xsl:for-each-group>
```

XSLT 3.0

- standardizováno v červnu 2017
- podpora balíčků: <xsl:package> a <xsl:use-package> pro oddělenou kompilaci
- lepší podpora pro streaming (průběžné zpracování bez vytváření celého stromu v paměti)
- mapy – množiny dvojic klíč–hodnota (např. klíčem číslo, hodnotou jméno měsíce), analogie asociativních polí

Iterace (1)

- **<xsl:iterate select="XPath výraz">**
  tělo
  </xsl:iterate>
- *tělo* se vyhodnotí pro každou položku vstupní sekvence (výsledek select)
- podobá se for-each, ale ve for-each se vyhodnocuje nezávisle, lze i paralelně
- v iterate se postupuje sekvenčně, položka může připravit hodnotu pro následující (parametry)

Iterace (2)

- uvnitř lze **<xsl:next-iteration>**, která nemá žádný výstup, ale pomocí <xsl:with-param> připraví parametry pro další iteraci
  - nemá-li parametr přiřazenu hodnotu, zachová si pro příští iteraci svou stávající hodnotu
- **<xsl:break>** ukončí zpracování – žádná další položka vstupní sekvence už nebude zpracována

Příklad iterací (1)

- máme informace o změnách na účtu:

```
<zmeny>
  <zmena>10000</zmena>
  <zmena>-2100</zmena> ...
</zmeny>
```

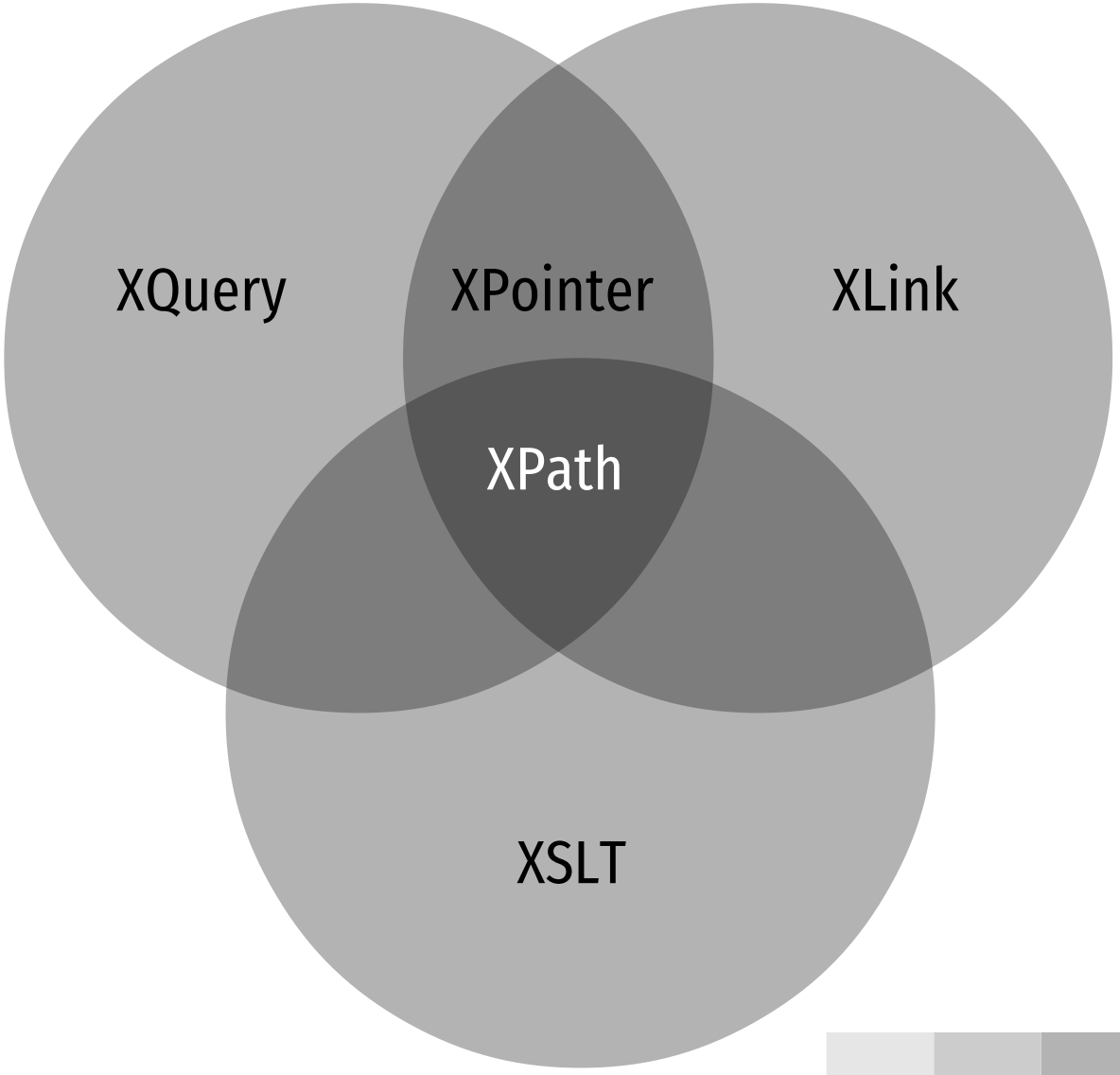
- chceme průběžné stavy účtu po změnách:

```
<ucet>
  <stav>10000</stav>
  <stav>7900</stav> ...
</ucet>
```

Příklad iterací (2)

