Kapitola 1. Moderní značkovací jazyky a jejich aplikace

Obsah

Úvod do XML, odkazy na specifikace

Úvod, definice, motivace, historie,…

- XML je standard (přesněji doporučení konsorcia W3C [http://www.w3.org]) jak vytvářet značkovací jazyky.
- Jedná se tedy o metajazyk.
- Ideově vychází ze zhruba o deset let mladšího standardu SGML (Structure Generalized Markup Language).
- Se základním standardem úzce souvisí několik dalších, např. XML Namespaces, XInclude, XML Base.
- Tyto spolu s dalšími standardy (XSLT, XSL-FO, XHTML, CSS...) tvoří "rodinu" standardů XML.

Jazyky rodiny SGML a jejich aplikace (HTML)

- SGML jako standard ISO 8879: http://www.iso.ch/cate/d16387.html
- Stručný "laický" úvod do SGML: http://www.sil.org/computing/noc/156ac.htm
- Další stručný, věcný ale přesný úvod do SGML: http://www.oasis-open.org/cover/naggumWhat.html
- Další Úvod do SGML: http://www.personal.u-net.com/~sgml/sgml.htm
- Srovnání SGML-XML a další zdroje: http://www.oasis-open.org/cover/sgml-xml.html

Extensible Markup Language (XML)

- World Wide Web Consortium (W3C): http://www.w3.org/
- Přehled XML aktivit W3C: http://www.w3.org/XML/Activity - specifikace standardů, konference, odkazy na SW, referenční nástroje, odkazy *(obnovováno podle potřeby)*
- XML Startkabel (EN/NL): http://xml.startkabel.nl - aktuality, odkazy *(obnovováno cca 1x týdně)*
- Zvon: http://zvon.org - asi nejlepší soubor tutoriálů, on-line referencí v mnoha jazycích, místo je hostované v ČR.
- *What is XML?* na XML.COM: http://www.xml.com/pub/a/98/10/guide0.html - jeden z úvodních článků ke XML
- The Extensible Markup Language (XML) USENET newsgroup: news:comp.text.xml - nejznámější USENET news skupina ke XML

- XML-DEV: mailto:xml-dev@xml.org - nejznámější maillist ke xml standardům
- Archív xml-dev: http://lists.xml.org/archives/xml-dev/ - archív předchozího maillistu
- XML: XML Quick Syntax Reference Card [http://www.mulberrytech.com] - výborná stručná referenční karta

Extensible Markup Language (XML) - další odkazy, zejména na software

- IBM DeveloperWorks, sekce XML: http://ibm.com/developer/xml [http://ibm.com/developer/xml/] - články, tutoriály, software atd. na vysoké technické úrovni
- IBM AlphaWorks: http://www.alphaworks.ibm.com - alpha-software fy IBM k volnému vyzkoušení
- O'Reilly XML.COM: http://xml.com - články, tutoriály atd. na vysoké technické úrovni
- Free XML Software (L. M. Garshol): http://www.garshol.priv.no/download/xmltools/ - asi nejlepší kolekce odkazů na nekomerční XML software
- XMLSoftware: http://xmlsoftware.com - asi nejlepší kolekce odkazů na obecný XML software (i komerční)
- *XML Cover Pages* na www.oasis-open.org [http://www.oasis-open.org]: xml.coverpages.org [http://xml.coverpages.org] - denně aktualizovaný soubor odkazů na články, publikace standardů, software, atd. v oblasti XML. Nejlepší zdroj v této kategorii.

Charakteristika a základní zásady XML

Charakteristika XML jazyků

- XML není jeden konkrétní značkovací jazyk; je to specifikace určující, jak mají vypadat značkovací jazyky
- jedná se tedy o "metajazyk"
- konceptuálně jde o zjednodušení SGML standardu, aby se usnadnila práce tvůrcům parserů (analyzátorů) a aplikací

  - například v tom, že každý element musí být uzavřen; pak na přečtení struktury dokumentu nemusíme mít DTD
- XML navazuje na úspěšnou implementaci SGML - jazyk HTML; má podobné charakteristiky z hlediska zaměření na internet

- Momentálně je aktuální specifikací [http://www.w3.org/TR/REC-xml] Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation 4th February 2004, François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler
- Připravuje se *XML 1.1*, nyní jako *Candidate Recommendation*.
- Vážné diskuse se vedou okolo *binárního XML*, což by měla být rovnocenná reprezentace stejného modelu, jako má "textové" XML.

Základní specifikace: XML 1.0 (Third Edition)

- Momentálně je aktuální specifikací [http://www.w3.org/TR/REC-xml] *Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation* 4th February 2004, autorů François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler
- Současně s tím je zveřejněna *XML 1.1, W3C Recommendation*, 4th February 2004, François Yergeau, John Cowan, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler.
- Vážné diskuse se vedou okolo *binárního XML*, což by měla být rovnocenná reprezentace stejného modelu, jako má "textové" XML.

Deset zásad pro specifikaci XML standardů

vyňato z preambule ke specifikaci XML 1.0 (Second Edition)

1. XML shall be straightforwardly usable over the Internet.

   XML bude přímočaře použitelné na Internetu.

2. XML shall support a wide variety of applications.

   XML bude podporovat širokou škálu aplikací.

3. XML shall be compatible with SGML.

   XML bude kompatibilní se SGML.

4. It shall be easy to write programs which process XML documents.

   Tvorba programů zpracovávajících XML bude jednoduchá.

5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

   Počet volitelných prvků XML standardu bude málo, optimálně 0.

6. XML documents should be human-legible and reasonably clear.

   XML dokumenty by měly být "lidsky" čitelné a rozumně jednoduché.

7. The XML design should be prepared quickly.

   Návrh XML standardu by měl být rychle hotov.

8. The design of XML shall be formal and concise.

   Návrh XML musí byt formální a správný.

9. XML documents shall be easy to create.

   XML dokumenty bude možné snadno vytvořit.

10. Terseness in XML literal is of minimal importance.

    Úspornost XML značkování není podstatná.

Struktura XML dokumentů

Syntaxe XML dokumentů

Základním požadavkem kladeným na *každý* XML dokument je, že musí být *dobře utvořen (well-formed)*.

Toto nastane, právě když:

1. Taken as a whole, it matches the production labeled `document`: `[1] document ::= prolog element Misc*` tj. obsahuje *prolog (hlavičku)* a právě jeden, tzv. kořenový *element*. Dále může před a po kořenovém elementu obsahovat instrukce pro zpracování, komentáře atd. (`Misc`)

2. It meets all the well-formedness constraints given in the specification.

   Musí vyhovovat všem pravidlům pro správné utvoření uvedeným ve specifikaci.

3. Each of the *parsed entities* which is referenced directly or indirectly within the document *is well-formed*.

   Totéž platí pro každou *analyzovanou (parsovanou) entitu* přímo nebo nepřímo odkazovanou v dokumentu.

Podívejte se na tutoriál základů XML v češtině [http://zvon.org/xxl/XMLTutorial/General_cze/book.html]

Rejstřík (glossary) pojmů ke XML [http://zvon.org/index.php?nav_id=173]

Fyzická a logická struktura XML dokumentu

U XML dokumentů rozlišujeme strukturu fyzickou a logickou. Aplikační programátory zajímá většinou jen struktura *logická*, autory obsahu, XML editorů, procesorů, atd. může zajímat i struktura *fyzická*.

Struktura *logická* - *dokument* členíme na *elementy* (jedne z nich je *kořenový* - *root*), jejich *atributy*, *textové uzly* v elementech, *instrukce pro zpracování*, *notace*, *komentáře*

Struktura *fyzická* - jeden logický dokument může být uložen ve více fyzických jednotkách - entitách; vždy alespoň v jedné - tzv. *entitě dokumentu* - *document entity*.

Elementy

Jsou objekty *ohraničené počáteční a koncovou značkou* - *start and end tag*, např.:

```
<tagname ...tag_attributes...>
  tag_content
</tagname>
```

Příklad 1.1. Příklad elementu s obsahem (neprázdného)

```
<body background="yellow">
   <h1>textový uzel - obsah elementu h1</h1>
   <p>textový uzel - obsah elementu p</p>
</body>
```

Elementy - prázdné

Je-li obsah prázdný (žádné dceřinné elementy ani textový obsah), pak píšeme pouze *značku prázdného elementu* - *empty element tag*, např.:

```
<tagname tag_attributes/>
```

Příklad 1.2. Příklad elementu bez obsahu (prázdného)

```
<hr width='50%'/>
```

Atributy

- Nesou "dodatečné informace" k elementu - např. jeho ID, požadované formátování - styl, odkazy na další elementy...
- Konceptuálně by bylo možné atributy nahradit elementy, ale z důvodu přehlednosti obvykle používáme obojí.
- Obsah atributu na rozdíl od obsahu elementu není nijak (na úrovni obecných zásad XML standardů)

dále strukturován.

Občas se u některých značkování vyskytne snaha o strukturaci obsahu atributů, to však obecně vede k více problémům, než řeší...

Atributy - zápis

- Atribut je tvořen *jménem* a *hodnotou*.
- Atributy zapisujeme do počáteční značky elementu (který může být i prázdný).
- Hodnota je *vždy* vložena v uvozovkách či apostrofech a od jména ji dělí znak rovnítka (=).
- Pro *názvy* atributů platí stejná omezení jako pro názvy elementů.
- V rámci jednoho elementu *nejsou* přípustné dva atributy se stejným jménem.

Atributy - příklad

Příklad 1.3. Atribut 'width' v prázdném elementu

```
<hr width='50%'/>
```

Příklad 1.4. Atribut 'border' v neprázdném elementu

```
<table border='1'>
   <tr><td>jedna</td><td>dve</td></tr>
   <tr><td>tri</td><td>ctyri</td></tr>
</table>
```

Textové uzly

Nesou textovou informaci.

Např. v následující ukázce je *text* `ahoj!` (nikoli celý element `em`!) textovým uzlem:

```
<em>ahoj!</em>
```

Instrukce pro zpracování

Instrukce pro zpracování (*processing-instruction*) píšeme do značek `<?target content>`

Informují aplikaci o postupu či nastavení nutném pro zpracování daných XML dat. Nepopisují (nepředstavují) obsah, ale *zpracování* dokumentu.

```
<?xsl-stylesheet href="mystyle.xsl">
```

Poznámka

`href` v příkladu *neznamená* atribut; atributy nejsou u instrukce pro zpracování možné!

Notace

Notace (*notation*) píšeme do značek `<!NOTATION name declaration >`

Slouží zejména k popisu binárních (non-XML) entit - např. obrázků GIF, PNG,...

Jde vlastně o *deklaraci způsobu zpracování*.

Komentáře

Podobně jako v HTML - komentář (*comment*) píšeme do značek `<!-- text komentáře -->`

Komentář nebývá obvykle pro zpracování významný, ale záleží na aplikaci - může např. uchovávat značky *Servlet-side Includes (SSI)*

Parsery by proto *měly* komentáře zpracovávat - předávat dále.

Např. *SAX parsery* však tak nečiní!!! (resp. činí až v rozšířené verzi SAX2, v Javě balík `org.xml.sax.ext`).

Entity

Entita je základní jednotkou *fyzické* stavby dokumentu. Odpovídá řetězci, souboru...

Parsery by měly entity zpracovávat tak, aby aplikace nemusela o entitách "nic tušit".

Podrobněji...

Podrobné informace k syntaxi se dozvíme v následující kapitole Standardy rodiny XML [../standards/index.html]

Specifikace XML 1.1

XML 1.0 (Third Edition)

- Původní specifikace (W3C Recommendation) XML 1.0 na W3C: http://www.w3.org/XML/
- 3rd Edition (aktualizace, ne změny) na http://www.w3.org/TR/2000/REC-xml-20001006
- výborná komentovaná verze téhož na XML.COM (Annotated XML): http://www.xml.com/pub/a/axml/axmlintro.html
- na XML 1.1 (Candidate Recommendation) [http://www.w3.org/TR/xml11/] - změny indukované zavedením *UNICODE 3*, lepší možnosti *normalizace*, upřesnění postupu manipulace se znaky *ukončení řádku*.

XML - další tutoriály a články

- (výborný úvodní) Koskův seriál o XML pro Softwarové noviny: http://kosek.cz/clanky/swn-xml/index.html
- Seriál o XML na ŽIVĚ [http://zive.cz]
- (obsahuje hodně příkladů) Zvon XML Tutorial: http://www.zvon.org/xxl/XMLTutorial/General/book_en.html
- Microsoft XML Tutorial: http://msdn.microsoft.com/xml/tutorial/
- 101 XML Tutorials: http://www.xml101.com/xml/default.asp
- XML Tutoriály na Beginners.co.uk [http://tutorials.beginners.co.uk]
- Tutoriály na Developerlife.com: http://developerlife.com

Terminologie

Terminologie

- Opakování: správně utvořený (*well formed*) dokument
- Nové: platný (*valid*) dokument

  Platný podle specifikace znamená *přísnější* omezení než správně utvořený.

  Obvykle se validitou myslí soulad s *DTD* (Document Type Definition) dokumentu.

Terminologie - opakování (2)

- uzel (element, atribut, textový uzel, instrukce pro zpracování, komentář)

- element
- atribut
- textový uzel
- instrukce pro zpracování
- komentář
- dále viz např. Koskův seriál o XML na http://kosek.cz/clanky/swn-xml/index.html

Terminologie - opakování (3)

- uzel dokumentu

  - ten je nadřazený kořenovému elementu

  - může kromě něj obsahovat též komentáře, instrukce pro zpracování, notaci DOCTYPE atd

- kořenový element

Znaky v XML dokumentech

Znaky v XML dokumentech

Specifikace povoluje na určitých místech v XML dokumentech (např. název elementu, obsah atributu...) jen některé znaky.

Vzhledem k internacionalizaci a nutnosti zvládnout i exotické jazyky je třeba znát, co se čím myslí.

Musíme rozlišovat:

- *znakové sady* (množiny znaků s pořadovými čísly), tj. přiřazení ordinální hodnoty znaku (např. Unicode) a
- *kódování znaků* (z dané sady), např. UTF-8, tj. ordinální hodnota znaku se kóduje do posloupnosti bajtů

Standardy Unicode, ISO 10646

Oba standardy se zabývají podobnými problémy: řeší znakové sady s více než 256 znaky.

- Původní návrh tzv. 16bitového Unicode: až 64 K znaků, stačí pro evropské, nestačí pro světové jazyky (např. dnes frekventovaná čínstina).

- 32bitový Unicode: pokrývá znaky už "na věky".

V současnosti se z 32bitové škály většinou používá jen tzv. Basic Multilingual Plane (BMP) pokrývající většinu jazyků.

V XML je možné pro názvy (nonterminál *kvalifikovaná jména* - QName) použít znaky z BMP.

Jinak lze v XML dokumentech používat všechny znaky Unicode.

Kódování Unicode

Všechny aplikace XML (zejména aplikace univerzální, parsery) musejí být schopny zpracovat znaky Unicode bez ohledu na kódování.

Přesto je dobré znát nejběžnější kódování:

- osmibitová, tradiční: US-ASCII, ISO-8859-2 (ISO Latin 2), Windows-1250 (=Cp1250) - kódování jen vybrané podmnožiny Unicode.
- UTF-8: kódování všech znaků Unicode, každý znak na 1-6 bajtech, US-ASCII na jednom bajtu, "čeština" na dvou.
- UTF-16: princip stejný jako UTF-8, ale základní ukládací jednotkou je dvoubajtové slovo (16 bitů)
- UCS-2: přímé kódování Unicode, čísla znaků z BMP se zapíší přímo jako dva bajty
- UCS-4: dtto, ale pro celý Unicode a na 4 bajtech - neúsporné, 4 bajty i pro US-ASCII, evropské jazyky...

Pro XML mají klíčový význam UTF kódování, zejména UTF-8 (ale parsery musejí umět obě).

Znaky v XML dokumentech

- Přípustné jsou jakékoli UNICODE znaky po x10FFFF (kromě xFFFE, xFFFF a rozmezí xD800 - xDFFF).
- *jména* (*names*) musí být složena ze nemezerových znaků: číslice, písmena, **.** (tečka) **-** (pomlčka, minus) **_** (podtržítko) **:** a dalších, musí začínat písmenem nebo **_** **:**
- Kódování těchto UNICODE znaků není podstatné.
- Jako implicitní - není-li v prologu (hlavičce), např.

  ```
  <?xml version="1.0" encoding="Windows-1250"?>
  ```

  uvedeno jinak - se používá UTF-8 nebo UTF-16.
- Rozlišení UTF-8 a UTF-16 se děje pomocí prvních dvou bajtů dokumentové entity (tj. souboru), po-

mocí tzv. byte-order-mark xFFFE

- Není-li uvedena, předpokládá se UTF-8, čili UTF-8 je implicitní kódování UNICODE znaků v XML dokumentech.

Teoreticky by tedy bylo možné z obsahu souboru rozpoznat přesně, o jaké kódování se u XML dokumentu jedná...

Document Type Definition (DTD)

Document Type Definition (DTD)

- Definice typu dokumentu (použití této definice je pak **deklarace typu dokumentu**)
- Specifikována přímo standardem XML 1.0
- Popisuje přípustný **obsah elementů**, **atributů**, jejich implicitní (default) hodnoty, definuje použité **entity**
- Může být uvedena jako **interní** nebo **externí** DTD (*internal and external subset*) nebo "napůl" - tam i tam.
- Dokument vyhovující DTD je označován jako *valid* (platný).

DTD - tutoriály

- Webreview: http://www.webreview.com/2000/08_11/developers/08_11_00_2.shtml
- ZVON: http://www.zvon.org/xxl/DTDTutorial/General/contents.html
- XML DTD Tutorial (101): http://www.xml101.com/dtd/
- W3Schools DTD Tutorial: http://www.w3schools.com [http://www.w3school.com]

DTD - deklarace typu dokumentu podrobněji

Uvádí se těsně před kořenový elementem konstrukcí

- `<!DOCTYPE jméno-kořenového-elt Externí-ID [ interní část DTD ]>`

Interní nebo **externí** část (*internal or external subset*) nemusí být uvedena nebo mohou být uvedeny obě.

Externí identifikátor může být buď

- `PUBLIC "PUBLIC ID" "URI"` (hodí se pro "veřejná", obecně uznané DTD) nebo
- `SYSTEM "URI"` - pro soukromá nebo jiná "ne zcela standardizovaná" DTD ("URI" nemusí být jen URL na síti, může být i jméno souboru, vyhodnocení se děje podle systému, na němž se vyhodnocuje)

Význam interní a externí části je rovnocenný (a nesmí si odporovat - např. dvě definice téhož elementu).

Obsahem DTD je seznam deklarací jednotlivých prvků - *elementů, seznamů atributů, entit, notací*

DTD - podmíněné sekce

Slouží k "zakomentářování" úseků DTD např. při experimentování.

