XML schémata

Úvod do XML pro vývojáře

Jirka Kosek

Poslední modifikace: $Date: 2020/03/18 18:51:52 $

Obsah

Úvod

Proč potřebujeme schéma dokumentů XML

- XML umožňuje vytvářet dokumenty s libovolně pojmenovanými a vnořenými elementy
- příliš volnosti škodí
  - formáty pro výměnu dat
- schéma XML dokumentu umožňuje definovat
  - elementy a atributy použitelné v dokumentu
  - přípustné možnosti kombinování jednotlivých elementů
  - datový typ pro obsah elementu/atributu
  - další integritní omezení
- schéma XML dokumentu plní podobnou funkci jako schéma relační databáze

Přínosy použití schématu

- schéma je formální definice jazyka (výměnného formátu) založeného na XML
- dokument XML můžeme kdykoliv během jeho životního cyklu validovat
- validace = ověření shody dokumentu se schématem
- validace výrazně zjednodušuje kontroly vstupu na úrovni aplikace
- komfortnější zadávání dat do editorů XML
- snazší programová manipulace s dokumenty XML (PSVI, data-binding)
- generování dokumentace
- informace ze schématu potřebují některé další navazující XML jazyky – například dotazovací jazyk XQuery

Jazyky pro popis schématu

- DTD
  - nejstarší, vychází ještě z SGML, přímo součást specifikace XML
  - nepodporuje jmenné prostory a datové typy
- W3C XML Schema
  - podpora jmenných prostorů, datových typů
  - poměrně složitá specifikace
  - široká podpora komerčních firem: MS, IBM, Oracle, Sun, …
- Relax NG
  - nový a elegantní jazyk pro popis schématu
  - podpora zatím spíše jen ve světě OSS
  - standardizováno v rámci OASIS a ISO
- Schematron
  - sada XPath výrazů, které musí dokument splňovat

Jaký jazyk používat?

- nepotřebujeme jmenné prostory a datové typy ⇒ DTD
- potřebujeme jmenné prostory a datové typy
  - nemusíme používat nástroje od MS, IBM, Oracle, Sun ⇒ Relax NG
  - musíme používat nástroje od MS, IBM, Oracle, Sun ⇒ W3C XML Schema
- různé jazyky pokrývají různé potřeby
- projekt DSDL (Document Schema Definition Languages)
  - standardní prostředí pro validaci oproti několika schématům
  - vzniká na půdě ISO

Ukázky nejpoužívanějších jazyků pro popis schématu dokumentu

Relax NG

Kompaktní syntaxe

-

DTD

DTD – Definice typu dokumentu

- DTD je jazyk pro definici nových jazyků, které základní syntaxí vycházejí z XML
- DTD umožňuje definovat:
  - elementy použitelné v dokumentu
  - jejich přípustné vztahy
  - atributy použitelné u jednotlivých elementů a jejich typ
- DTD má v XML podobný význam jeho schéma v relačních databázích
- dokument vyhovující DTD = validní dokument
  - lze kontrolovat pomocí validujícího parseru
- v současné době existuje mnoho DTD, které definují jazyky používané v mnoha oblastech
  - WML, XHTML, DocBook, SVG, MathML, RSS, …

Ukázka DTD a dokumentu

Příklad 1. DTD – `dtd/dokument.dtd`

Příklad 2. Dokument vyhovující DTD – `dtd/dokument.xml`

Deklarace elementů

- `<!ELEMENT` *`název model obsahu`*`>`
- model obsahu:
  - sekvence – (a,b,c,d)
  - alternativa – (a|b|c)
  - `ANY`
  - `EMPTY`
  - `(#PCDATA)`
- opakování
  - `element` – právě jednou
  - `element?` – 0 nebo 1
  - `element+` – alespoň jednou
  - `element*` – libovolný počet
- lze navzájem kombinovat pomocí závorek

```
(nazev, (autor|editor)?, p*, (nadpis,p+)*)
```

- každý element použitý v modelu obsahu musí být samostatně deklarován
- smíšený obsah
  - element obsahuje jak text, tak další elementy na stejné úrovni
  - `<!ELEMENT p (#PCDATA|em|strong)*>`
  - neumožňuje kontrolovat počet a pořadí podelementů

Deklarace atributů

- <!ATTLIST *element*
  název typ default. hodnota>

- typy
  - CDATA, NMTOKEN, NMTOKENS, ID, IDREF, IDREFS, ENTITY, ENTITIES, výčet
- default. hodnota:
  - "hodnota"
  - #REQUIRED
  - #IMPLIED
  - #FIXED "hodnota"

