XPath; validace a schémata XML dat

Obsah

XPath

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 i 2.0 jsou doporučeními W3C (W3C Recommendation) - http://www.w3.org/TR/xpath

XPath - aplikační oblasti

- Pokročilá navigace v XML datech

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c/>
 </b>
 <b>
   <c/>
 </b>
</a>
```

  - Vybrat třetí uzel b:

    ```
    //b[3]
    ```

  - Vybrat uzel b, který má potomka c:

```
//b[./c]
```

- Vybrat prázdný uzel b:

```
//b[count(./*)=0]
```

Příklad 1. //b[count(./*)=0]

XPath - aplikační oblasti

- Transformace (XSLT [http://www.w3.org/TR/xslt])
  - slouží k výběru uzlů, které se mají zpracovávat

Příklad 2. <xsl:value-of select="./c"/>

XPath - aplikační oblasti

- V "selekční části" XML dotazovacích jazyků (XQuery [http://www.w3.org/XML/Query/])
- V některých modelovacích jazycích (Schematron [http://www.schematron.com/], XML Schema [http://www.w3.org/XML/Schema])
- ...

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

Cesta (path) určuje (tj. „naviguje nás na“) lokaci v dokumentu. Cesty jsou konstruovány podobně jako cesty v systému souborů, tj. jako

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

XPath - 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 (jedn. číslo *axis*, množné *axes*) jsou množiny prvků dokumentu, vymezené (obvykle relativně) vůči *kontextu*.

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

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", atd. až 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`

Příklad osa child

Obrázek 1. //b/child::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c/>
 </b>
 <b>
   <c/>
 </b>
</a>
```

Příklad osa descendant

Příklad 3. //b/descendant::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c>
     <d/>
   </c>
 </b>
 <b>
   <c/>
 </b>
```

```
</a>
```

Příklad osa parent

Příklad 4. //d/parent::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c>
     <d/>
   </c>
 </b>
 <b>
   <c/>
 </b>
</a>
```

Příklad osa ancestor

Příklad 5. //d/ancestor::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c>
     <d/>
   </c>
 </b>
 <b>
   <c/>
 </b>
</a>
```

Příklad osa following-sibling

Příklad 6. //b/following-sibling::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c>
     <d/>
   </c>
 </b>
 <b>
   <c/>
 </b>
</a>
```

Příklad osa preceding-sibling

Příklad 7. //b/preceding-sibling::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c>
     <d/>
   </c>
 </b>
 <b>
   <c/>
 </b>
</a>
```

Příklad osa following

Příklad 8. /a/b/c/following::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c>
     <d/>
   </c>
   <e/>
 </b>
 <b>
   <c/>
 </b>
</a>
```

Příklad osa preceding

Příklad 9. /a/b/e/preceding::*

```
<?xml version="1.0"?>
<a>
 <b/>
 <b>
   <c>
     <d/>
   </c>
 </b>
 <b>
   <d/>
   <e/>
 </b>
</a>
```

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, tj. proti směru fyzického umístění v textové podobě dokumentu
- Specifikaci pozice **3** možno nahradit výrazem `position()=3`

XPath - výrazy

Určeny k použití v predikátech, k výpočtům, atd. Mohou obsahovat XPath funkce.

Výrazy mohou být:

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

XPath - zkrácená notace - Příklady

- `para` *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` *vybere všechny element para, jež jsou následníky chapter*
- `//para` *vybere všechny elementy para z dokumentu*
- `//olist/item` *vybere všechny elementy item, které mají za rodiče olist. vybere kontextový uzel*
- `.//para` *vybere všechny elementy-potomky kontextového uzlu, které nesou značku para*

- `..` *vybere rodičovský uzel od kontextového*
- `../@lang` *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 `..`

Poznámka

Pro přehlednost někdy delší formu zachováváme: nebraňme se jí za každou cenu!

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

- Již finální specifikace - 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://www.saxonica.com/, kde nalezneme i XPath/XSLT/XQuery procesor *Saxon*.

XPath 2.0 - příklady

- Řetězcové funkce [http://www.fi.muni.cz/~tomp/xml03/xpath20/string.html]
- Numerické funkce [http://www.fi.muni.cz/~tomp/xml03/xpath20/numeric.html]
- Funkce nad sekvencemi [http://www.fi.muni.cz/~tomp/xml03/xpath20/sequence.html]

- Booleovské funkce [http://www.fi.muni.cz/~tomp/xml03/xpath20/boolean.html]

Specifikace a validita XML

Aktuální specifikace XML

- Původní specifikace (W3C Recommendation) XML 1.0 na W3C: http://www.w3.org/XML/
- 4th Edition (aktualizace, opravy, ne změny) na Extensible Markup Language (XML) 1.0 (Fourth Edition) [http://www.w3.org/TR/2006/REC-xml-20060816/]
- výborná komentovaná verze téhož na XML.COM (Annotated XML): http://www.xml.com/pub/a/axml/axmlintro.html
- XML 1.1 (Second Edition) [http://www.w3.org/TR/2006/REC-xml11-20060816/] - změny indukované zavedením *UNICODE 3*, lepší možnosti *normalizace*, upřesnění postupu manipulace se znaky *ukončení řádku*. XML 1.1 není už vázaný na konkrétní verzi UNICODE, ale vždy na verzi poslední.

Jakou verzi použít?

Jakou verzi specifikace bychom měli v nových aplikacích používat?

Odpověď dává W3C XML Core Working Group [http://www.w3.org/XML/Core/#Publications]:

- nepíšeme-li parser, ale aplikaci, která generuje nebo vytváří XML (editor), používejme XML 1.0 (zpětná kompatibilita)
- nové parsery by měly umět XML 1.1

Validita XML dokumentů

- Opakování: každý XML dokument MUSÍ být správně utvořený (*well formed*)
- Nové: XML dokument může být platný (*valid*) dokument:

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

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

  (moderněji) - soulad s XML Schematem, případně jinými schématy (RelaxNG, Schematron).

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 elementů

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