- `<![IGNORE[ toto se bude ignorovat ]]>`
- `<![INCLUDE[ toto se zahrne do DTD (tj. nebude se ignorovat)]]>`

DTD - definice typu elementu

Popisuje možný obsah elementu, má formu `<!ELEMENT jméno-elementu ... >`,kde `...` může být

- `EMPTY` - prázdný element, může být zobrazen jako `<element/>` nebo `<element></element>` - totéž
- `ANY` - povolen je libovolný obsah elementu, tj. text, dceřinné elementy, ...
- může obsahovat **dceřinné elementy** - `<!ELEMENT jméno-elementu (specifikace dceřinných elementů)>`
- může být **smíšený** (MIXED) - obsahující text i dceřinné elementy dané výčtem `<!ELEMENT jméno-elementu (#PCDATA | přípustné dceřinných elementy)*>`. Nelze specifikovat pořadí nebo počet výskytů konkrétních dceřinných elementů. Hvězdička za závorkou je *povinná* - vždy je možný libovolný počet výskytů.

Pro specifikaci dceřinných elementů používáme:

- operátor **sekvence** (*sequence*, *follow with*) `,`
- operátor **volby** (výběru, *select*, *choice*) `|`
- závorky `()` mají obvyklý význam
- nelze kombinovat v jedné skupině různé operátory `, |`

- počet výskytů dceřinného elementu omezujeme specifikátory "hvězdička", "otazník", "plus" s obvyklými významy. Bez specifikátoru znamená, že je povolen právě jeden výskyt.

DTD - definice atributu

Popisuje (datový) typ, případně implicitní hodnoty atributu u daného elementu.

Má tvar `<!ATTLIST jméno-elementu jméno-atributu typ-hodnoty implicitní-hodnota>`

DTD - definice typu hodnoty atributu

Přípustné *typy hodnot* jsou:

- `CDATA`
- `NMTOKEN`
- `NMTOKENS`
- `ID`
- `IDREF`
- `IDREFS`
- `ENTITY`
- `ENTITIES`
- výčet hodnot - např. `(hodnota1|hodnota2|hodnota3)`
- výčet notací - např. `NOTATION (notace1|notace2|notace3)`

Atribut (i nepovinný) může mít implicitní hodnotu:

- `"implicitní hodnota"` - atribut je nepovinný, ale není-li uveden, chápe se to, jako by měl hodnotu `implicitní hodnota`

DTD - předpis kardinality (počtu výskytů) atributu

Atributy mohou mít předepsán (povinný) výskyt:

- `#REQUIRED` - atribut je povinný
- `#IMPLIED` - atribut je nepovinný

- `#FIXED "pevná-hodnota"` - atribut je povinný a musí mít právě hodnotu `pevná-hodnota`

DTD - implicitní hodnota atributu

Atribut (i nepovinný) může mít implicitní hodnotu:

- `"implicitní hodnota"` - atribut je nepovinný, ale není-li uveden, chápe se to, jako by měl hodnotu `implicitní hodnota`

Fyzická struktura (entity)

Entita - deklarace a použití

Rozlišuje se:

- deklarace
- reference (tj. použití) dané (již deklarované) entity.

Entity obecné (general) - mohou být

- *parsované* - soubory se (správně utvořeným) značkováním,
- *neparsované* - např. binární soubory,
- *znakové* - znaky, např. `>` je referencí na znakovou entitu

Entity parametrické (parametric)

- mohou být použity *jen v rámci DTD*
- hodí se při např. deklaracích *seznamu atributů* (pokud je dlouhý a vícekrát použitý, nahradíme ho referencí na parametrickou entitu)
- viz např. DTD pro HTML 4.01 - http://www.w3.org/TR/html4/sgml/dtd.html
- definicí parametrické entity je např. `<!ENTITY % heading "H1|H2|H3|H4|H5|H6">`

Jmenné prostory

Jmenné prostory (XML Namespaces)

- XML Namespaces (W3C Recommendation): http://www.w3.org/TR/REC-xml-names
- Existuje také nové *Namespaces in XML 1.1 W3C Recommendation [http://www.w3.org/TR/xml-names11/]* 4th February 2004. Andrew Layman, Richard Tobin, Tim Bray, Dave Hollander
- Definují "logické prostory" jmen (elementů, atributů) v XML dokumentu.
- Dávají uzlům ve stromu XML dokumentu "třetí dimenzi".
- Logickému prostoru jmen odpovídá jeden globálně ("celosvětově") jednoznačný identifikátor, daný URI (URI tvoří nadmnožinu URL).
- NS odpovídající danému URI nemá nic společného s obsahem nacházejícím se případně na tomto URL ("nic se odnikud automaticky nestahuje" - nedochází k tzv. dereferenci daného URI).

Prefixy jmenných prostorů, shoda...

- V rámci dokumentů se místo těchto URL používají zkratky, *prefixy* těchto NS namapované na příslušné URI atributem `xmlns:prefix="URI"`.

  Jméno elementu či atributu obsahující dvojtečku se označuje jako *kvalifikované jméno, QName*.
- Dva NS jsou stejné, jestliže se jejich URI shodují po znacích přesně (v kódování UNICODE).
- NS neovlivňují význam textových uzlů.
- Element/atribut nemusí patřit do žádného NS.
- Deklarace prefixu NS nebo implicitního NS má platnost na všechny dceřinné uzly rekurentně, dokud není uvedena jiná deklarace "přemapující" daný prefix.
- Jeden NS je tzv. *implicitní* (*default NS*), deklarovaný atributem `xmlns=`
- Na atributy se *implicitní NS nevztahuje*!!!, čili atributy bez explicitního uvedení prefixu nejsou *v žádném NS*.

Příklad implicitního jmenného prostoru

V následující ukázce je pro celý úryvek platný deklarovaný implicitní jmenný prostor charakterizovaný URI (URL) `http://www.w3.org/1999/xhtml`

Příklad 1.5. Implicitní jmenný prostor

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

```
   <body>
      <h1>Hurááááá</h1>
   </body>
</html>
```

Příklad explicitního jmenného prostoru

V následující ukázce je deklarován a přiřazen prefixu `xhtml` jmenný prostor charakterizovaný URI (URL) `http://www.w3.org/1999/xhtml`

Příklad 1.6. Jmenný prostor mapovaný na prefix

```
<xhtml:html xhtml:xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <xhtml:body>
      <xhtml:h1>Hurááááá</xhtml:h1>
   </xhtml:body>
</xhtml:html>
```

Obtíže se jmennými prostory

Dosud ne všechny parsery dokážou rozpoznávat NS. *...i když problémy jsou s tím dnes výjimečné...*

NS jsou nekompatibilní s DTD (DTD přísně rozlišuje např. jméno `xi:include` a `include`, přestože patří do stejného NS a mají tedy z hlediska aplikace obvykle stejnou interpretaci/význam).

XML Base

XML Base

- XML Base, W3C Recommendation 27 June 2001: http://www.w3.org/TR/xmlbase/
- Standard pro vyhodnocování relativních URL v odkazech z/na XML dokumenty.
- Definuje použití vyhrazeného atributu `xml:base` označujícího základ pro vyhodnocování relativních URL.
- Doplňuje se se standardem *XLink*.
- Respektuje princip "překrývání" bázové adresy nastavené v nadřazeném elementu.

XML Base - příklad

Příklad 1.7. xml:base určuje základ pro relativní URL

```
  <!--  Slides RelaxNG locations   -->
- <group xml:base="schema/relaxng/" id="slides-relaxng"
    prefer="public">
    <uri name="slides.rng" uri="slides.rng" />
    <uri name="slides-full.rng" uri="slides-full.rng" />
  </group>
```

Poznámka

Všimněte si použití vyhrazeného prefixu `xml`:

XML Inclusions

XML Inclusions (XInclude)

- XML Inclusions (XInclude) Version 1.0 W3C Working Draft 10 November 2003, http://www.w3.org/TR/xinclude/
- XInclude umožňuje vkládání (částí) XML dokumentů do dokumentů.
- Je ortogonální k entitám (lze použít oboje v rámci jednoho dokumentu, "nevadí si").
- Nezávislé na DTD (zpracování XInclude probíhá až po validaci)
- Nezávislé na XML Schema

XInclude: použití

- Specifikace definuje *jmenný prostor* a v něm jeden *element* `<xi:include>` s *atributy*:
- `href=` - vkládaný dokument
- `parse=` - hodnota je buď "text", pak se obsah vkládá jako (neparsovaný) text, nebo "xml", pak se hodnota vkládá jako značkovaný obsah
- `encoding=` - v případě `encoding="text"` specifikuje (je-li to nutné) kódování vkládaného textu
- a dalšími atributy (`xpointer`, `accept`, `accept-charset`, `accept-language`...) viz specifikace.

- Na FI je k dispozici interpret rozšířeného XInclude - xincluder-fi [http://www.fi.muni.cz/~tomp/xincluder-fi], který umí vkládat části textových souborů.

XInclude: příklad

Příklad 1.8. Vložení textového souboru (jako textového uzlu)

```
<xhtml:html xhtml:xmlns="http://www.w3.org/1999/xhtml"
            xml:lang="en" lang="en">
   <xhtml:body>
      <xhtml:h1>Huráááá</xhtml:h1>
      <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
                  href="obsah.txt" encoding="Windows-1250"/>
   </xhtml:body>
</xhtml:html>
```

XML Catalogs

XML Catalogs

- Vycházejí ze starších SGML katalogů
- Jde o prostředek, jak se jednotně odkazovat na entity (dokumenty) umístěné na různých systémech na různých místech.
- Dovoluje také praktické použití identifikátorů URI typu PUBLIC, které neodkazují na žádnou reálnou lokaci na internetu.
- Existuje několik formátů pro katalogy - bohužel.

XML Catalogs - příklad

Příklad 1.9. Katalog pro styly značkování DocBook Slides

```
<?xml version="1.0"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
<!-- Slides DTD locations -->
<group xml:base="schema/dtd/"
       id="slides-dtd"
       prefer="public">
```

```
  <public
    publicId="-//Norman Walsh//DTD Slides Custom V3.1.0//EN"
    uri="slides-custom.dtd"/>

  <public
    publicId="-//Norman Walsh//DTD slides Full V3.1.0//EN"
    uri="slides-full.dtd"/>
</group>

  <rewriteURI
    uriStartString="http://docbook.sourceforge.net/release/xsl/current/"
    rewritePrefix="file:/c:/devel/docbook-xsl-1.62.4/"/>

<!-- Map web references to DocBook 4.2 DTD -->
   <nextCatalog catalog="file:/c:/devel/docbook4.2/catalog.xml" />
</catalog>
```

XML Information Set

XML Information Set (XML Infoset) - cíle

- *XML Infoset 2nd Edition W3C Recommendation* First published.og 24 October 2001, revised 4 February 2004, John Cowan, Richard Tobin, http://www.w3.org/TR/xml-infoset/
- Infoset popisuje "jaké všechny informace lze o uzlu (elementu, dokumentu, atributu...) získat"
- Jinými slovy: aplikace by neměla spoléhat na informace z XML dokumentu, které se po analýze (parsingu) neobjeví v Infosetu.
- Každý správně utvořený XML dokument vyhovující standardu pro jmenné prostory má Infoset.

XML Infoset - struktura

- Infoset se skládá z *Information items*
- Infoset se týká dokumentu s již expandovanými entitami
- Rozlišuje se infoset *dokumentu*, *elementu*, *atributu*, *znaku*, *instrukci pro zpracování*, *neexpandované entitě*, *neanalyzované entitě*, *notaci*
- Podrobněji viz specifikace.

Kanonický tvar XML

Kanonický tvar XML dokumentu

- Canonical XML Version 1.0, W3C Recommendation 15 March 2001, http://www.w3.org/TR/xml-c14n
- Smyslem je popsat kritéria (a algoritmy), které pomohou rozhodnout, zda jsou dva XML dokumenty ekvivalentní, lišící se pouze fyzickou reprezentací (entity, pořadí atributů, kódování znaků)
- Kanonizace "setře" rozdíly mezi takovými dokumenty, k nimž se analyzátor "bude jistě chovat stejně", tj. z pohledu aplikace jsou totožné.
- Použití kanonického tvaru je nutné např. u *elektronického podpisu* XML dat (při výpočtu hodnoty *digest*).
- Bylo by možné nad XML dokumenty definovat i jiné relace ekvivalence než je *Canonical XML*.

Kanonický tvar - zásady konstrukce

Hlavní zásady konstrukce kanonického tvaru XML dokumentu:

- kódování v UTF-8
- zlomy řádků (CR, LF) jsou normalizovány podle algoritmu uvedeného v std. XML 1.0
- hodnoty atributů jsou normalizovány
- reference na znakové a parsované entity jsou nahrazeny jejich obsahem
- CDATA sekce jsou nahrazeny jejich obsahem
- hlavička "xml" a deklarace typu dokumentu jsou odstraněny
- bílé znaky mimo kořenový element jsou normalizovány
- jiné bílé znaky (vyjma normalizace zlomu řádků) jsou zachovány
- hodnoty atributů jsou uvozeny "
- speciální znaky v hodnotách atributů a textovém obsahu elementů jsou nahrazeny referencemi na entity
- nadbytečné deklarace jmenných prostorů jsou z každého elementu odstraněny
- implicitní hodnoty atributů jsou dodány do každého elementu (kde je to relevantní)
- na pořadí atributů a deklarací jmenných prostorů se uplatní lexikografické řazení

Potíže při definici kanonického tvaru

Ztráta řady informací (typicky pocházejících z DTD):

- neparsované entity (např. binární entity) jsou po kanonizaci nepřístupné
- notace
- typy atributů (vč. implic. hodnot)

Základní pojmy

Cílem rozhraní je

- poskytnout jednoduchý standardizovaný přístup ke XML datům
- "napojit" analyzátor (parser) na aplikaci a aplikace navzájem
- odstínit aplikaci od fyzické struktury dokumentu (entity)
- zefektivnit zpracování XML dat

Hlavní typy rozhraní pro zpracování XML dat:

- Stromově orientovaná rozhraní (Tree-based API)
- Rozhraní založená na událostech (Event-based API)
- Rozhraní založená na "vytahování" událostí/prvků z dokumentu (Pull API)

Stromově orientovaná rozhraní (Tree-based API)

Mapují XML dokument na stromovou strukturu v paměti

- dovolují libovolně procházet ("traverse") vzniklý strom;
- nejznámější je *Document Object Model* (DOM) konsorcia W3C, viz http://www.w3.org/DOM [http://www.w3.org/DOM/]

Modely specifické pro konkrétní prostředí

- pro Javu: JDOM - http://jdom.org
- pro Javu: dom4j [http://www.instantweb.com/foldoc/foldoc.cgi?dom4j] - http://dom4j.org
- pro Python: 4Suite - http://4suite.org

Rozhraní založená na událostech (Event-based API)

Při analýze ("parsing") dokumentu "vysílají" zpracovávající aplikaci *sled událostí*.

- technicky realizováno jako *volání metod* ("callback")
- aplikace poskytuje *handlery*, které volání zachytávají a zpracovávají
- událostmi řízená rozhraní jsou "nižší úrovně" než stromová, protože
- pro aplikaci zůstává "více práce"
- jsou však úspornější na paměť (většinou i čas), samotná analýza totiž nevytváří žádné „trvalé“ objekty

Událostmi je např.:

- začátek a konec dokumentu (start document, end document)
- začátek a konec elementu (start element, end element) - předá současně i atributy
- instrukce pro zpracování (processing instruction)
- komentář (comment)
- odkaz na entitu (entity reference)
- Nejznámějším takovým rozhraním je SAX http://www.saxproject.org

SAX - příklad analýzy dokumentu

```
<?xml version="1.0"?>
<doc>
   <para>Hello, world!</para>
</doc>
```

vyprodukuje při analýze (parsingu) sled událostí:

```
start document
start element: doc {seznam atributů: prázdný}
start element: para {seznam atributů: prázdný}
characters: Hello, world!
end element: para
end element: doc
end document
```

Kdy zvolit událostmi řízené rozhraní?

- O co snazší pro autora parseru, o to náročnější pro aplikačního programátora...
- Aplikace si musí (někdy složitě) pamatovat stav analýzy, nemá nikdy "celý dokument pohromadě".
- Na úlohy, které lze řešit "lokálně", bez kontextu celého dokumentu, je to vhodné rozhraní.
- Obvykle poskytuje nejrychlejší možné zpracování.
- Aplikační nepříjemnosti lze obejít použitím nadstaveb, např. Streaming Transformations for XML (STX) [http://stx.sourceforge.net]

Vlastnosti (features) nastavitelné pro analýzu - parsing

Chování parseru produkujícího SAX události je možné ovlivnit nastavením tzv. *features* a *properties*.

- *Vlastnosti* (*features*) nastavitelné pro analýzu (parsing) http://www.saxproject.org/?selected=get-set
- Blíže k jednotlivým properties a features v článku Use properties and features in SAX parsers [???] (IBM DeveloperWorks/XML).

SAX filtry

SAX rozhraní nabízí možnost napsat třídu jako tzv. SAX filtr (přesněji implementaci rozhraní org.xml.sax.XMLFilter).

Objekt takové třídy na jedné straně události přijímá, zpracuje je a posílá dále.

Další informace k filtrování událostí naleznete např. v článku Change the events output by a SAX stream [http://www.ibm.com/developerworks/xml/library/x-tipsaxfilter/] (IBM DeveloperWorks/XML).

Další odkazy k SAX

- "Přímo od zdroje" http://www.saxproject.org

- SAX Tutorial k JAXP - http://java.sun.com/webservices/docs/ea1/tutorial/doc/JAXPSAX.html

Rozhraní založená na technice "pull"

Rozhraní založená na technice "pull"

- Aplikace "nečeká na události", ale "vytahuje si" příslušná data ze vstupního parsovaného souboru.
- Využíváme tam, kde "víme, co ve zdroji očekávat" a "postupně si to bereme"
- ... vlastně opak API řízeného událostmi.
- Z hlediska aplikačního programátora velmi pohodlné, ale implementace bývají o něco pomalejší než klasická "push" událostmi řízená rozhraní.
- Pro Javu existuje *XML-PULL parser API* - viz Common API for XML Pull Parsing [http://www.xmlpull.org/] a také
- nově vyvíjené rozhraní Streaming API for XML (StAX) [http://www.jcp.org/en/jsr/detail?id=173] vznikající "shora i zdola" jako produkt JCP (Java Community Process).

Streaming API for XML (StAX)

Toto API se později může stát standardní součástí javového prostředí pro práci s XML, tzv. JAXP.