Příklad 3. Ukázka deklarace atributů a jejich použití v dokumentu

```
<!ATTLIST odstavec
          id        ID                      #REQUIRED
          zarovnání (vlevo|nastřed|vpravo) #IMPLIED
          autor     CDATA                   "nobody">


<odstavec id="n012"
          autor="Jan Novák">...</odstavec>

<odstavec id="n3.4.1"
          zarovnání="nastřed">...</odstavec>
```

Připojení DTD k dokumentu

- DTD se přidává pomocí deklarace typu dokumentu (DOCTYPE)
- DTD v externím souboru (externí podmnožina)

```
<!DOCTYPE dokument SYSTEM "dok.dtd">
<dokument>
  ...
</dokument>
```

  jedno DTD lze používat opakovaně
- DTD přímo v dokumentu (interní podmnožina)

```
<!DOCTYPE dokument [
<!ELEMENT ...>
   ...
<!ATTLIST ...>
   ...
]>
<dokument>
  ...
</dokument>
```

  nepraktické, protože nemůžeme DTD využít opakovaně
- lze kombinovat interní a externí DTD

```
<!DOCTYPE dokument SYSTEM "dok.dtd" [
<!ELEMENT ...>
   ...
<!ATTLIST ...>
   ...
]>
<dokument>
  ...
</dokument>
```

  lokální deklarace mají přednost před externími → pomocí parametrů lze upravit DTD

Parsery

- MSXML
  - standardní komponenta IE a dalších MS aplikací
  - velmi rychlý
  - lze využívat ze všech jazyků přes COM rozhraní
  - spuštění např. pomocí WSH skriptu

    ```
    msxml dokument.xml
    ```

  - POZOR – IE validaci neprovádí (lze doplnit plug-in[1])
- Xerces[2]
  - open-source implementace v Javě i C++
  - spuštění ve validačním režimu

    ```
    xerces -v dokument.xml
    ```

- xmllint (součást libxml2)[3]
  - open source implementace v C dostupná pro Unix i Windows
  - velmi rychlý
  - kontrola well-formdness

    ```
    xmllint --noout dokument.xml
    ```

  - validace

    ```
    xmllint --noout --valid dokument.xml
    ```

1 http://www.microsoft.com/downloads/details.aspx?FamilyId=D23C1D2C-1571-4D61-BDA8-ADF9F6849DF9&displaylang=en
2 http://xml.apache.org/xerces2-j/
3 http://xmlsoft.org/

Cvičení

DTD a validace

1. vytvořte si DTD soubor popisující dokument objednávky v souboru `objednavka.xml`
2. připojte vytvořený DTD soubor k dokumentu XML pomocí `!DOCTYPE`
3. zkuste dokument zvalidovat
4. upravujte DTD tak dlouho, dokud dokument neprojde validací

Parametrické entity

- podobné použití jako textové entity, ale lze je použít pouze v rámci DTD
- interní parametrické entity

```
<!ENTITY % všude "id    ID    #REQUIRED
                  autor CDATA 'nobody'">

<!ATTLIST nadpis
          %všude;>
<!ATTLIST odstavec
          %všude;
          zarovnání (vlevo|vpravo) "vlevo">
```

  použití: společné atributy, opakující se modely obsahu

- externí parametrické entity

```
<!ENTITY % ISOpub SYSTEM "iso-pub.ent">
%ISOpub;
```

  použití: modularizace DTD, opakované využití standardních entit v různých DTD

- standardizované sady entit – http://www.w3.org/2003/entities/

Příklad 4. Načtení deklarací pomocí parametrické entity – `dtd/parametricke-entity.xml`

Příklad 5. Soubor `isopub.ent`

```
…
<!ENTITY emsp " ">
<!ENTITY ensp " ">
<!ENTITY numsp " ">
<!ENTITY puncsp " ">
<!ENTITY thinsp " ">
<!ENTITY hairsp " ">
<!ENTITY mdash "&#x2014;">
<!ENTITY ndash "&#x2013;">
 …
```