```
<ucet>
 <xsl:iterate select="zmeny/zmena">
  <xsl:param name="stav" select="0.00" as="xs:decimal"/>
  <xsl:variable name="novystav"
                select="$stav + xs:decimal(.)"/>
  <stav><xsl:value-of select="$novystav"/></stav>
  <xsl:next-iteration>
   <xsl:with-param name="stav" select="$novystav"/>
  </xsl:next-iteration>
 </xsl:iterate>
</ucet>
```

Rodina XML konceptů



XLink

XLink

- XML Linking Language
- obecný mechanismus pro definici odkazů v XML dokumentech
- nedefinuje jména prvků, ale mechanismy (atributy), jak prohlásit libovolný prvek za odkaz a popsat jeho strukturu a chování
- jmenný prostor XLink: http://www.w3.org/1999/xlink

Jednoduchý odkaz

- základním atributem je xlink:type
  - určuje XLink typ prvku
  - hodnota simple pro jednoduché odkazy
  - atribut xlink:href obsahuje cíl
- příklad: doplníme ke zboží odkazy na stránky

```
<cenik xmlns:xlink="http://www.w3.org/1999/xlink">
<zbozi id="zb78546578"> ...
    <web   xlink:type="simple"
           xlink:href="http://www.kdesi.cz/produkty/houska.html"/>
</zbozi>
</cenik>
```

Atributy XLink

type	typ (z hlediska odkazů) dotyčného prvku
href	lokátor
title	význam odkazu (běžný text)
role, arcrole	úloha (obsahuje URI dokumentu)
show	jak prezentovat odkazovaný materiál
actuate	kdy jej prezentovat
label	definuje návěští
from, to	odkud kam vede

Atributy chování

- show – požadovaný způsob prezentace
  - new – do nového okna
  - replace – nahradit obsah stávajícího okna
  - embed – vložit do stávajícího
  - …
- actuate – kdy má dojít k následování odkazu
  - onLoad – při načtení dokumentu
  - onRequest – na žádost uživatele (kliknutí,…)
  - …

Rozšířené odkazy

- umožňují kombinovat několik zdrojů
- rodičovský prvek má xlink:type=“extended“
- obsahuje potomky typů
  - locator – externí zdroj
  - resource – interní zdroj (přímo obsažen)
  - arc – pravidla pro přechod mezi zdroji
  - title – slovní popis

Příklad

```
<vyrobek xlink:type="extended">
    <info xlink:type="locator" xlink:label="produkt"
            xlink:href="/prod/stan.xml" />
    <soucast xlink:type="locator" xlink:label="komponenta"
            xlink:href="/prod/tycka.xml"/>
    <soucast xlink:type="locator" xlink:label="komponenta"
            xlink:href="/prod/plachta.xml"/>
    <slozeni xlink:type="arc"
            xlink:from="produkt" xlink:to="komponenta"/>
</vyrobek>
```

Problém: implementace

- málo a nedokonalé
- přehled:
  http://www.w3.org/XML/2000/09/LinkingImplementations.html
- WWW klienti v podstatě nepodporují (velmi omezeně jen Mozilla & spol.)
- ostatní aplikace nevyužívají specifika odkazů

XPointer

XPointer

- XML Pointer Language
- umožňuje **odkazy na konkrétní části XML dokumentů**
- cíl: umožnit adresaci míst v dokumentu bez nutnosti jeho úpravy (nepotřebuje id)
- rozdělen do čtyř dokumentů:
  - rámec – definuje základní pravidla pro schémata
  - schémata element(), xmlns() a xpointer()

Rámec pro XPointer

- zavádí schéma jako formát odkazujících dat
- **zkrácené ukazatele**
  - obsahují jen jméno, vycházejí z XML identifikátorů (ID)
    <h2 id="instalace">Postup instalace</h2>
    XPointer: instalace
- **ukazatele založené na schématu**
  - *schéma(odkazující_data)*
    syntaxe a význam odkazujících dat závisí na schématu; může mít více částí (oddělovány prázdným místem), použije první úspěšnou část

Schéma element()

- jednoduchá základní identifikace prvků
- základem jsou čísla oddělovaná „/“ – odkazují na n-tého potomka předchozího uzlu
  element(/1/3) – třetí potomek kořenového prvku
- lze používat identifikátory
  element(zb002) je totéž co zkrácený zb002
  element(zb002/2) – druhý potomek prvku
  s identifikátorem zb002

Schéma xmlns()

- pro správnou reprezentaci jmenných prostorů v ukazatelích
- definuje prefixy, které lze používat v následujících XPointerech
  xmlns(*prefix=URI*)
- např.
  xmlns(zb=http://www.kdesi.cz/zbozi)
  a poté lze xpointer(//zb:zbozi)

Schéma xpointer()

- nejsložitější, vypracován návrh, později opuštěn
- vychází z XPath a přidává možnost adresovat řetězce a další prvky à la DOM 2
- XPath identifikuje uzly, XPointer přidává
  - bod – místo bez obsahu a potomků (např. uvnitř řetězce či mezi dvěma sousedními uzly)
  - rozsah – část mezi dvěma body

Vztah XLink a XPointer

- XLink definuje konstrukce obsahující odkazy (na jiné dokumenty či jejich části)
- XPointer lze použít jako obsah v atributech xlink:href při vytváření konkrétních odkazů
- XPointer je od části URI identifikující XML dokument oddělena znakem „#“ (jako v HTML)
- /doc/manual.xml#xpointer(id('hw')/para[3])
  #instalace