Nabízí dva přístupy k "pull" zpracování:

- přístup k "vytahovaným" událostem prostřednictvím iterátoru - pohodlnější
- nízkoúrovňový přístup přes tzv. kurzor - rychlejší

StAX - příklad s iterátorem

Příklad 1.10. StAX - přístup iterátorem

```
import java.io.*;
import java.util.Iterator;
import javax.xml.namespace.QName;
import javax.xml.stream.*;
import javax.xml.stream.events.*;
public class ParseByEvent {
   public static void main(String[] args)
```

```
                        throws FileNotFoundException, XMLStreamException {
      // Use the reference implementation for the XML input factory
      System.setProperty("javax.xml.stream.XMLInputFactory",
                                 "com.bea.xml.stream.MXParserFactory");
      // Create the XML input factory
      XMLInputFactory factory = XMLInputFactory.newInstance();
      // Create the XML event reader
      FileReader reader = new FileReader("somefile.xml");
      XMLEventReader r =
         factory.createXMLEventReader(reader);
      // Loop over XML input stream and process events
      while(r.hasNext()) {
         XMLEvent e = r.next();
         processEvent(e);
      }
   }
/**
 * Process a single XML event
 * @param e - the event to be processed
 */
  private static void processEvent(XMLEvent e) {
     if (e.isStartElement()) {
        QName qname = ((StartElement) e).getName();
        String namespaceURI = qname.getNamespaceURI();
        String localName = qname.getLocalPart();
        Iterator iter = ((StartElement) e).getAttributes();
        while (iter.hasNext()) {
           Attribute attr = (Attribute) iter.next();
           QName attributeName = attr.getName();
           String attributeValue = attr.getValue();
        }
     }
     if (e.isEndElement()) {
        QName qname = ((EndElement) e).getName();
     }
     if (e.isCharacters()) {
        String text = ((Characters) e).getData();
     }
     if (e.isStartDocument()) {
        String version = ((StartDocument) e).getVersion();
        String encoding = ((StartDocument) e).getCharacterEncodingScheme();
        boolean isStandAlone = ((StartDocument) e).isStandalone();
     }
  }
}
```

Poznámka

příklad převzat z Tip: Use XML streaming parsers [http://www.ibm.com/developerworks/xml/library/x-tipstx] (IBM DeveloperWorks, sekce XML).

StAX - příklad s kurzorem

Příklad 1.11. StAX - přístup kurzorem

```
import java.io.*;
import javax.xml.stream.*;
public class ParseByIterator {
   public static void main(String[] args)
                        throws FileNotFoundException, XMLStreamException {
      // Use reference implementation
      System.setProperty(
         "javax.xml.stream.XMLInputFactory",
         "com.bea.xml.stream.MXParserFactory");
      // Create an input factory
      XMLInputFactory xmlif = XMLInputFactory.newInstance();
      // Create an XML stream reader
      XMLStreamReader xmlr =
         xmlif.createXMLStreamReader(new FileReader("somefile.xml"));
      // Loop over XML input stream and process events
      while (xmlr.hasNext()) {
         processEvent(xmlr);
         xmlr.next();
      }
   }
   /**
    * Process a single event
    * @param xmlr - the XML stream reader
    */
   private static void processEvent(XMLStreamReader xmlr) {
      switch (xmlr.getEventType()) {
         case XMLStreamConstants.START_ELEMENT :
            processName(xmlr);
            processAttributes(xmlr);
            break;
         case XMLStreamConstants.END_ELEMENT :
            processName(xmlr);
            break;
         case XMLStreamConstants.SPACE :
         case XMLStreamConstants.CHARACTERS :
            int start = xmlr.getTextStart();
```

```
            int length = xmlr.getTextLength();
            String text =
               new String(xmlr.getTextCharacters(), start, length);
            break;
         case XMLStreamConstants.COMMENT :
         case XMLStreamConstants.PROCESSING_INSTRUCTION :
            if (xmlr.hasText()) {
               String piOrComment = xmlr.getText();
            }
            break;
      }
   }
   private static void processName(XMLStreamReader xmlr) {
      if (xmlr.hasName()) {
         String prefix = xmlr.getPrefix();
         String uri = xmlr.getNamespaceURI();
         String localName = xmlr.getLocalName();
      }
   }
   private static void processAttributes(XMLStreamReader xmlr) {
      for (int i = 0; i < xmlr.getAttributeCount(); i++)
         processAttribute(xmlr, i);
   }
   private static void processAttribute(XMLStreamReader xmlr, int index) {
      String prefix = xmlr.getAttributePrefix(index);
      String namespace = xmlr.getAttributeNamespace(index);
      String localName = xmlr.getAttributeName(index);
      String value = xmlr.getAttributeValue(index);
   }
}
```

Poznámka

příklad převzat z Tip: Use XML streaming parsers [http://www.ibm.com/developerworks/xml/library/x-tipstx] (IBM DeveloperWorks, sekce XML).

Document Object Model (DOM)

Základní rozhraní pro tvorbu a přístup ke stromové re-prezentaci XML dat.

- existují verze *DOM Level 1, 2, 3*

- DOM je obecně *nezávislý* na způsobu analýzy (parsingu) vstupního XML
- Je popsán IDL definicemi+popisy rozhraní v jednotlivých jazycích (zejm. C++ a Java)

Specifický DOM pro HTML dokumenty

- Core (základ) DOM pro HTML je nyní "víceméně" sloučen s DOM pro XML
- určen pro styly CSS
- určen pro programování dynamického HTML (skriptování - VB Script, JavaScript)
- kromě samotného dokumentu model zahrnuje i prostředí prohlížeče (např. `window` [http://www.instantweb.com/foldoc/foldoc.cgi?window], `history` [http://www.instantweb.com/foldoc/foldoc.cgi?history]...)

Odkazy k DOM

- Tutoriál k JAXP, část věnovaná DOMPart III: XML and the Document Object Model (DOM) [http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/index.html]
- Portál věnovaný DOM http://www.oasis-open.org/cover/dom.html
- Vizuální přehled DOM 1 rozhraní http://www.xml.com/pub/a/1999/07/dom/index.html
- Tutoriál "Understanding DOM (Level 2)" na http://ibm.com/developer/xmlhttp://ibm.com/developer/xml [http://ibm.com/developer/xml]

Implementace DOM

- v mnoha parserech, např. Xerces [http://xml.apache.org]
- jako součást JAXP (Java API for XML Processing) - http://java.sun.com/xml/jaxp/index.html
- i jako samostatné, nezávislé na parserech:
  - např. `dom4j` [http://www.instantweb.com/foldoc/foldoc.cgi?dom4j] - http://dom4j.org
  - `EXML` [http://www.instantweb.com/foldoc/foldoc.cgi?EXML] (Electric XML) - http://www.themindelectric.net

Alternativní stromové modely - XOM

- XOM (*XML Object Model*) vznikl jako one-man-show projekt (autor Elliote Rusty Harold) rozhraní, které je "papežštější než papež" a striktně respektuje model XML dat.
- Motivaci a specifikaci najdete na domovské stránce XOM [http://cafeconleche.org/XOM/].
- Tam je též k získání open-source implementace XOM [http://cafeconleche.org/XOM/xom-1.0d24.zip] a
- dokumentace API [http://cafeconleche.org/XOM/apidocs/].

Alternativní parsery a stromové modely - NanoXML

- velmi malé (co do velikosti kódu) stromové rozhraní a parser v jednom
- dostupné jako open-source na http://nanoxml.n3.net
- adaptované též pro mobilní zařízení
- z hlediska rychlosti a paměťové efektivity za běhu ale nejlepší *není*

Prakticky dobře použitelný stromový model: dom4j

- pohodlné, rychlé a pamětově efektivní stromově-orientované rozhraní
- psané pro Javu, optimalizované pro Javu...
- dostupné jako open-source na http://dom4j.org
- nabízí perfektní přehled díky "kuchařce" [http://dom4j.org/cookbook/cookbook.html]
- dom4j je výkonný, viz srovnání efektivity jednotlivých stromových modelů [http://www.ibm.com/developerworks/xml/library/x-injava/]

Kombinace stromových a událostmi řízených přístupů

Události -> strom

- Je např. možné "nezajímavou" část dokumentu *přeskočit* nebo odfiltrovat pomocí sledování událostí a pak
- za "zajímavé" části vytvořit strom v paměti a ten zpracovávat.

Strom -> události

- Vytvoříme strom dokumentu (a zpracujeme ho) a
- strom následně procházíme a generujeme události jako bychom četli výchozí soubor.
- Toto umožňuje snadnou integraci obou typů zpracování v jedné aplikaci

Virtuální objektové modely

- DOM model dokumentu není přítomen v paměti, je zprostředkováván "on demand" při přístupu k jednotlivým uzlům
- spojuje výhody událostmi řízeného a stromového modelu zpracování (rychlost + komfort)
- implementován např. u procesoru `Sablotron` [http://www.instantweb.com/foldoc/foldoc.cgi?Sablotron] (např. viz http://www.xml.com/pub/a/2002/03/13/sablotron.html nebo http://www.gingerall.org/charlie/ga/xml/p_sab.xml)

Cíle a charakteristiky jazyků schémat

Cíle modelování XML dat

Cílem je poskytnout abstraktní model dat v příslušném značkování tak, abychom:

- mohli validovat, zda dokumenty jsou syntakticky korektní, zda odpovídají schématu
- následně mohli dokumenty (data) zpracovávat jako silně typované:
  - hodnoty textových uzlů pak interpretovány jako hodnoty primitivních datových typů (int, float, boolean...), nebo vestavěných neprimitivních typů - řetězec, datum
  - elementy interpretovány jako hodnoty uživatelských objektových typů, např. element `person` jako objekt třídy `Person`.
- používat informace o struktuře validních dokumentů např. při vyhledávání:
  - víme-li např., že element `from` se vyskytuje jen jako dceřinný v elementu `message`, můžeme a priori rozhodnout, že výsledek XPath dotazu `/address/from` bude nad validními dokumenty vždy prázdný
  - totéž např. pro dokumenty, kde známe pořadí výskytu dceřinných elementů za sebou - pak můžeme např. predikovat obsah XPath osy `following-sibling`.

- používat je při vizualizaci, formátování, editaci(!)

Přístupy k modelování XML dat

XML 1.0, 1.0 SE, 2.0 standardy definují metajazyky, tj.:

- neříkají ni o konkrétním značkování
- ale sděluje, jaká obecná omezení pro konstrukci značkování platí (např. správné vnoření elementů)
- definuje DTD jako základní formální jazyk definice struktury dokumentu
- ale nevylučuje existenci dalších takových jazyků

Popis příslušného značkování, tj. návod, které dokumenty v daném značkování chápeme jako platné (validní), nazýváme *schématem* příslušného značkovacího jazyka.

(Meta)jazyk, v němž je schéma zapsáno, nazveme *jazykem schémat*.

Neplést schéma a jazyk schématu v tomto širokém smyslu s *XML Schema* (což je speciální případ jazyka schémat, případně jeho instance).

Kategorie jazyků schemat

Podle základního výrazového prostředku, kterým definujeme „správnou“ strukturu dokumentů, rozlišujeme jazyky schémat založené na:

- *gramatikách*
- objektovém přístupu, skládání, dědičnosti
- *vzorech*

Modelování pomocí gramatik

Typická situace:

- značkování je bezkontextovým jazykem definovaným gramatikou
- typickým představitelem je gramatika zapsaná v *DTD* (s jistým non-CF omezeními, jako je např. vazba ID-IDREF)
- jazyky XML Schema i Relax NG převzaly částečně tento přístup - kombinují gramatiku s možností dědičnosti

Typickým představitelem je XML DTD [http://w3.org/XML].

Modelování pomocí dědičnosti

Hlavní charakteristiky:

- jednou vytvořené schéma či jeho část je možné pojmenovat a znovu použít
  - beze změny
  - při znovupoužití je možné pozměnit jeho vlastnosti (např. zpřísnit některá omezení), obvykle ale tak, že je zachováno pravidlo, že instance podtřídy může vždy nahradit instanci nadtřídy (má aspoň její vlastnosti)

Typickými představiteli jsou XML Schema [http://www.w3.org/XML/Schema] a Relax NG [http://relaxng.org] (čteme jako „relaxing“).

Modelování pomocí vzorů

Schéma tohoto typu říká, jaké vzory se mohou/musejí/nesmějí v dokumentu nacházet.

- nejméně používaná schémata, ale v mnohých situacích *nenahraditelná*
- vhodná pro XML data dokumentové (tj. velmi heterogenní, mnohotvaré) povahy - značkované texty
- Používáme tam, kde je prostředky (CF) gramatiky určité omezení (constraint) těžko popsatelný - např. složité *kontextové závislosti*. To by se jinak muselo např. „dovalidovat“ externím nástrojem.
- prostředkem popisu vzorů je určitý „vnořený“ jazyk - např. *stromové regulární výrazy*, výrazy jazyka *XPath*...

Typickým představitelem je Schematron [http://www.ascc.net/xml/resource/schematron/schematron.htm] a Examplotron [http://examplotron.org].

Funkcionální modelování

Funkcionální modelování

XML Schema

XML Schema - základní zdroje informací

Specifikace XML Schema - http://www.w3.org/XML/Schema

Tutoriál *Using W3C XML Schema*: http://www.xml.com/pub/a/2000/11/29/schemas/part1.html - stručný

XML Schema Tutorial - http://www.w3schools.com/schema/default.asp - obsáhlejší

vynikající komplexní tutoriál na http://www.xfront.com

XML Schema - motivace

Dát silnější prostředek pro specifikaci modelu XML dat než je DTD; mít možnost:

- Oddělit koncept *typu* (např. typu elementu) od jeho *výskytu* (instance, např. elementu s určitým názvem) - to DTD neumí
- Poskytnout bohatší škálu *primitivních datových typů*
- Umožnit použití *jmenných prostorů*
- Umožnit jemnější specifikaci *modelu obsahu* (elementů)
- Umožnit *odvozování* nových typů (*dědičností*)
- Umožnit *modularizaci* a znovupoužitelnost schémat
- Zapisovat schéma v XML

XML Schema - hlavička definice schématu

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   .../...
</xs:schema>
```

XML Schema - přiřazení typu elementu s daným názvem

```
<xs:element name="element_name">
   ... definice typu - je přímo zde - tzv. "local" nebo daná odkazem - tzv. "globa
</xs:element>
```

XML Schema - definice jednoduchého typu

- Neobsahuje dceřinné elementy, lze použít jako typ obsahu elementu nebo atributu

- Lze definovat restrikcí z existujícího typu

```
<xs:simpleType name="TypeName">
   <xs:restriction base="BaseTypeName"> ... </xs:restriction>
</xs:simpleType>
```

XML Schema - definice jednoduchého typu - příklad 1

Restrikce délky obsahu

```
<xs:simpleType name="nameType">
   <xs:restriction base="xs:string">
      <xs:maxLength value="32"/>
   </xs:restriction>
</xs:simpleType>
```

XML Schema - definice jednoduchého typu - příklad 2

Restrikce obsahu regulárním výrazem

```
<xs:simpleType name="isbnType">
   <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{10}"/>
   </xs:restriction>
</xs:simpleType>
```

XML Schema - jednoduché typy - "union"

Zhruba odpovídá konceptu "union" v C

Výsledkem je jednoduchý typ

Lze spojovat bázový typ a výčet hodnot

Příklad:

```
<xs:simpleType name="isbnType">
   <xs:union>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{10}"/>
         </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
         <xs:restriction base="xs:NMTOKEN">
```

```
            <xs:enumeration value="TBD"/>
            <xs:enumeration value="NA"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:union>
</xs:simpleType>
```

XML Schema - jednoduché typy - seznam hodnot

Lze definovat typ jako seznam hodnot oddělených bílými znaky

Dalším odvozením lze omezit počet prvků seznamu

Příklad

```
<xs:simpleType name="isbnTypes">
   <xs:list itemType="isbnType"/>
</xs:simpleType>
<xs:simpleType name="isbnTypes10">
   <xs:restriction base="isbnTypes">
      <xs:minLength value="1"/>
      <xs:maxLength value="10"/>
   </xs:restriction>
</xs:simpleType>
```

XML Schema - definice složeného typu

```
<xs:complexType name="TypeName">
   <xs:sequence>
      <xs:element ...>
      ...
      <xs:attribute ...>
   </xs:sequence>
</xs:complexType>
```

Místo sekvence lze použít `<xs:choice>` a `<xs:all>`

XML Schema - definice složeného typu - skupiny

při definici složeného typu lze použít skupiny (group)

Skupina elementů:

```
<xs:group name="GroupName">
   <xs:sequence>
      <xs:element ... />
```

```
      ...
   </xs:sequence>
</xs:group>
```

Místo sekvence lze použít <xs:choice> a <xs:all>

XML Schema - definice složeného typu - skupiny atributů

Skupina atributů:

```
<xs:attributeGroup name="AttributesGroupName">
   <xs:attribute ... use="required"/>
   ...
</xs:attributeGroup>
```

Může být uvedena povinnost výskytu (use=required)

XML Schema - použití skupin

Příklad použití skupin elementů a atributů

```
<xs:complexType name="bookType">
   <xs:sequence>
      <xs:group ref="mainBookElements"/>
      <xs:element name="character" type="characterType"
                  minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attributeGroup ref="bookAttributes"/>
</xs:complexType>
```

XML Schema - kompozitor "sequence"

Předepisuje výskyt dceřinných elementů v určitém pořadí

```
<xs:element name="element_name">
   <xs:complexType>
      <xs:sequence>
         .../...
      </xs:sequence>
      .../...
   </xs:complexType>
</xs:element>
```

sequence označuje model obsahu připouštějící výskyt dané posloupnosti (sekvence) dceřinných elemen-

tů

xs je prefix vázaný na NS s URL http://www.w3.org/2001/XMLSchema

Místo <xs:sequence> lze použít <xs:choice> nebo <xs:all>

XML Schema - kompozitor "choice"

Předepisuje výskyt jednoho z dceřinných elementů nebo skupin elementů

```
<xs:element name="element_name">
   <xs:complexType>
      <xs:choice>
        .../...
      </xs:choice>
      .../...
   </xs:complexType>
</xs:element>
```

XML Schema - kompozitor "all"

Předepisuje výskyt dceřinných elementů bez určeného pořadí

Smí být jen na nejvyšší úrovni definice obsahu

Dceřinné elementy nesmí mít kardinalitu větší než 1

Příklad:

```
<xs:complexType name="bookType">
   <xs:all>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
      <xs:element name="character" type="characterType"
                  minOccurs="0" maxOccurs="unbounded"/>
   </xs:all>
   <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
```

XML Schema - jednoduchý obsah elementu

Příklad:

```
<xs:element name="book">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="isbn" type="isbnType"/>
```

```
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

XML Schema - smíšený obsah elementu

Nelze validovat textový obsah (textové dceřinné uzly)

Lze validovat dceřinné elementy

Příklad:

```
<xs:element name="book">
   <xs:complexType mixed="true">
      <xs:all>
         <xs:element name="title" type="xs:string"/>
         <xs:element name="author" type="xs:string"/>
      </xs:all>
      <xs:attribute name="isbn" type="xs:string"/>
   </xs:complexType>
</xs:element>
```

XML Schema - další možnosti

Možnost specifikace integritních omezení:hodnota je jedinečná - `xs:unique`hodnota je klíčem - `xs:key`hodnota je odkazem na klíč - `xs:keyref`

XML Schema - anotace schémat

Anotace je (lidsky čitelná) poznámka-komentář ke schématu

Může též obsahovat informace pro zpracování - viz příklad - xs:appinfo

Další obsah není předepsán (omezen) - viz příklad - bind, class

Příklad

```
<xs:annotation>
   <xs:documentation xml:lang="en">Top level element.</xs:documentation>
   <xs:documentation xml:lang="fr">Element racine.</xs:documentation>
   <xs:appinfo source="http://example.com/foo/">
      <bind xmlns="http://example.com/bar/">
         <class name="Book"/>
      </bind>
   </xs:appinfo>
</xs:annotation>
```

XML Schema - znovupoužití definice schématu

Přímo:

```
<xs:include schemaLocation="character.xsd"/>
```

S předefinováním:

```
<xs:redefine schemaLocation="character12.xsd">
   <xs:simpleType name="nameType">
      <xs:restriction base="xs:string">
         <xs:maxLength value="40"/>
      </xs:restriction>
   </xs:simpleType>
</xs:redefine>
```

XML Schema - abstraktní a konečné typy

abstract - nelze instanciovat, pouze jako zákad k odvozování dědičností

final - nelze rozšiřovat/odvozovat dědičností

XML Schema - jmenné prostory

Příklad

```
<xs:schema targetNamespace="http://example.org/ns/books/"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:bk="http://example.org/ns/books/"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   .../...
</xs:schema>
```

XML Schema - nespecifikované elementy a atributy

Umožní připustit i něco, co předem neznáme

Příklad

```
<xs:complexType name="descType" mixed="true">
   <xs:sequence>
      <xs:any namespace="http://www.w3.org/1999/xhtml"
              processContents="skip"
              minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
```

```
</xs:complexType>
```

Pro atributy - xs:anyAttribute

XML Schema - odkaz na definici schématu

```
<book isbn="0836217462"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="file:library.xsd">
```

```
<book isbn="0836217462"
      xmlns="http://example.org/ns/books/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="file:library.xsd">
```

Relax NG

Relax NG - motivace

XML Schema:

- Je (zbytečně) složité (specifikace má přes 200 stran)
- Může vést v jistých situacích k nejednoznačnostem.
- Snaží se o pokrytí všech aplikačních oblastí (dokumentové i databázové použití XML a všechno mezi tím).
- Obtížně (úplně) implementovatelné.
- Dále viz http://www.xml.com/lpt/a/2002/01/23/relaxng.html

Relax NG - základní zdroje informací

Vznikl z RELAXu při skupině OASIS-OPEN:

- http://www.oasis-open.org/committees/relax-ng

Jazyky schémat používající vzory

Schematron

Schematron home page [http://www.ascc.net/xml/resource/schematron/schematron.html]

Examplotron

Examplotron home page [http://examplotron.org]

Ostatní jazyky schémat

DSD 2.0

Vznikl na Univerzitě v Aarhusu, DK

Podobně jako RELAX NG je jednodušší než XML Schema

viz http://www.brics.dk/~amoeller/XML/

Spíše akademický charakter, skutečnými soupeři zůstávají XML Schema a RELAX NG

Vyjadřovací síla těchto modelů, jejich nedostatky

viz http://www.xml.com/lpt/a/2001/12/12/schemacompare.html

Nástroje na validaci XML dat modelovaných podle těchto standardů

Nástroje na validaci XML dat modelovaných podle těchto standardů

XPath - aplikační oblasti

- Pokročilá navigace v XML datech
- Transformace (XSLT)
- V "selekční části" XML dotazovacích jazyků
- V některých modelovacích jazycích - Schematron

XPath - hlavní principy

- XPath je syntaxe pro specifikaci *částí* XML dokumentů (uzly, množiny uzlů, sekvence uzlů; nelze specifikovat *části* textových uzlů).
- XPath používá syntaxi obdobnou jako *cesty v souborovém systému*.
- XPath používá knihovnu standardních funkcí (evt. uživatelsky definovaných - v XPath 2.0 nebo i

XPath 1.x, ale proprietárně - podle procesorů)

- XPath je od v 1.0 základem pro XSLT, od 2.0 i pro XQuery
- XPath syntaxe *není XML* (bylo by příliš "upovídané")
- XPath 1.0 je W3C Recommendation - http://www.w3.org/TR/xpath

XPath - pojem cesty (paths) a lokace (locations)

Cesta určuje lokaci v dokumentu

Cesty jsou konstruovány podobně jako cesty ve FS, tj.

- relativní - vyhodnocovány vůči kontextovému uzlu (KU), viz dále
- absolutní - od kořene, ale výrazy (predikáty) také vyhodnocovány vůči KU

Syntaktická pravidla:

```
[20] PathExpr   ::=  AbsolutePathExpr |  RelativePathExpr
[22] AbsolutePathExpr   ::=  ("/"  RelativePathExpr?)  |  ("//"  RelativePathExpr)
[23] RelativePathExpr   ::=  StepExpr (("/"  |  "//")  StepExpr)*
[24] StepExpr   ::=  AxisStep |  GeneralStep
[25] AxisStep   ::=  (Axis?  NodeTest StepQualifiers)  |  AbbreviatedStep
```

XPath - osy (axes)

Osy jsou množiny prvků dokumentu, vymezené (obvykle relativně) vůči *kontextu*.

Kontext je tvořen především *dokumentem* a *aktuálním (kontextovým) uzlem*.

Osami jsou:

- child - obsahuje dceřinné uzly kontextového (aktuálního) uzlu
- descendant - obsahuje všechny potomky kontextového (aktuálního) uzlu (dále jen KU). Nepočítají se mezi ně atributy!!!
- parent - obsahuje rodičovský uzel KU (existuje-li)
- ancestor - obsahuje všechny předky - rodiče, "prarodiče"... kořenový element (pokud KU není sám kořenový)
- following-sibling - obsahuje všechny následující sourozence KU (pro NS a atributy je tato osa prázdná)

- preceding-sibling - dtto, ale obsahuje *předchozí* sourozence
- following - obsahuje všechny uzly nacházející se *po* KU (mimo atributů, potomků a NS uzlů
- preceding - dtto, ale obsahuje předchozí uzly (ale mimo předky, attributy, NS!)
- attribute - obsahuje atributy (jen pro uzly - elementy)
- namespace - obsahuje všechny NS uzly KU (jen pro uzly - elementy)
- self - obsahuje samotný KU
- descendant-or-self - obsahuje sjednocení os descendant a self
- ancestor-or-self - obsahuje sjednocení os ancestor a self

XPath - predikáty (predicates)

Určeny k selekci (výběru) z uzlů specifikovaných např. cestou

př.: /article/para[**3**] - vybere třetí odstavec v článku

Nejjednodušším výrazem v predikátu je specifikace *pozice (blízkosti)* (proximity position) - viz výše

- pozor u reverzních os (ancestor, preceding...) - pozice se počítá v rámci množiny uzlů vždy OD KONTEXTOVÉHO UZLU!
- specifikaci pozice **3** možno nahradit výrazem position()=3

XPath - Výrazy

Určeny k použití v predikátech, k výpočtům, atd.

Výrazy mohou být:

- řetězcové
- numerické (hodnotami jsou floating-point čísla)
- logické (boolean)
- uzly
- sekvence

XPath - zkrácená notace - Příklady

- para selects the para element children of the context node

  vybere všechny dceřinné elementy kontextového uzlu jmenující se para

- * selects all element children of the context node
- text() selects all text node children of the context node
- @name selects the name attribute of the context node
- @* selects all the attributes of the context node
- para[1] selects the first para child of the context node
- para[last()] selects the last para child of the context node
- */para selects all para grandchildren of the context node
- /doc/chapter[5]/section[2] selects the second section of the fifth chapter of the doc
- chapter//para selects the para element descendants of the chapter element children of the context node

  vybere všechny element para, jež jsou následníky chapter

- //para selects all the para descendants of the document root and thus selects all para elements in the same document as the context node

  vybere všechny elementy para z dokumentu

- //olist/item selects all the item elements in the same document as the context node that have an

  vybere všechny elementy item, které mají za rodiče olist

- . selects the context node

  vybere kontextový uzel

- .//para selects the para element descendants of the context node

  vybere všechny elementy-potomky kontextového uzlu, které nesou značku para

- .. selects the parent of the context node

  vybere rodičovský uzel od kontextového

- ../@lang selects the lang attribute of the parent of the context node

  vybere atribut lang rodičovského uzlu od kontextového

XPath - zkrácená notace (2)

Nejpoužívanější zkracování je *osy child* :

- tj. píšeme article/para místo child::article/child::para.
- a *atributu*: píšeme para[@type="warning"] místo child::para[attribute::type="warning"]
- Další používané zkracování je // místo /descendant-or-self::node()/
- a samozřejmě zkratky . a ..
- Pro přehlednost někdy delší formu zachováváme

Informační zdroje k XPath

- XPath na W3C: http://www.w3.org/TR/xpath
- Zvon XPath Tutorial: http://zvon.org/xxl/XPathTutorial/Output/index.html
- XPath Tutorial na W3Schools: http://www.w3schools.com/xpath/xpath_intro.asp

XPath 2.0

- zatím jako PRACOVNÍ NÁVRH - http://www.w3.org/TR/xpath20/
- Změna pohledu na hodnoty vrácené XPath výrazem: vše jsou **sekvence** (byť jednoprvkové)
- ->odstraňuje problémy s "pořadím" uzlů v množině
- Zavádí **podmíněné výrazy** a **cykly**
- Zavádí možnost uživatelských funkcí (psaných jako dynamicky vyhodnocované výrazy v XPath)
- Lze použít existenční a obecné kvantifikátory, např. exist student/name="Fred" nebo all student/@id
- Dále viz např. http://saxon.sourceforge.net/saxon7.8/index.html

XPath 2.0 - příklady

- Řetězcové funkce [/~tomp/xml03/xpath20/string.html]
- Numerické funkce [/~tomp/xml03/xpath20/numeric.html]
- Funkce nad sekvencemi [/~tomp/xml03/xpath20/sequence.html]
- Booleovské funkce [/~tomp/xml03/xpath20/boolean.html]

XML Linking Language (XLink)

XLink - úvod

XLink pracuje na úrovni XML Infoset

Dovoluje odkazovat se:

- v rámci dokumentu,
- z jednoho dokumentu na další,
- mezi dokumenty (odkaz je uložen mimo ně)

Rozlišuje kategorie:

- **adresa** (též *zdroj* nebo *lokace*, tj. odkazovaný objekt - element, skupina elementů, text...)
- a vlastní **odkaz**
- Odkaz pak představuje vazbu mezi zdroji s případným upřesněním sémantiky odkazu.

XML Linking Language (XLink) - původ standardu

- W3C Recommendation 27 July 2001
- Obecný mechanizmus na propojování XML zdrojů (=dokumentů a jejich částí)
- Inspirován zejména std. **HyTime** (http://www.hytime.org/)
- Ortogonální k entitám (lze použít současně oboje)
- mnohem bohatší sémantika než u entit,
- entity se vyhodnocují (resolve) při parsingu,
- XLink odkazy většinou až v aplikaci

XLink - historie a motivace

- Vychází z pokročilých hypertextových technik - HyTime [http://www.instantweb.com/foldoc/foldoc.cgi?HyTime] (pro SGML - viz http://info.admin.kth.se/SGML/Anvardarforening/Arbetsgrupper/HyTime/Reports/tr1v1.html), TEI [http://www.instantweb.com/foldoc/foldoc.cgi?TEI]...

- Konstruován tak, že rozšiřuje a upřesňuje syntaxi i sémantiku HTML odkazů

Výhody odkazové infrastruktury na bázi XLink

- Na rozdíl od SGML linků (např. v HyTime) můžeme v XML využívat *jmenné prostory*. Nenutí nás to upravovat DTD při použití odkazů.
- Infrastruktura odkazů může existovat (v jiném NS) nezávisle na schématu (struktuře) XML dokumentů, v nichž se odkazy vyskytují.
- Odkazy mohou být fyzicky mimo soubory s odkazovanými lokacemi.

Specifikace, tutoriály

- XML Linking Language (XLink) Version 1.0 - Specification [http://www.w3.org/TR/xlink/]
- Zvon simple XLink tutorial [http://www.zvon.org/xxl/xlink/OutputExamples/xlinksimple_intro.html]
- Zvon extended XLink tutorial [http://www.zvon.org/xxl/xlink/xlink_extend/OutputExamples/xlinkextend_intro.html]

XLink - základní principy

Integrace XLinku do (schémat) dokumentů

- Atributy a elementy `XLink` [http://www.instantweb.com/foldoc/foldoc.cgi?XLink]u mají vlastní jmenný prostor
- Jako prefix NS se obvykle používá `xlink` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink]
- Příslušnost k XLinku je dána deklarací daného elementu/atributu v XLink NS
- to umožňuje flexibilní *integraci stávajících schémat (modelů) a XLinku* (nemusíme měnit jména "linkovacích" elementů)
- XLink odkaz "dovnitř" dokumentů používají standard `XPointer` [http://www.instantweb.com/foldoc/foldoc.cgi?XPointer] (http://www.w3.org/TR/xptr).

XLink - základní typy

- Jednoduchý (`simple` [http://www.instantweb.com/foldoc/foldoc.cgi?simple]) - vždy *"in-line"*
- Rozšířený (`extended` [http://www.instantweb.com/foldoc/foldoc.cgi?extended]) - může být *"out-of-line"* (i v jiném dokumentu/souboru/databázi odkazů)

Simple XLink odkaz

- Váže jeden lokální zdroj na jeden vzdálený (*single local -> single remote entity*);
- Je vyznačen atributem `xlink:type="simple"` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:type="simple"];

Extended XLink odkaz

- Váže **jeden nebo více lokálních** a **jeden nebo více vzdálených** zdrojů
- Může mít přesněji definovanou sémantiku (nad rámec výše uvedených možností)
- Odkazy mohou být uloženy mimo odkazující zdroj(e)
- je vyznačen atributem `xlink:type="extended"` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:type="extended"]

`extended` [http://www.instantweb.com/foldoc/foldoc.cgi?extended] odkaz se může odkazovat na

- lokální zdroje - vyznačené atributem `xlink:resource` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:resource] nebo
- vzdálené zdroje - vyznačené atributem `xlink:locator` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:locator]

Směr odkazu a role participujících zdrojů mohou být upřesněny (nad rámec výše uvedených možností):

- uvedením podelementů `arc` [http://www.instantweb.com/foldoc/foldoc.cgi?arc] v `extended` [http://www.instantweb.com/foldoc/foldoc.cgi?extended] odkazu

Příklad odkazu XLink (1) - jednoduchý XLink

```
<zvon:logo xmlns:zvon = "http://www.zvon.org"
           xmlns:xlink = "http://www.w3.org/1999/xlink"
```

```
xlink:type="simple" xlink:href="zvon.gif" />
```

Příklad odkazu XLink (2) - rozšířený XLink

```
<?xml version="1.0"?>
<zvon_tutorial xmlns:xlink="http://www.w3.org/1999/xlink">
   <extendedlink xlink:type="extended"> Any content here </extendedlink>
</zvon_tutorial>
```

Sémantika odkazů XLink

Chování aplikací nad odkazy XLink

Co má aplikace udělat s odkazovaným dokumentem/prvkem dokumentu je specifikováno atributem `xlink:show` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:show]

- `new` [http://www.instantweb.com/foldoc/foldoc.cgi?new] - chápat jako nový dokument (např. otevřít v novém okně)
- `replace` [http://www.instantweb.com/foldoc/foldoc.cgi?replace] - nahradit jím zdrojový dokument
- `embed` [http://www.instantweb.com/foldoc/foldoc.cgi?embed] - vložit odkazovaný obsah do zdrojového dokumentu

"kdy to má udělat" - specifikováno atributem `xlink:actuate` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:actuate]

- `onLoad` [http://www.instantweb.com/foldoc/foldoc.cgi?onLoad] - tj. inhed při zpracování zdrojového dokumentu (ale kdy? při parsingu?)
- `onRequest` [http://www.instantweb.com/foldoc/foldoc.cgi?onRequest] - na požádání (např. po kliknutí na zobrazený obsah odkazujícího elementu)

Upřesnění role odkazu pro aplikace

roli odkazu lze upřesnit v podobě *srozumitelné aplikacím*:

- atributem `xlink:role` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:role] - u `simple` [http://www.instantweb.com/foldoc/foldoc.cgi?simple] i `extended`

[http://www.instantweb.com/foldoc/foldoc.cgi?extended] odkazu

- atributem `xlink:arcrole` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:arcrole] - u elementu `arc` [http://www.instantweb.com/foldoc/foldoc.cgi?arc] v `extended` [http://www.instantweb.com/foldoc/foldoc.cgi?extended] odkazu

Upřesnění role odkazu pro člověka

Lze upřesnit **roli odkazu** v podobě *srozumitelné člověku*:

- atributem `xlink:title` [http://www.instantweb.com/foldoc/foldoc.cgi?xlink:title] - u `simple` [http://www.instantweb.com/foldoc/foldoc.cgi?simple] i `extended` [http://www.instantweb.com/foldoc/foldoc.cgi?extended] odkazu i u elementu `arc` [http://www.instantweb.com/foldoc/foldoc.cgi?arc] v `extended` [http://www.instantweb.com/foldoc/foldoc.cgi?extended] odkazu

Použití XLink a implementace procesorů XLink

Procesory XLink

Zatím nepříliš rozšířené (řádově jednotky aplikací). Nejznámějšími procesory jsou:

- Fujitsu XLink Processor (XLiP) - http://www.labs.fujitsu.com/free/xlip/en/
- Open source projekt `xlinkit` [http://www.instantweb.com/foldoc/foldoc.cgi?xlinkit] - http://www.xlinkit.com/xtoox/index.html
- X2X (komerční, velmi drahý produkt) podporuje XLink - http://www.stepuk.com/products/prod_X2X.asp

Prohlížeče

podpora *simple* odkazů

- Mozilla (open source) a Netscape (komerční)
- DocZilla (komerční)
- MSIE ani Opera (vč. posledních verzí) XLink *nepodporují*

Příčiny nízkého rozšíření

- Vývoj trval dlouho (cca 5 let), mezitím aplikace začaly používat vlastní řešení
- Sémantika XLinku je:
  - **příliš obecná** (aplikace raději používaly vlastní, na míru šitý odkazovací jazyk, viz (X)HTML) a současně
  - **málo obecná** (např. RDF metadata popíší vztahy mezi zdroji podrobněji, s typovou kontrolou přes RDF Schemata, atd.)

XLink je také nahrazován aktivitou směřující ke specifikaci *XML Topic Maps*

Alternativy k XLink

HLink

- Cílem je adaptovat XLink pro potřeby XHTML, kde "nelze použít přímo XLink"
- viz HLink - Link recognition for the XHTML Family - W3C Working Draft 13 September 2002 [http://www.w3.org/TR/hlink/]
- specifikace je navržena *W3C Working Group*, jí oponovala *W3C Technical Architecture Group (TAG)*
- Princip: mapuje vybrané atributy na jejich přesně dané XLink protějšky
- tzn. i to, co normálně nemůže být XLinkem, se na XLink může "automaticky" mapovat
- Tento standard je kritizován jako "hybrid, který jedině dokazuje praktickou nevhodnost XLinku"
- dále viz např. diskuse na xmlhack.com [http://xmlhack.com]

VELLUM

- Nezávislá alternativa Simona St. Laurenta
- viz Very Extensible Linking Language Unafraid of Markup (VELLUM) [http://www.simonstl.com/projects/vellum/]
- Nevýhodou je poněkud rozvleklejší ("upovídaný", *verbose*) zápis, než u XLink
- VELLUM is aimed exclusively at "external" or out-of-line linking. This frees VELLUM from the constraints imposed by developer expectations for inline linking.
- VELLUM accepts the cost of a verbose form in exchange for the extensibility and precision that ele-

ment forms and indirection can offer.

- VELLUM does permit the use of existing abbreviations (like URIs and URI references), but simultaneously allows developers to specify more information than is carried in those identifiers.
- VELLUM also looks forward to the prospect of VELLUM processors which maintain state across multiple traversals and permit the creation of interactive hypertexts, not just collections of connections.

VELLUM - přednosti a nedostatky

- VELLUM attempts to provide a *general-purpose solution* to linking which addresses the complexities raised by the W3C's XPointer and XLink specifications by taking a very different approach. VELLUM *does not assume that URIs and URI references are adequate to the task of identifying resources*, representations, and fragments of representations, and strives to put XML hyperlinking on firmer but still approachable foundations. VELLUM supports and uses URIs and URI references, but offers *options that extend those capabilities*.
- VELLUM is *not a general-purpose solution to hypertext linking*. VELLUM is intended to be used in cases where *precision is important and verbosity is not a problem*. While VELLUM could conceivably be mixed with other vocabularies and used to define links within them, it is not designed explicitly for such use. VELLUM is more appropriate for use in cases like *external links and linkbases*, where the links are stored separately from the resources they connect. (VELLUM's designer hopes that a simpler mechanism for in-line linking will emerge to complement VELLUM.)
- VELLUM both builds on the *URI framework and goes beyond* the URI framework. URIs and URI references may be used within the VELLUM framework if the level of precision they provide is adequate, but developers can specify more information about issues like *content-negotiation* within the VELLUM framework if they choose. VELLUM also makes it possible to explicitly specify whether a connection *involves an abstract resource or a particular concrete representation*.
- VELLUM also makes it possible for developers to *create metadata which applies to their links in a local context*. While XLink uses URIs for everything from href to arcrole, VELLUM lets developers use more intelligible identifiers whose meaning is defined within a particular VELLUM context. While VELLUM may be more verbose than a comparable XLink linkbase, it should (if designed thoughtfully) be more readable.

VELLUM - ukázka

```
<piece xmlns="http://simonstl.com/ns/vellum" >
   <connections>
      <traverse>
         <from href="http://www.w3.org/TR/REC-xml#sec-common-syn" />
         <to href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames" />
      </traverse>
      <traverse>
         <from href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames" />
```

```
         <to href="http://www.w3.org/TR/REC-xml#sec-common-syn" />
      </traverse>
   </connections>
</piece>
```

VELLUM - ukázka (2)

```
<piece xmlns="http://simonstl.com/ns/vellum">
   <connections>
      <set id="namespaceUses" >
         <member href="http://www.w3.org/TR/REC-xml#sec-common-syn" />
         <member href="http://www.w3.org/TR/xmlschema-2/#QName" />
      </set>
      <traverse>
         <from ref="namespaceUses"/>
         <to href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames" />
      </traverse>
   </connections>
</piece>
```

VELLUM - ukázka (3)

```
<piece xmlns="http://simonstl.com/ns/vellum">
   <targets>
      <target id="_xml-names" representation= "http://www.w3.org/TR/REC-xml#sec-co
      <target id="_xml-schema-qnames" representation= "http://www.w3.org/TR/xmlsch
      <target id="_xmlns-qual-names" representation= "http://www.w3.org/TR/REC-xml
   </targets>
   <connections>
      <set id="namespaceUses" >
         <member ref="_xml-names" />
         <member ref="_xml-schema-qnames" />
      </set>
      <traverse>
         <from ref="namespaceUses" />
         <to ref="_xmlns-qual-names" />
      </traverse>
   </connections>
</piece>
```

VELLUM - srovnání s RDF

koncept VELLUM out-of-line odkazů je podobný metadatům podle rámce RDF (odkazy jsou také meta-data...)

proto se k VELLUM vrátíme po prostudování RDF

Odkazy na další zdroje

- XLink Tutoriál na Zvonu (http://zvon.org)
- Specifikace XLink (http://www.w3.org/TR/xlink)
- IBM Developerworks/XML
- Portál XML.COM (např. článek *What is XLink?* http://www.xml.com/lpt/a/2000/09/xlink/index.html)
- xmlhack.com [http://xmlhack.com]

XPointer

XML Pointer Language (XPointer)

W3C Working Draft 16 August 2002

- Definuje mechanizmus **adresování** (tj. "pointing" - ne odkazování!) **v rámci XML dokumentu**
- Rozšiřuje koncept *kotvy* (anchor) známé z HTML - *#jménoKotvy*
- Umožňuje **explicitní adresování navigací** i **implicitní adresování dotazováním**
- Umožňuji adresování i v rámci textových uzlů
- Umožňuje rozlišit adresu **bodu** (point) a **úseku** (range) v XML dokumentu
- Jako základ pro specifikaci adresy bere jazyk `XPath` [http://www.instantweb.com/foldoc/foldoc.cgi?XPath]

Aktuální specifikace skupiny XPointer

POZOR - specifikace XPointer se v současnosti rozpadla na 4 následovníky:

- *XPointer Framework* - This specification defines the XML Pointer Language (XPointer) Framework, an extensible system for XML addressing that underlies additional XPointer scheme specifications. The framework is intended to be used as a basis for fragment identifiers for any resource whose Internet media type is one of text/xml, application/xml, text/xml-external-parsed-entity, or application/xml-external-parsed-entity. Other XML-based media types are also encouraged to use this framework in defining their own fragment identifier languages.

- *XPointer element() Scheme* - The XPointer `element()` [http://www.instantweb.com/foldoc/foldoc.cgi?element()] scheme is intended to be used with the *XPointer Framework* to allow basic addressing of XML elements.
- *XPointer xmlns() Scheme* - The XPointer xmlns() scheme is intended to be used with the *XPointer Framework* to allow correct interpretation of namespace prefixes in pointers, for instance, namespace-qualified scheme names and namespace-qualified element or attribute names appearing within scheme data.
- *XPointer xpointer() Scheme* - The XPointer xpointer() scheme is intended to be used with the *XPointer Framework* to provide a high level of functionality for addressing portions of XML documents. It is based on XPath, and adds the ability to address strings, points, and ranges in accordance with definitions provided in DOM 2: Range.

XPointer - terminologie

XPointer - terminologie

pojem Point

`point` [http://www.instantweb.com/foldoc/foldoc.cgi?point] - bod, daný svým kontejnerem (`container` [http://www.instantweb.com/foldoc/foldoc.cgi?container]) a celočíselným indexem (pozicí) v rámci kontejneru

bod může mít *explicitní* nebo *implicitní* hranici

může být typu `node-point` [http://www.instantweb.com/foldoc/foldoc.cgi?node-point] nebo `character-point` [http://www.instantweb.com/foldoc/foldoc.cgi?character-point]

dále viz specifikace - typ Point [http://www.w3.org/TR/xptr-xpointer/#dt-point]

viz také tutoriál node-point [http://zvon.org/xxl/xpointer/tutorial/OutputExamples/xml30_out.xml.html]

a character-point [http://zvon.org/xxl/xpointer/tutorial/OutputExamples/xml40_out.xml.html]

pojem Range

`range` [http://www.instantweb.com/foldoc/foldoc.cgi?range] - je obsah mezi dvěma body V RÁMCI jednoho dokumentu nebo jedné externí analyzované entity

dále viz specifikace - typ Range [http://www.w3.org/TR/xptr-xpointer/#dt-range]

pojem Location

`location` [http://www.instantweb.com/foldoc/foldoc.cgi?location] - "místo" - může být DOM

[http://www.instantweb.com/foldoc/foldoc.cgi?point], `range` [http://www.instantweb.com/foldoc/foldoc.cgi?range]

dále viz specifikace - typ Location [http://www.w3.org/TR/xptr-xpointer/#dt-location]

XPointer - ukázky

XPointer - ukázka (1)

```
<link xmlns:xlink="http://www.w3.org/2000/xlink"
      xlink:type="simple"
      xlink:href="mydocument.xml#xpointer(//AAA/BBB[1])"/>
```

XPointer - ukázka (2)

Xpointer výraz `b2/3` [http://www.instantweb.com/foldoc/foldoc.cgi?b2/3] nebo `xpointer(id('b2')/*[3])` [http://www.instantweb.com/foldoc/foldoc.cgi?xpointer(id('b2')/*[3])] vybere z následujícího dokumentu

```
<AAA>
   <BBB myid="b1" bbb="111"> Text in the first element BBB.</BBB>
   <BBB myid="b2" bbb="222">
      Text in another element BBB.
      <DDD ddd="999"> Text in more nested element.</DDD>
      <DDD ddd="888"> Text in more nested element.</DDD>
      <DDD ddd="777"> Text in more nested element.</DDD>
   </BBB>
   <CCC ccc="123" xxx="321"> Again some text in some element.</CCC>
</AAA>
```

...třetí uzel DDD ddd="777"...

Zvon XPointer Tutorial

Zvon XPointer Tutorial - http://www.zvon.org/xxl/xpointer/tutorial/OutputExamples/xpointer_tut.html

Jazyk XSLT

Souvislosti, historie

- XSLT (eXtensible Stylesheet Language Transformation) [http://w3.org/style/XSL] je jazyk pro specifikaci *transformací XML dokumentů* na (obvykle) XML výstupy, případně *textové*, *HTML* či jiné výstupní formáty.

- Původní aplikační oblastí byla transformace XML dat na XSL:FO (formátovací objekty), tedy *vizualizace XML*.
- XSLT byl tedy součástí specifikací XSL [http://w3.org/style/XSL] (eXtensible Stylesheet Language).
- Později se z XSL vyčlenil a začal být chápán jako univerzální jazyk popisu obecných XML->XML(txt, HTML) transformací.
- Aktuální verze je dána specifikací XSLT 1.0 [http://www.w3.org/TR/xslt].

  Práce na verzi 1.1 byly zastaveny ve prospěch vývoje XSLT 2.0 [http://www.w3.org/TR/xslt20].

Hlavní principy

- XSLT je *funkcionálním jazykem*, kde *redukční pravidla* mají podobu *šablon*, které předepisují, jak se *uzly* zdrojového dokumentu přepisují do výstupního dokumentu.
- Specifikace XSLT transformace je obsažena v tzv. *stylu (stylesheet)*, což je XML dokument tvořený podle syntaxe XSLT. Kořenovým elementem je stylesheet nebo transformation (to jsou synonyma).
- XSLT styl obsahuje tzv. *šablony (template)*.
- Šablony mají *výběrovou část* - která reprezentuje levou stranu funkcionálního redukčního pravidla a *konstrukční část* představující pravou stranu red. prav.
- Výběrovou část tvoří atribut match šablony.

  Konstrukční část představuje tělo elementu šablony.
- Vlastní transformace pak znamená, že interpreter XSLT stylů (*XSLT procesor*, *XSLT engine*) bere uzly vstupního dokumentu, vyhledá k nim vhodnou šablonu - podle její výběrové části - a vyprodukuje výsledek odpovídající konstrukční části pravidla daného touto šablonou.

Hlavní informační zdroje - specifikace, reference, tutoriály, FAQ

- XSLT 1.0 W3C Recommendation: http://www.w3.org/TR/xslt
- *What is XSLT?* na XML.COM: http://www.xml.com/pub/a/2000/08/holman/index.html
- Mulberrytech.com XSLT Quick Reference (2xA4, PDF): http://www.mulberrytech.com/quickref/XSLTquickref.pdf
- Dr. Pawson XSLT FAQ: http://www.dpawson.co.uk/xsl/xslfaq.html
- Zvon XSLT Tutorial: http://zvon.org/xxl/XSLTutorial/Books/Book1/index.html

Syntaxe XSLT

Struktura celého XSLT stylu

Kořenový element xsl:transform nebo xsl:stylesheet uzavírá celý XSLT styl a specifikuje NS prefix pro XSLT elementy.

V kořenovém elementu je:

- Deklarace *parametrů* (a jejich implic. hodnoty) - elt. xsl:param . Parametry lze nastavit při volání XSLT procesoru - např. **java net.sf.saxon.Transform -o outfile.xml infile.xml style.xsl -Dparam=paramvalue** [http://www.instantweb.com/foldoc/foldoc.cgi?java net.sf.saxon.Transform -o outfile.xml infile.xml style.xsl -Dparam=paramvalue]
- Deklarace a inicializace *proměnných* - elt. xsl:variable - proměnné jsou de facto totéž, co parametry, ale nejsou nastavitelné zvenčí.
- Je třeba si uvědomit, že XSLT (bez procesorově-specifických rozšíření) je čistý funkcionální jazyk, tj. aplikace šablony nemá vedlejší efekt -> proměnné lze přiřadit jednou, pak už jen číst!
- Deklarace (formátu) výstupu - elt. xsl:output
- ...kromě toho tam mohou být další, méně používané XSL elementy - viz např. dokumentace SAXO-Nu [http://saxon.sf.net]
- pak následují vlastní šablony - elt. xsl:template

XSLT šablony

Šablona (*template*) je specifikace *který uzel přepsat* a *na co (jak)*.

Který uzel se přepisuje, je dáno *atributem* match .

Na co se přepisuje, je uvedeno v *těle šablony*.

Šablona může být explicitně *pojmenovaná* (*named template*), v tom případě ji lze volat přímo/explicitně pomocí xsl:call-template .

Sémantika XSLT

XSLT - postup zpracování vstupního dokumentu

- Nejdříve se za aktuální uzel zvolí kořen, tj. uzel odpovídající XPath výrazu /

- Najde se šablona (*explicitní* nebo *implicitní* - viz např. XSLT/XPath Quick Reference [http://www.mulberrytech.com/quickref/XSLTquickref.pdf]), jejíž match atribut chápaný jako XPath predikát vrátí v kontextu aktuálního uzlu true (tedy tzn. "matchuje" aktuální uzel).
- Pokud je jich více - nastává *nejednoznačnost* - pak je indikována chyba.
- Pokud je taková šablona právě jedna, aplikuje se, což znamenápřenesení jejího obsahu do výstupního *result tree fragmentu.*

XSLT - pořadí volání šablon

Je možné je specifikovat:

1. Přímo/explicitně voláním (pojmenované) šablony - což ale odpovídá spíše *přístupu procedurálních jazyků*, takže se tomu spíše vyhýbáme.
2. Nepřímo/implicitně tím, že se zavolá šablona, jejíž vzor (obsah atr. match) "pasuje" ("matchuje") na vybraný uzel - **funkcionální přístup**. Výběr uzlu se přitom děje opět:
   - Explicitně ("řízeně") uvedením atributu select u apply-templates . Takto můžeme vybrat jak dceřinné elementy, tak dceřinné uzly, tak jakékoli jiné uzly odpovídající XPath výrazu uvedenému v select .
   - Implicitně, necháme-li procesor sám "si uzel vybrat" (u apply-templates neuvádíme select). V tomto případě se ale vybírají pouze *dceřinné elementy* kontextového uzlu.

XSLT - specifikace výstupu/"výsledku" šablony

- Výstupem aplikace šablony je část tzv. *result tree fragmentu*.
- Výstupy jednotlivých šablon se "skládají" na to místo *result tree fragmentu*, který odpovídá pořadí volání šablon.
- Výstup celé transformace pak směřuje standardně do jednoho proudu, kde se z výstupního proudu událostí generuje výsledný (XML, text, HTML) dokument.
- Výstup bývá procesorem primárně generován jako sled událostí (např. SAX2), které jsou až druhotně převáděny na výsledný dokument - s uplatněním výstupního kódování, atd.

XSLT - výstup textových uzlů

Jak dostat text (textový uzel) na výstup?

1. Vepsat text přímo (jako literál) do výstupu (konstrukční části) šablony. Pozor na bílé znaky (mezery, CR/LF)!

2. vepsat text přímo (jako literál) do výstupu šablony. Pozor na bílé znaky (mezery, CR/LF)!
3. do speciálního elt. <xsl:text>textový uzel</xsl:text> . Bílé znaky jsou v něm vždy zachovány/respektovány!

Implicitní šablony

Implicitní šablony jsou "vestavěné" v každém korektním procesoru XSLT:

- aby byly (alespoň jistým standardním "fallback" způsobem) zpracovány základní struktury (procházení stromu dokumentu)
- abychom "ušetřili psaní" často používaných šablon (ignorování komentářů a PI).
- Jsou překrytelné, abychom mohli chování změnit uvedením *vlastní šablony*, která bude mít stejnou (nebo překrývající se) klauzuli match= .

Přehled implicitních šablon

- "Default tree (do-nothing) traversal":

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

- "Default tree (do-nothing) traversal for specified mode":

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

Přehled implicitních šablon (2)

- "Copy text nodes and attributes" (do výsledku zkopíruje textové uzly a atributy):

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

- "Ignore PIs and comments" ignoruje (nezahrnuje do výsledku PI a komentáře):

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

Vybrané XSLT konstrukce podrobněji

Generování pevně daného elementu s atributy

Cíl: Vygenerovat na výstup předem daný element (s předem známým jménem), ale s atributy s hodnotami kalkulovanými při transformaci.

Řešení: Použít normální postup - literal result element - a hodnoty atributy specifikovat jako tzv. *attribute value templates (AVT)*:

Vstup:

```
<link ref="odkaz_dovnitr_dok">
   ...
</link>
```

Šablona:

```
<xsl:template match="link">
   <a href="#{@ref}"> ... </a>
</xsl:template>
```

Transformuje odkaz link na a , hodnotu atributu href spočte tak, že před hodnotu původního atributu ref přidá znak #

Generování elementu s kalkulovaným názvem i atributy

Cíl: Vygenerovat na výstup element, jehož název, atributy i obsah předem - při psaní stylu - neznáme.

Řešení: Použít do konstrukční části šablony xsl:element :

Vstup:

```
<generate element="elt_name"> ... </generate>
```

Šablona:

```
<xsl:template match="generate">
   <xsl:element name="@element">
      <xsl:attribute name="id">ID1</xsl:attribute>
   </xsl:element>
</xsl:template>
```

Vytvoří element s názvem elt_name , opatří jej atributem id="ID1" .

Řízení chodu transformace uvnitř šablony - větvení

Cíl: Větvit generování výstupu na základě podmínky.

Řešení: Použít do konstrukční části šablony větvení - jednoduché xsl:if nebo vícecestné xsl:choose / xsl:when / xsl:otherwise :

Vstup:

```
<rohlik cena="5"> ... </rohlik>
```

Šablona:

```
<xsl:template match="rohlik">
   <p>
      <xsl:if test="cena>2">
         <span class="expensive">Drahý</span>
      </xsl:if> rohlík - cena <xsl:value-of select="@cena"/> Kč </p>
</xsl:template>
```

Vytvoří element p , do něj vloží info o rohlíku - se zvýrazněním, je-li drahý.

Řízení chodu transformace uvnitř šablony - vícecestné větvení

Vstup:

```
<rohlik cena="5"> ... </rohlik>
<rohlik cena="2"> ... </rohlik>
<rohlik cena="0.9"> ... </rohlik>
```

Šablona:

```
<xsl:template match="rohlik">
   <p>
      <xsl:when test="cena>2">
         <span class="expensive">Drahý</span>
      </xsl:when>
      <xsl:when test="cena<1">
         <span class="strangely-cheap">Podezřele levný</span>
      </xsl:when>
      <xsl:otherwise>
         <span class="normal-price">Běžný</span>
      </xsl:otherwise> rohlík - cena <xsl:value-of select="@cena"/> Kč </p>
</xsl:template>
```

Odfiltruje dvě extrémní úrovně ceny - pro xsl:otherwise zůstane „normální“ cena.

Řízení chodu transformace uvnitř šablony - cykly

Cíl: Větvit generování výstupu na základě podmínky.

Řešení: Použít do konstrukční části šablony větvení - jednoduché xsl:if nebo vícecestné xsl:choose / xsl:when / xsl:otherwise :

Vstup:

```
<pecivo>
   <rohlik cena="5"> ... </rohlik>
   <rohlik cena="2"> ... </rohlik>
   <rohlik cena="0.9"> ... </rohlik>
</pecivo>
```

Šablona:

```
<xsl:template match="pecivo">
   <xsl:for-each select="rohlik">
      <p>Rohlík - cena <xsl:value-of select="@cena"/> Kč</p>
   </xsl:for-each>
</xsl:template>
```

Vytvoří element p , do něj vloží info o rohlíku - se zvýrazněním, je-li drahý.

Pozor: Konstrukce xsl:for-each má typicky procedurální charakter, je dobré s ní šetřit. D8v8 toti6 minumum flexibility na obsah iterované množiny uzlů - tj. *předem musím vědět, co tam bude*.

Pokročilá témata

Režimy (módy) zpracování

Motivace: Módy umožňují mít paralelně sadu šablon se stejnými vzory match , používaných ale pro různé účely, např.:

- jedna sada pro generování obsahu (*index*) dokumentu
- druhá pro formátování plného textu dokumentu

Při explicitním vyvolání aplikace šablon (apply-templates) lze uvést mód (atributem mode=):

- uvede-li se, aplikují se pouze šablony se stejným módem, jaký byl uveden v xsl:apply-templates mode="mód" .

- neuvede-li se, aplikují se pouze šablony *bez* specifikace módu (bez atributu mode=).

Deklarace a volání pojmenovaných šablon

Deklarace - xsl:template name="jmeno_sablony"

Šablona smí obsahovat deklarace parametrů:

- <xsl:param name="jmenoParametru"/>

Volání - <xsl:call-template name="jmenoSablony">

volání smí specifikovat parametry:

- <xsl:with-param name="jmenoParametru" select="hodnotaParametru"/> nebo
- <xsl:with-param name="jmenoParametru">hodnota parametru</xsl:with-param>

Automatické (generované) číslování

Vložíme-li do konstrukční části šablony (do těla šablony) element xsl:number , zajistí nám vygenerování čísla daného čítačem.

Je možné uvést, podle čeho se má číslovat, např.:

- pořadového čísla zdrojového elementu v rámci jeho rodičovského elementu

  - a to i víceúrovňově, např. číslo kapitoly 1.1. apod.

Příklad 1.12. Automatické číslování podle pozice elementu

Aplikujeme-li tento styl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
   <xsl:template match="/">
      <html>
         <body>
            <xsl:for-each select="devguru_staff/programmer">
               <xsl:number value="position()" format="1. " />
               <xsl:value-of select="name" />
               <br/>
            </xsl:for-each>
```

```
        </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

na následující zdrojový soubor

```
<devguru_staff>
    <programmer>
        <name>Bugs Bunny</name>
        <dob>03/21/1970</dob>
        <age>31</age>
        <address>4895 Wabbit Hole Road</address>
        <phone>865-111-1111</phone>
    </programmer>
    <programmer>
        <name>Daisy Duck</name>
        <dob>08/09/1949</dob>
        <age>51</age>
        <address>748 Golden Pond</address>
        <phone>865-222-2222</phone>
    </programmer>
    <programmer>
        <name>Minnie Mouse</name>
        <dob>04/13/1977</dob>
        <age>24</age>
        <address>4064 Cheese Factory Blvd</address>
        <phone>865-333-3333</phone>
    </programmer>
</devguru_staff>
```

dostaneme výslednou HTML stránku (nebrat v úvahu odsazení - to bude jiné...)

```
<html>
    <body>1. Bugs Bunny<br>
          2. Daisy Duck<br>
          3. Minnie Mouse<br>
    </body>
</html>
```

Automatické číslování (2)

Příklad 1.13. Automatické víceúrovňové číslování

Aplikujeme-li tento styl

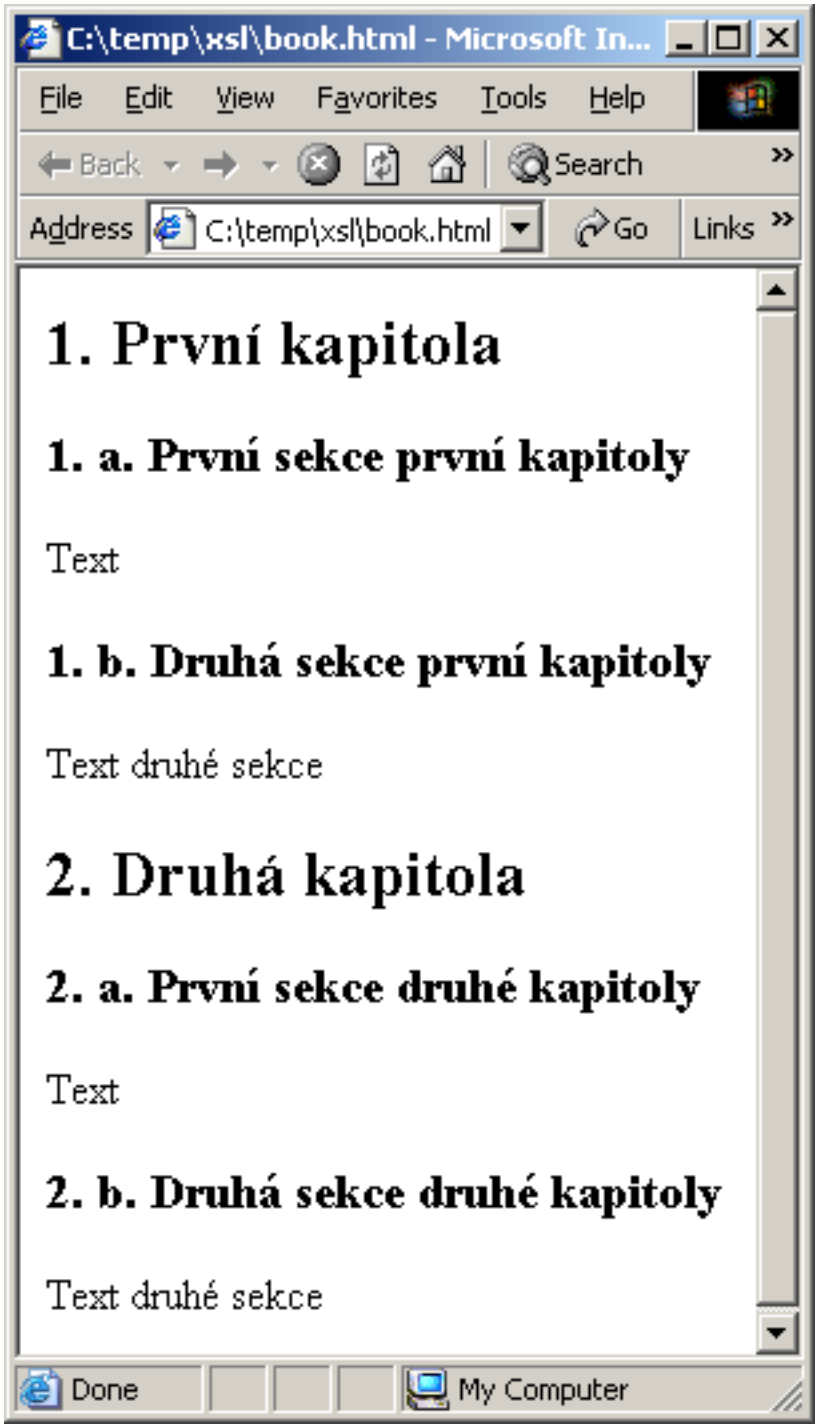
```
<xsl:stylesheet
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0">
    <xsl:template match="/book">
        <html>
            <body>
                <xsl:for-each select="chapter">
                    <h2>
                        <xsl:number count="chapter" format="1. "/>
                        <xsl:value-of select="title" />
                    </h2>
                    <xsl:for-each select="sect1">
                        <h3>
                            <xsl:number count="chapter" format="1. "/>
                            <xsl:number count="sect1" format="a. "/>
                            <xsl:value-of select="title" />
                        </h3>
                        <xsl:apply-templates select="para"/>
                    </xsl:for-each>
                </xsl:for-each>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

na následující zdrojový soubor

```
<book>
    <title>Moje nová kniha</title>
    <chapter>
        <title>První kapitola</title>
        <sect1>
            <title>První sekce první kapitoly</title>
            <para>Text</para>
        </sect1>
        <sect1>
            <title>Druhá sekce první kapitoly</title>
            <para>Text druhé sekce</para>
        </sect1>
    </chapter>
    <chapter>
        <title>Druhá kapitola</title>
        <sect1>
            <title>První sekce druhé kapitoly</title>
            <para>Text</para>
        </sect1>
```

```
        <sect1>
            <title>Druhá sekce druhé kapitoly</title>
            <para>Text druhé sekce</para>
        </sect1>
    </chapter>
</book>
```

dostaneme výslednou HTML stránku



Co používat raději?

- Preferovat funkcionální přístup - např. xsl:template match= a xsl:apply-templates select=
- před procedurálním přístupem - xsl:template name= a xsl:call-template name=

Používat módy zpracování (xsl:template ... mode= a xsl:apply-templates ... mode=)

módy lze dobře kombinovat s funkcionálním přístupem:

- xsl:apply-templates select=... mode=...
- xsl:template match=... mode=...

Znovupoužitelnost stylů

Co pro ni můžeme udělat?

- Členit styly do menších znovupoužitelných celků (souborů) a podle potřeby je vřazovat pomocí xsl:include a nebo, ještě lépe, xsl:import - protože import upřednostňuje šablony uvedené přímo v základním stylu nad šablonami importovanými.

Podrobněji viz příspěvek TP pro DATAKON 2001 - fulltext příspěvku [/~tomp/xml03/pitner.doc] a slidy [/~tomp/xml03/prezentace.ppt].

Návrhové vzory

Identická transformace 1 (nepřevede do výsledku atributy kořenového elementu!) http://wwbota.free.fr/XSLT_models/identquery.xslt

Identická transformace 2 http://wwbota.free.fr/XSLT_models/identquery2.xslt

Identická transformace s potlačením elementů, které nemají na ose // (v dceřinných uzlech ani jejich potomcích) žádné textové uzly http://wwbota.free.fr/XSLT_models/suppressEmptyElements.xslt

Nahradí atributy pomocí elementů http://wwbota.free.fr/XSLT_models/attributes2elements.xslt

Dtto, ale elementy vzniklé z atributů jsou ve zvláštním jmenném prostoru xslt/attributes2elements.xslt [/~tomp/xml03/xslt/attributes2elements.xslt]

Reverzní transformace xslt/elements2attributes.xslt [/~tomp/xml03/xslt/elements2attributes.xslt]

Odkazy na pokročilá témata

XSLT Design Patterns - výběr [http://www.dpawson.co.uk/xsl/sect1/N169.html]

The Functional Programming Language XSLT [http://www.topxml.com/xsl/articles/fp/1.asp]

Základní problémy efektivního ukládání a zpracování XML dat

Základy efektivního ukládání XML dat

Přednáška Petra Adámka (PDF) [../../../PB138-2003-XML.Databaze.pdf]

Rozhraní pro práci s XML databázemi

Rozhraní XML:DB

- Specifikováno konsorciem XML:DB [http://xmldb.org]
- Podobná koncepce jako JDBC [http://java.sun.com/products/jdbc/]
- Rozhraní je specifikováno na poměrně abstraktní úrovni, implementační detaily jsou skryty.
- Základní objekty:
  - `Driver` - podobně jako JDBC Driver - abstrahuje přístup ke konkrétnímu DBS, implementuje rozhraní Database
  - `DatabaseManager` - řídí zavádění a správu jednotlivých ovladačů (Driver) databázových systémů
  - `Collection` - kolekce XML dokumentů v databázi. Konceptuálně srovnatelné s relační tabulkou (či celou databází). Kolekce totiž mohou být libovolně vnořené.
  - Services - rozhraní konkrétních služeb. Bez nich by XML:DB takřka nemělo smysl - teprve služby definují, co databáze „umí“. Typickou službou je např. `XPathQueryService` na vyhledávání dokumentů a jejich částí přes XPath. Další službou je např. `XUpdateQueryService`.
  - `Resource` - zhruba odpovídá JDBC resource. Obecně „nějaký“ zdroj - nemusí být jen XML, ale i binární. Je-li XML, pak např. SAX, DOM, XML text...

Vrstvy XML:DB API

Rozhraní XML:DB je pro pohodlí programátora členěno do úrovní. Vždy si jednu z nich vybereme a využíváme její nabídky:

- XML:DB Core Level 0 - musí implementovat všechny DBS. Obsahuje základní rozhraní pro *kolekce* (collections), *zdroje* (resources), and *služby* (services).
- XML:DB Core Level 1 - navíc obsahuje `XPathQueryService`.

Ukázka XML:DB programu

Příklad 1.14. Příklad programu využívajícího XML:DB

```
import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;
import org.xmldb.api.*;
public class Query {
    public static void main(String[] args) throws Exception {
        Collection col = null;
        try {
            String driver = null;
            String prefix = null;

            if ( ( args.length == 1 ) && args[0].equals("dbxml") ) {
                driver = "org.dbxml.client.xmldb.DatabaseImpl";
                prefix = "xmldb:dbxml:///db/";

            } else {
                driver = "org.xmldb.api.reference.DatabaseImpl";
                prefix = "xmldb:ref:///";
            }

            Class c = Class.forName(driver);
            Database database = (Database) c.newInstance();

            if ( ! database.getConformanceLevel().equals("1") ) {
                System.out.println("This program requires a Core Level 1 XML:DB "
                System.exit(1);
            }
            DatabaseManager.registerDatabase(database);

            col = DatabaseManager.getCollection(prefix + "addresses");
            String xpath = "/address[@id = 1]";

            XPathQueryService service = (XPathQueryService) col.getService("XPathQ

            ResourceSet resultSet = service.query(xpath);
            ResourceIterator results = resultSet.getIterator();

            while (results.hasMoreResources()) {
                Resource res = results.nextResource();
                System.out.println((String) res.getContent());
            }
        } catch (XMLDBException e) {
            System.err.println("XML:DB Exception occurred " + e.errorCode + " " +
        } finally {
            if (col != null) { col.close(); }
        }
    }
}
```

Použití XUpdate v databázích s XML:DB

Příklad 1.15. Příklad modifikace pomocí XUpdate

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
   <xupdate:update select="/address[@id = 1]/name/last">Herman</xupdate:update>
</xupdate:modifications>
```

Tento XUpdate dotaz nahradí příjmení za „Herman“.

Implementace XML:DB rozhraní

Apache Xindice

- Open-source referenční implementace XML:DB.
- Původně nazývána dbXML.
- Velmi dobré úvodní informace lze získat v Introduction to Xindice [http://www-106.ibm.com/developerworks/web/library/wa-xindice.html]

Ukázka interakce s Xindice

Kolekce mohou nebo nemusí mít XML Schema.

Příklad 1.16. Vytvoření a manipulace s kolekcí „mojedok“

```
xindiceadmin ac -a /db -n mo
```

systém odpoví

```
Created: /db/mojedok
```

dotaz na seznam kolekcí (Listing Collection - lc) v `/db`

```
xindiceadmin lc -c /db
```

zrušení kolekce (Delete Collection - dc) v `/db`

```
xindiceadmin dc -c /db -n mojedok
```

Ukázka interakce s Xindice (2)

Příklad 1.17. Přidání, získání, zrušení dokumentu do/z kolekce „mojedok“

Přidání souboru (-Add Document, -File [filename], -n specifikuje klíč) :

```
xindice ad -c /db/mojedok -f c:/devel/mojedokumenty/md.xml -n mujklic
```

Získání dokumentu zpět (-Retrieve Document, -File)

```
xindice rd -c /db/mojedok -f c:/devel/mojedokumenty/md.out.xml
```

Zrušení dokumentu z databáze (-Delete Document)

```
xindice dd -c /db/mojedok -n mujklic
```

Ukázka interakce s Xindice (3)

Příklad 1.18. Dotazování na kolekci „mojedok“

zrušení kolekce (Delete Collection - dc) v `/db`

```
xindice xpath -c /db/mojedok -q /parts/part[@sku="101"]
```

eXist

`eXist` je podobně jako `Xindice` open-source databáze podporující XML:DB.

Je možné ji provozovat jako:

- samostatně běžící (standalone) server, přístupný soketovým spojením (XML-RPC, HTTP)
- jako in-process (embedded) -server běžící v témže běhu JVM jako aplikace, která jej používá
- jako webová aplikace - .war archív, který se instaluje (deploy) na servletový kontejner (např. Tomcat, Jetty, Bajie...)
- eXist má Jetty server přibalen v instalačním balíku, viz eXist download [???].

eXist: instalace a spuštění

Je možno instalovat na Win NT/2000, Linuxu...

Postupujeme přesně podle instrukcí v eXist Quickstart [http://exist-db.org/quickstart.html].

Doporučuji (odzkoušeno na Win 2000 Pro):

- spustit **java -jar eXist-0.9.1-install.jar** [http://www.instantweb.com/foldoc/foldoc.cgi?java -jar eXist-0.9.1-install.jar]
- řídit se instalačními pokyny, instalovat např. do `\devel\eXist` [http://www.instantweb.com/foldoc/foldoc.cgi?\devel\eXist].
- přepnout se do instalačního adresáře, otevřít Command Shell/Prompt a spustit eXist přes Jetty webový server: **bin\startup.bat** [http://www.instantweb.com/foldoc/foldoc.cgi?bin\startup.bat].
- eXist ohlásí, že se spustil. Případné chybové hlášky loggeru ignorovat.

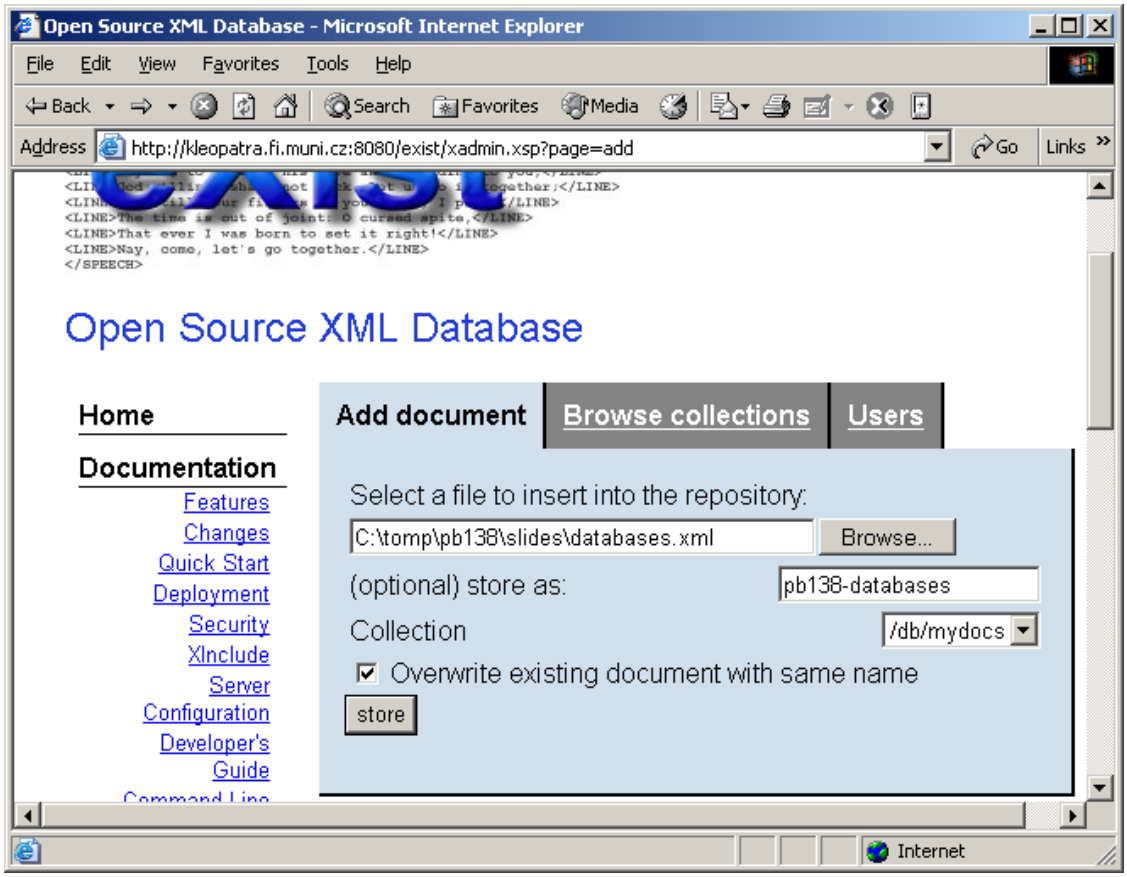
eXist: použití přes webové rozhraní

Pokud se v konfiguracích nic neměnilo, je služba eXist dostupná přes URL podobné tomuto: http://kleopatra.fi.muni.cz:8080/exist/. (tj. port 8080, cesta /exist)

Nyní můžeme vytvořit kolekci, přidat soubor, dotazovat se...

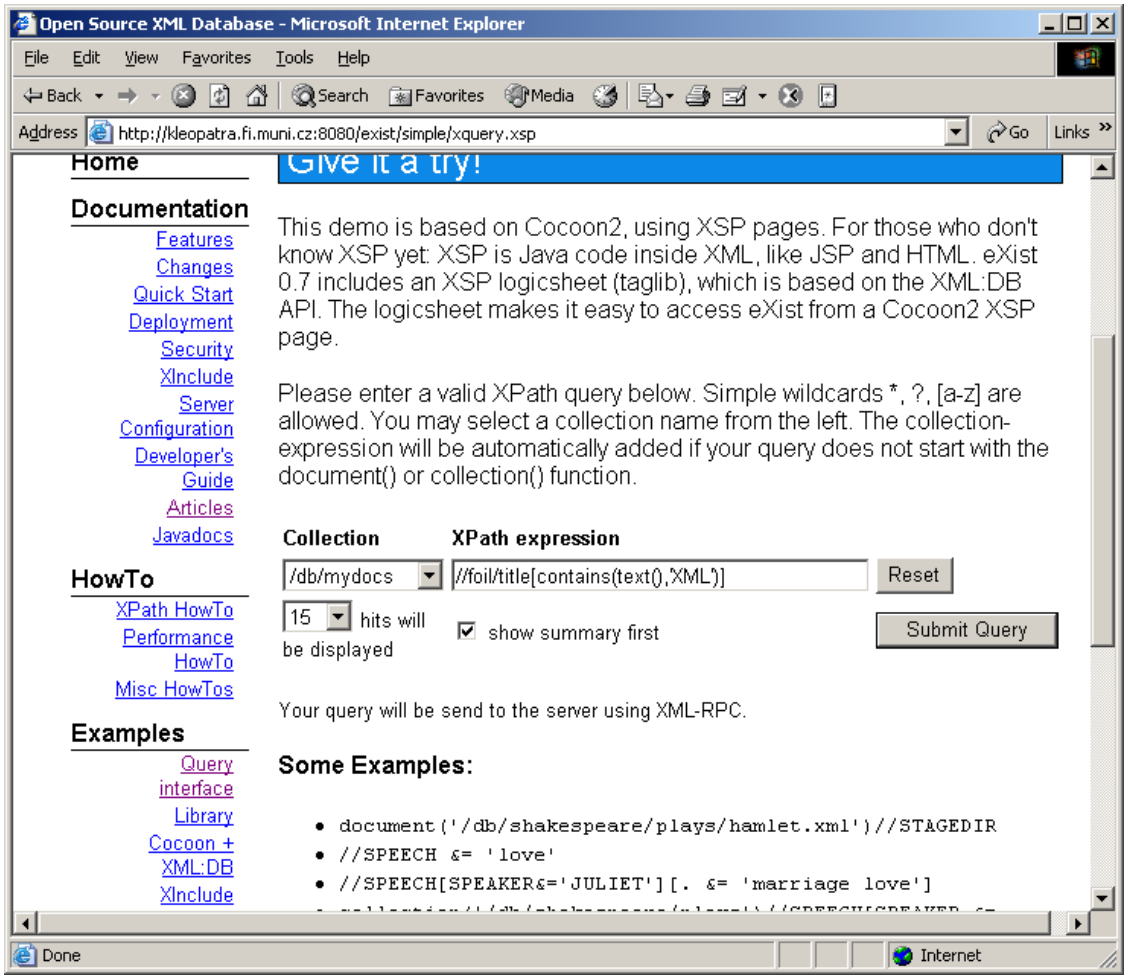
eXist: vložení dokumentu do kolekce

Vytvoříme kolekci `mydocs` a do ní přidáme dokument `databases.xml` [http://www.instantweb.com/foldoc/foldoc.cgi?databases.xml] (tyto slidy):



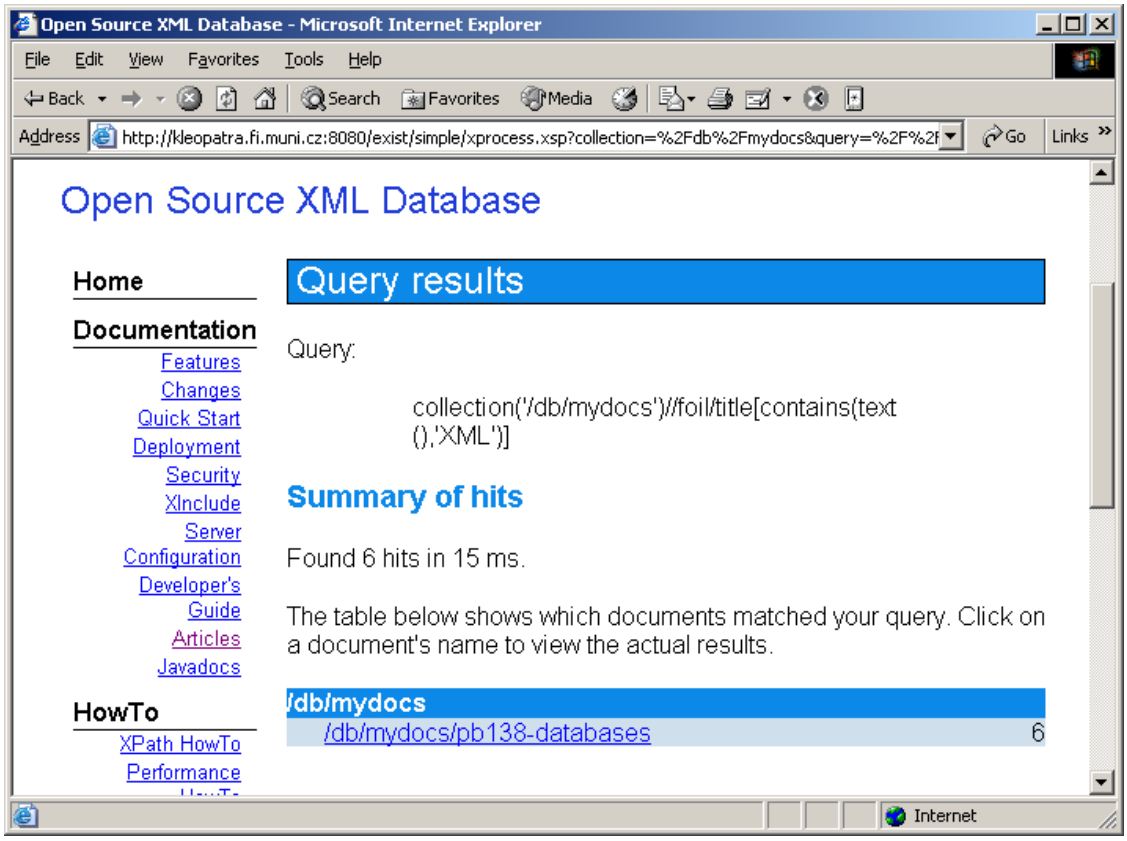
eXist: dotazování - zadání dotazu

Zadáme XPath dotaz, specifikujeme rozsah (ve které koleci hledat), uvedeme, kolik vyhovujících dokumentů v odpovědi vrátit.



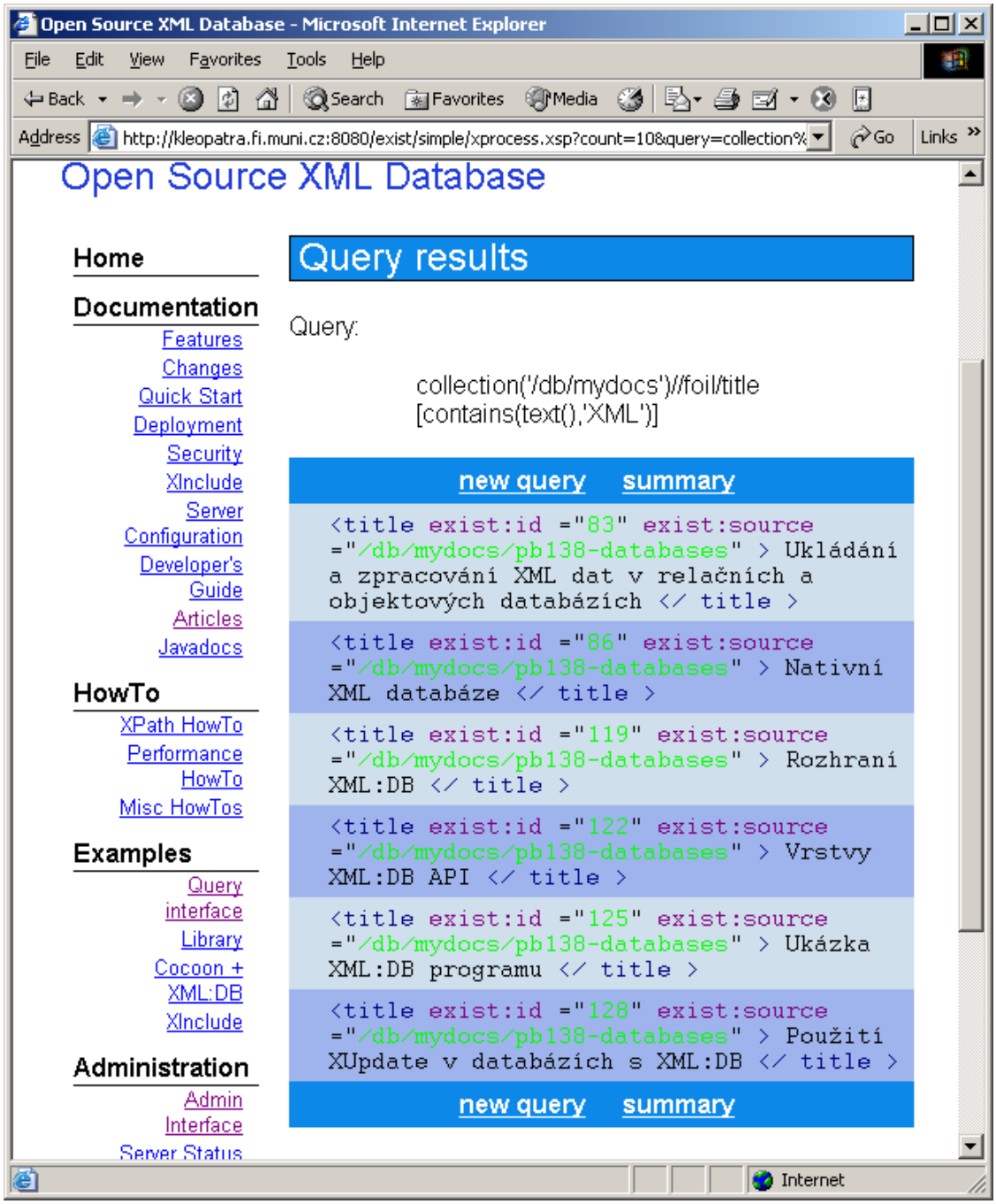
eXist: dotazování - sumarizovaný výsledek dotazu

eXist sdělí, ve kterých kolekcích které dokumenty vyhovují dotazu:



eXist: dotazování - prohlížení jednotlivých výsledků dotazu

eXist pro každý vyhovující dokument vrátí uzly, které dotazu vyhovují



Úvod k formátování

DocBook: příklad složitějšího značkování

- rozsáhlý projekt - poskytnout jednotný komplexní značkovací jazyk pro „vešekerou“ programátorskou dokumentaci
- nyní používáno k celé řadě jiných účelů - psaní článků (article), knih (book), jednotlivých kapitol

(chapter), sekcí (section, sectX)

- autorem je Norman Walsh (Sun Microsystems Inc.)
- podrobnosti, DTD, help, software, styly k dispozici viz docbook.org [http://docbook.org]
- pravděpodobně nejrozsáhlejší existující značkování pro logický popis dokumentu
- k DB existuje TDG (DocBook: The Definitive Guide) - také jako Windows Help [/~tomp/xml/tdg-en-2.0.7.chm]

DocBook: vrstvy a přizpůsobení

- DocBook lze používat jako základní (Full)
- zjednodušený (Simplified) nebo
- si jej přizpůsobit

přizpůsobení znamená:

- upravit DTD (přes parametrické entity)
- evt. upravit (XSL) styly
- XSL styly jsou upravovány na základě importu původního stylu a překrytí vybraných šablon

DocBook: styly

- k vizualizaci (a převodu do tiskových formátů) z napsaného DocBookového dokumentu lze použít XSL styly
- XSL styly jsou na http://docbook.sf.net
- v rámci toho je i přizpůsobení DocBook Slides
- styly formátují do HTML, XHTML, XSL:FO, ...
- zpracování lze parametrizovat bez přepisování stylů

Konceptuální, logické a fyzické formátování

Co a k čemu je formátování?

XML data jsou sice částečně „lidsky čitelná“, ale pro zpracování člověkem vyžadují transformaci do

kvalitně zobrazitelné podoby - *formátování*.

Formátování je proces transformace primárních dat do lidsky čitelné „prezentovatelné“ podoby, určené k zobrazení, hlasové interpretaci, atd.

Úrovně formátování

Rozlišujeme formátování na úrovni:

konceptuální
: maximum sémantiky v datech je zachováno, je přidána formátovací sémantika - např. struktura publikace, členení na kapitoly, sekce; datový obsah je vyfiltrován pro účely zobrazení (např. metadata jsou redukována). Příklad formátu - DocBook [http://docbook.org].

logické
: původní sémantiku již nelze rekonstruovat, je přidána další formátovací sémantika - např. členění na nadpisy, tabulky, styly, přibývají instrukce pro volbu písma, barvu, odsazení, mezery mezi odstavci... Formátovaný dokument je stále relativně univerzálně zobrazitelný a přenositelný. Příklad formátu - HTML [http://w3.org/HTML].

fyzické
: formátovaný dokument je v podstatě přesným popisem, jak vytisknout, zobrazit, hlasově přečíst dokument. Příklady formátu - PostScript, PDF.

Odkud kam sahají úrovně formátování?

- Hranice mezi jednotlivými úrovněmi nejsou ostré.

- Dokument na *vyšších formátovacích úrovních* (konceptuální) lze dále transformovat, zpracovávat...

  Část sémantiky původních dat je však již ztracena a naopak formátovací sémantika je doplněna.

  Viz např. formát DocBook - dokument v DocBooku už je článek, kniha, přednáška ve formě slidů... což původní zdrojová data nebyla.

- Dokument formátovaný do formátů *nižších úrovní* (logický, fyzický) není obvykle určen k dalšímu zpracování vyjma přípravy pro tisk, zobrazení.

  Viz např. HTML, PDF... (s jistými výjimkami)

Fáze formátování

Postup formátování a příklady nástrojů

Budeme uvažovat XML data a adekvátní nástroje se zaměřením na open-source.

data -> požadovaná data (filtrace)

Z primárních dat extrahujeme data potřebná. Vhodné nástroje:

filtrování proudu (SAX) událostí	poměrně nekomfortní, ale za běhu efektivní zpracování dat; pro pouhé filtrace elementů vyhoví; lze použít `XMLFilter`
manipulace se stromem dokumentu	`DOM`, `dom4j` a další reprezentace; náročnější na pamět i čas; dovoluje složitější manipulace
XSLT transformace	mocné, ale poněkud náročnější na zvládnutí
dotazovacím jazykem	vhodné řešení, ale vyžaduje stroj na interpretaci

Příklad 1.19. Příklad XSLT extrakce

Ze všech programátorů v Devguru chceme zobrazit jen mladší 30 let:

```
<?xml version="1.0" encoding="Windows-1250"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0" xmlns:saxon="http://icl.com/saxon"
     extension-element-prefixes="saxon">
   <xsl:strip-space elements="*" />
   <xsl:output method="xml" encoding="Windows-1250" indent="no" />
   <xsl:template match="programmer[number(age) &lt; 30]">
      <xsl:copy-of select="."/>
   </xsl:template>
   <xsl:template match="programmer"/>
   <xsl:template match="/">
      <young_programmers>
         <xsl:apply-templates/>
      </young_programmers>
   </xsl:template>
</xsl:stylesheet>
```

filtrovaná data -> konceptuální formát

Příklad 1.20. XSLT transformace do DocBooku

O všech mladších programátorech z Devguru napiš souhrnnou zprávu (styl je zkrácen):

```
<xsl:output method="xml" encoding="Windows-1250" indent="no"
     doctype-system="http://www.oasis-open.org/docbook/xml/simple/1.0/sdocbook.dtd
     doctype-public="-//OASIS//DTD Simplified DocBook XML V1.0//EN"/>
```

```
   <xsl:template match="programmer">
      <row>
          <entry>
             <xsl:value-of select="name"/>
          </entry>
          <entry>
             <xsl:value-of select="age"/>
          </entry>
      </row>
   </xsl:template>
   <xsl:template match="/">
      <article>
         <title>Seznam pracovníků mladších 30 let</title>
         <table>
            <title>Tabulka pracovníků mladších 30 let</title>
            <tgroup cols="2">
            <tbody>
               <row><entry>jméno</entry><entry>věk</entry></row>
               <xsl:apply-templates/>
            </tbody>
        </tgroup>
        </table>
      </article>
   </xsl:template>
   ...
```

Konceptuální formát -> logický formát

Příklad 1.21. XSLT transformace do DocBooku

Zprávu o mladších programátorech formátuj do HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
      <meta http-equiv="Content-Type" content= "text/html;charset=ISO-8859-1">
      <title>Seznam pracovn&iacute;k&#367; mlad&scaron;&iacute;ch 30 let</title>
      <link rel="stylesheet" href="html.css" type="text/css">
      <meta name="generator" content= "DocBook XSL Stylesheets V1.60.1">
   </head>
   <body bgcolor="white" text="black" link="#0000FF" vlink= "#840084" alink="#0000
   <div class="article" lang="en">
   <div class="titlepage">
       <div><div>
          <h2 class="title">
```

```
            <a name="d0e1"></a>Seznam pracovn&iacute;k&#367; mlad&scaron;&iacute;
          </h2>
        </div></div>
        <hr>
    </div>
    <div class="table">
       <a name="d0e4"></a>
       <p class="title">
          <b>Table 1. Tabulka pracovn&iacute;k&#367; mlad&scaron;&iacute;ch 30 let<
       </p>
       <table summary="Tabulka pracovn&iacute;k&#367; mlad&scaron;&iacute;ch 30 let
       <colgroup>
         <col>
         <col>
       </colgroup>
       <tbody>
         <tr><td> jm&eacute;no </td> <td> v&#283;k</td> </tr>
         <tr> <td> Minnie Mouse</td> <td> 24 </td> </tr>
       </tbody>
       </table>
    </div>
    </div>
    </body>
</html>
```

Logický formát -> fyzický formát

Výsledek zobrazený v prohlížeči

Formátování a výstupní média

Výstup na běžnou obrazovku, web (HTML, plaintext, RTF)

- k transformaci pro tato média lze použít XSLT (do plaintextu, HTML, XHTML...), speciálních nástrojů (např. RTF FormattingKit, viz RTF FormattingKit web [http://www.schema.de/sitehtml/site-e/xmlnach0.htm] pro RTF)
- pokud je formátování určeno k prohlížení na webu (prohlížečem), je možné pro specifikaci stylu použít konvenci (<?xml-stylesheet type="text/xsl" href="somestylesheet.xsl"?>)
- totéž lze použít i pro CSS styly
- pro XML je použitelná verze CSS2

(X)HTML, RTF, plaintext (2)

- k transformaci pro tato média lze použít XSLT (do plaintextu, HTML, XHTML...), speciálních nástrojů (např. RTF FormattingKit, viz RTF FormattingKit web [http://www.schema.de/sitehtml/site-e/xmlnach0.htm] pro RTF)
- pokud je formátování určeno k prohlížení na webu (prohlížečem), je možné pro specifikaci stylu použít konvenci typu <?xml-stylesheet type="text/xsl" href="somestylesheet.xsl"?>.
- přesný popis této konvence [http://web3.w3.org/TR/xml-stylesheet/]
- totéž lze použít i pro styly CSS [http://www.w3.org/Style/CSS] (pro XML je použitelná verze CSS2)
- dobré informační zdroje k použití CSS pro XML:
  - specifikace CSS2 [http://www.w3.org/Style/CSS]
  - CSS 2 Tutorial (zvon.org) [http://www.zvon.org/xxl/CSS2Tutorial/General/htmlIntro.html]
  - Building Documents with XML, XSL, and CSS [http://www.siteexperts.com/tips/xml/ts01/page1.asp]
  - displaying xml: working with cascading style sheets [http://www.javertising.com/webtech/cssxml.htm]

  Validace dokumentů s CSS styly:CSS Validátor na W3C: http://jigsaw.w3.org/css-validator/

Výstup pro tisk (PDF, TeX)

- obvyklý postup:
  - transformace XML -- XSL:FO pomocí XSLT
  - rendering XSL:FO do PDF/PS pomocí Apache FOP nebo jiného (komerčního) nástroje
  - nebo přes PassiveTex do TeXového zdroje a pak standardní TeXovou cestou (TeX -- DVI -- PS nebo TeX -- PDF s pomocí pdfTeXu)
- jak může vypadat celý postup zpracování, ukazuje návod J. Pavloviče k modulu xslt2 [http://www.fi.muni.cz/~xpavlov/xml/]

Výstup na malé displeje (WAP, PDA)

- pro mobilní telefony s WAP: v podstatě obdobné jako pro web, jazykem popisu WAP stránek je WML [http://www.wapforum.org/what/technical.htm]

- použijí se typicky opět XSLT transformace
- pro PDA: typicky nemají on-line spojení, připojují se "občas" přes stolní počítač zapojený do internetu
- proto potřebují systém na stažení, kompresi a off-line prohlížení webových stránek
- možným řešení je AvantGo.com [http://avantgo.com]: pro uživatele zdarma, pro poskytovatele obsahu je to komerční systém (omezené použití (málo uživatelů) zdarma)
- AvantGo systém je rozdělen na serverovou část (na výše uvedené adrese) a klientskou část rozdělenou na desktop (menší část) a PDA počítač (větší část)
- technicky je AvantGo schopno přizpůsobit a zobrazit "skoro normální HTML"
- problémy jsou pochopitelně s interaktivními aplikacemi (formuláře) -- částečně lze řešit tzv. Form Managerem
- bližší informace [http://www.avantgo.com/support/] k technice tvorby webů pro AvantGo

Hlasový výstup (VoiceXML)

VoiceXML je značkovací jazyk určený k popisu dialogu vedeného hlasovým rozhraním

Dave Raggett's Introduction to VoiceXML 2.0 [http://www.w3.org/Voice/Guide]

VoiceXML - Tutorials [http://www.voicexml.org/tutorials/]

VoiceXML Tutorial [http://cafe.bevocal.com/docs/tutorial/]

...další informace spolu se systémem Elvira (vyvíjeným na FI) najdete na domovské stránce [http://gin2.itek.norut.no/elvira/_elvira.php?p=introduction] projektu

Podrobněji k formátovacím objektům (XSL:FO)

Co a k čemu jsou XSL:FO

- *XSL:Formatting Objects* (XSL:FO) je standard pro platformově neutrální, v XML zapsaný popis (tiskového) formátu publikací
- dány standardem W3C - viz http://www.w3.org/Style/XSL/
- k transformaci z XML do XSL:FO se typicky využívá XSLT (proto původně v jedné specifikaci)
- dalším krokem bývá transformace XSL:FO do PDF/PS např. nástrojem Apache FOP [http://xml.apache.org/fop/index.html].
- obvykle kvalitnějšího výstupu dosáhneme použitím komerčního nástroje, který také podporuje širší

škálu formátovacích objektů. Viz např. RenderX XEP [http://www.renderx.com/].

Informační zdroje k XSL:FO

- dobrý úvodní článek What is XSL:FO [http://www.xml.com/pub/a/2002/03/20/xsl-fo.html] (Ken Hollman, XML.COM)
- kvalitní XSL FO Tutorial [http://www.renderx.com/tutorial.html] (firma RenderX)
- Stylesheet Tutorial, Sample Files of Formatting Objects and Sample Stylesheets [http://www.antennahouse.com/XSLsample/XSLsample.htm]

Rámce pro metadata popisující XML a jiné datové zdroje

Rámec RDF

RDF Model a RDf Schema jsou doporučeními W3C

Specifikace a další informace pracovní skupiny - http://www.w3.org/RDF

RDF Model

RDf je obecný mechanizmus pro specifikaci metadat

je použitelný k libovolných (i ne-digitálním) zdrojům

základem modelu jsou trojice:

- zdroj (resource) - např. `http://www.fi.muni.cz/~tomp/xml`
- vlastnost (property) - např. `popis`
- hodnota (value) - např. `Domovská stránka předmětu P138 na FI MU v Brně`

Trojice je možné znázornit

- graficky,
- jako trojice `(r, p, v)` nebo
- XML syntaxí

Blíže viz

- Dobrý úvodní článek na `xml.com`: What is RDF? [http://www.xml.com/pub/a/2001/01/24/rdf.html]
- RDF Tutoriál - Zvon RDF Tutorial [http://www.zvon.org/xxl/RDFTutorial/General/book.html]
- RDF Tutorial http://www710.univ-lyon1.fr/~champin/rdf-tutorial/node1.html
- Další RDF Tutorial (.ppt) [http://www.aifb.uni-karlsruhe.de/WBS/sst/Teaching/Intelligente%20System%20im%20WWW%20SS%202000/RDF-Tutorial.pdf]

RDF Schema

Specifikuje omezení na množiny vlastností, jejich definičních oborů a oborů hodnot

Modeluje se opět v RDF

RDF reprezentace užívaných metadatových schémat (Z39.50, Dublin Core atd.)

RDF je obecný rámec pro modelování metadat, pro konkrétní použití je obvykle nutné definovat *schéma* přípustných *vlastností*, jejich *domén* a množin (přípustných) *hodnot*.

Tím se vytvoří RDF reprezentace daného metadatového schématu.

Reprezentace může mít podobu *RDF Schematu*.

Dublin Core - příklad konkrétního metadatového schématu

Co je Dublin Core?

je generické metadatové schéma s univerzální použitelností

vznikl původně jako iniciativa knihovníků pro popis bibliografických informací

dnes univerzálně používán - např. pro metadatový popis informací ve veřejné správě (*e-Government*)

tvoří jej 15 základních elementů s rámcově definovanou sémantikou

elementy je možné rozšiřovat - rozkladem na (obvykle disjunktní) podmnožiny (vždy to musí být podmnožiny některého z původních elementů)

Jednoduchý (Simple) Dublin Core

"Jednoduchý" nebo "základní" Dublin Core (angl. Simple Dublin Core nebo Unqualified Dublin Core, dále jen "jednoduchý DC") představuje základní soubor patnácti prvků, který vyvinula a podporuje

- *Iniciativa pro metadata Dublin Core* (Dublin Core Metadata Initiative, DCMI, http://dublincore.org).
- přijat konsorciem IETF [http://ietf.org] jako tzv. *dokument RFC (Request For Comment) 2431*.
- Momentálně je aktuální verzí Dublin Core 1.1.

Dublin Core - elementy

```
Název       Jméno dané zdroji
Tvůrce      Entita primárně odpovědná za vytvoření obsahu zdroje
Předmět a klíčová slova     Téma obsahu zdroje
Popis       Vysvětlení obsahu zdroje
Vydavatel   Entita odpovědná za zpřístupnění zdroje
Přispěvatel Entita, která přispěla k vytvoření obsahu zdroje
Datum       Datum spojené s určitou událostí během existence zdroje
Typ zdroje  Povaha nebo druh obsahu zdroje
Formát      Fyzická nebo digitální reprezentace zdroje
Identifikátor zdroje    Jednoznačný odkaz na zdroj v rámci daného kontextu
Zdroj       Odkaz na zdroj, z něhož je popisovaný zdroj odvozen
Jazyk       Jazyk intelektuálního obsahu zdroje
Vztah       Odkaz na příbuzný zdroj
Pokrytí     Rozsah nebo záběr obsahu zdroje
Správa autorských práv Informace o právech vztahujících se k popisovanému zdroji
```

DC - příklad metadatového popisu

```
Název            Zelená kniha o elektronickém obchodu
Tvůrce           Úřad pro veřejné informační systémy, Úřad vlády
Předmět          Elektronický obchod, elektronický podpis,  bezpečnost, správa
Popis            Vládní návrh podpory elektronického obchodu v České republice
Datum vytvoření  2001-09-20
Datum zveřejnění 2001-10-17
Identifikátor    ISBN:?????
```

Kvalifikovaný Dublin Core

(Qualified Dublin Core) obsahuje stejný soubor prvků jako jednoduchý DC a doporučuje další upřesnění a omezení každého prvku.

Typicky se tak děje na základě formálního nebo de-facto mezinárodního standardu, např. může požadovat, aby prvek "jazyk" byl vyplněn v souladu se seznamem ISO pro jazyky (ISO 639).

Kódování DC v XML

DTD - http://dublincore.org/documents/2001/11/28/dcmes-xml/dcmes-xml-dtd.dtd [http://dublincore.org/documents/2001/11/28/dcmes-xml/dcmes-xml-dtd.dtd]

XML Schema - http://dublincore.org/documents/2001/11/28/dcmes-xml/dcmes-xml-xsd.xsd [http://dublincore.org/documents/2001/11/28/dcmes-xml/dcmes-xml-xsd.xsd]

RDF Schema - rdf/dc-rdf-schema-cz.rdf [/~tomp/xml/rdf/dc-rdf-schema-cz.rdf]

RDF Schema pro slovník typů (Type Vocabulary) - /~tomp/xml/rdf/dc-tv-rdf-schema-cz.rdf [/~tomp/xml/rdf/dc-tv-rdf-schema-cz.rdf]

Nástroje pro práci s RDF

Jena Java RDF API and toolkit http://www.hpl.hp.com/semweb/

The ICS-FORTH RDFSuite [http://139.91.183.30:9090/RDF/]

další viz http://www.w3.org/RDF [http://www.w3.org/RDF/]

Příklady praktického použití metadat - veřejná správa

Rámec pro metadata ISVS ČR

Kroky budování

- Přijmout doporučení **Dublin Core** a osvojit jej jako **Národní metadatový standard** (NMS).
- Rozšířit tento standard tak, aby vyhovoval potřebám veřejné správy jak pro snadné vyhledávání informací, tak pro správu informačních zdrojů.
- Vyvinout **Aplikační profil NMS**, který bude obsahovat předepsaná kódovací schémata a závazný výklad jednotlivých metadatových prvků.
- Připravit **Tezaurus veřejné správy**.

Adaptace Dublin Core pro potřeby veřejné správy

pro potřeby veřejné správy v zemích Evropské Unie, Austrálie, Kanady a Nového Zélandu je rozpracováván specifický *aplikační profil* Dublin Core.

Cílem MIReG je vytvořit metadatový rámec (metadata framework), příslušné referenční softwarové nástroje a soubor osvědčených postupů (best practice) pro implementaci rámce v jednotlivých zemích a sektorech. Přitom spolupracuje také s evropskou standardizační autoritou CEN, což dává předpoklad ce-

loevropského respektování vzniklého doporučení.

- proces zahájen na sérii pracovních seminářů **Managing information resources for e-government** (MIReG) a stal se součástí programu *Interchange of Data between Administrations (IDA)* Evropské Unie.
- Dalším partnerem při vytváření evropského metadatového rámce je též projekt **ParlML**, zaměřený na zpřístupňování informací Evropského parlamentu.
- Příslušná pracovní skupina připravuje doporučení **DC-Gov Application Profile**

Aplikační profil NMS

zahrnuje:

- **Upřesnění** (zjemnění, kvalifikaci, specializaci angl. element refinement) metadatových prvků, které přesněji určuje sémantiku daného prvku a tím jej rozděluje na jemněji (přesněji) určené podprvky - např. obecné datum lze kvalifikací rozdělit na menší části, a místo "datum" uvádět přesněji např. *"datum vytvoření", "datum zveřejnění", "datum platnosti", "nástupnické datum"*.
- Kvalifikovaný prvek lze však i nadále zpracovávat nástroji, které příslušné kvalifikaci "nerozumějí" - tyto nástroje potom chápou prvek jako by zůstal nekvalifikovaný (všeobecnější), tj. "datum zveřejnění" mohou chápat jako prosté "datum", čímž je sice část sémantiky ztracena, ale prvek může být stále užitečný např. pro vyhledávání.
- **Kódovací schémata** (též kvalifikace hodnoty, angl. encoding scheme nebo value qualification) specifikující formát, ve kterém bude uložena hodnota pro příslušný metadatový prvek, např. "datum" vždy bude uváděno ve formátu *rrrr-mm-dd* (rok-měsíc-den), což definuje standard ISO 8601.
- Kromě formátu může být kvalifikací hodnoty též např. specifikace *měrné jednotky*, v níž bude hodnota uváděna.

Ontologie

Co jsou ontologie?

prostředek jak popisovat znalosti

množina pojmů a konstruktů, jak je odvozovat, spojovat atd.

základní kategorie ontologií jsou

- **Classes** (general things) in the many domains of interest
- The **relationships** that can exist among things

- The **properties** (or **attributes**) those things may have

používá metadatové rámce (např. RDF), ale je

bohatší s přesnější sémantikou

předpokládá se vybudování obecného rámce pro tvorbu ontologií pro specifické domény

Aplikace ontologií (Use Cases)

- Web Portals
- Multimedia Collections
- Corporate Web Site Management
- Design documentation
- Intelligent agents
- Ubiquitous computing

Pracovní skupina při W3C [http://www.w3.org/2001/sw/WebOnt/]

XML Topic Maps

Další návrh pracovní skupině WebOnt - http://www.topicmaps.org/xtm/1.0 [http://www.topicmaps.org/xtm/1.0/]