Základy W3C XML Schema

XML schéma se zapisuje v XML

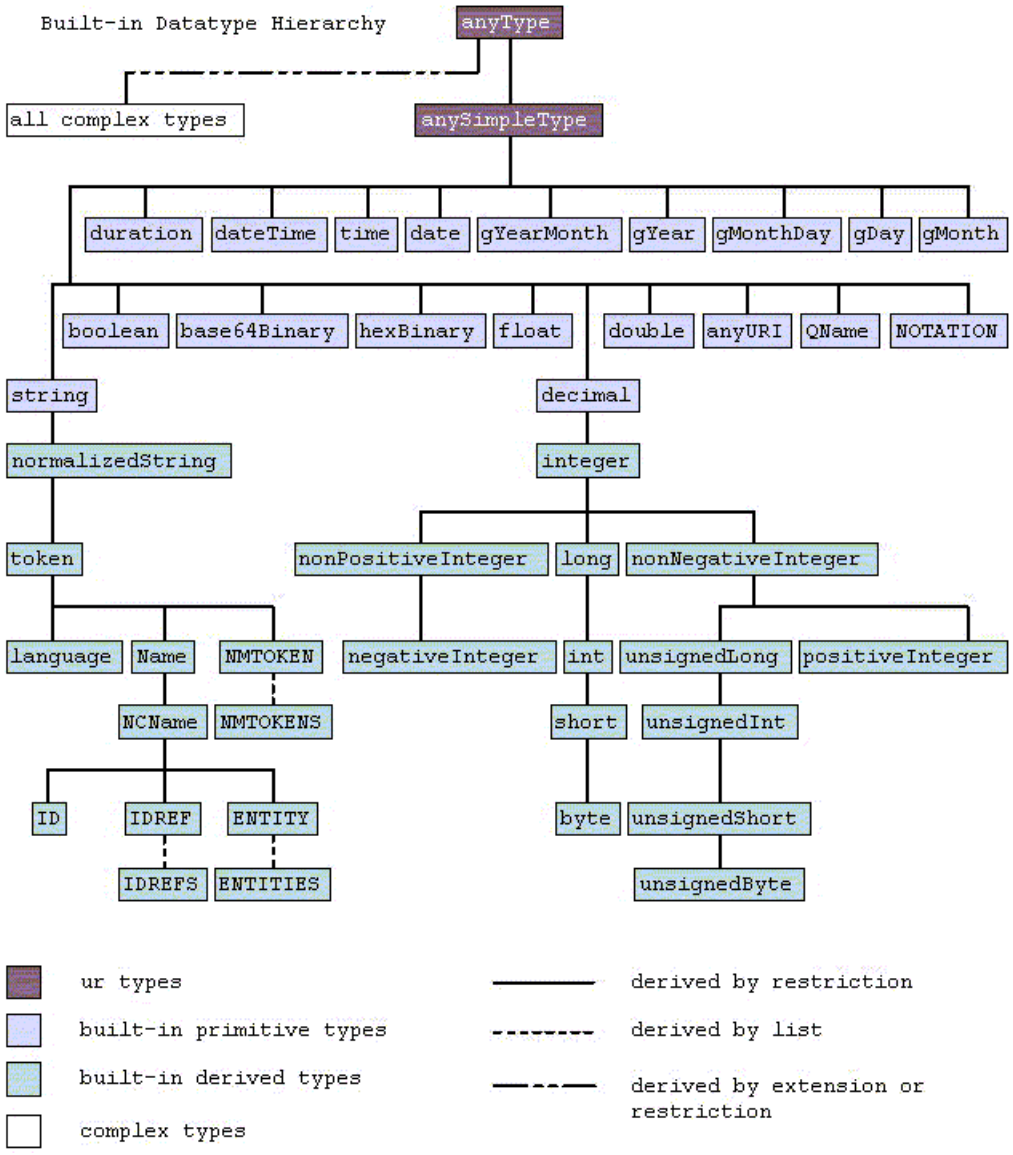
```
<zamestnanec id="101">
  <jmeno>Jan</jmeno>
  <prijmeni>Novák</prijmeni>
  <plat>25000</plat>
  <narozen>1965-12-24</narozen>
</zamestnanec>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="zamestnanec">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="jmeno" type="xs:string"/>
        <xs:element name="prijmeni" type="xs:string"/>
        <xs:element name="plat" type="xs:decimal"/>
        <xs:element name="narozen" type="xs:date"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

- všechny elementy a datové typy patří do jmenného prostoru XML schémat (`http://www.w3.org/2001/XMLSchema`)
- pro každý element/atribut musíme určit datový typ

Datové typy

- lze použít pro obsah atributů i elementů
- komplexní × jednoduché typy
  - komplexní – obsahují další elementy a atributy
  - jednoduché – obsahují pouze jednu hodnotu (řetězec, číslo apod.)
- od existujících typů lze odvozovat typy vlastní
- jednoduché typy:
  - textový řetězec, celá/desetinná čísla a jejich varianty
  - binární data, logická hodnota
  - datum, čas, časový interval
  - typy převzaté z DTD pro snazší přechod
- typy lze rozšiřovat/omezovat – obdoba integritních omezení
- lze definovat referenční integritu

Přehled zabudovaných typů



Samodokumentující formát

- přímo součástí schématu může být dokumentace
- pomocí XSLT lze pak generovat přehlednou dokumentaci schématu v HTML (http://titanium.dstc.edu.au/xml/xs3p/)

```
<xs:element name="zamestnanec">
  <xs:annotation>
    <xs:documentation>Element slouží pro uchování důležitých
      údajů o zaměstnanci.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:annotation>
      <xs:documentation xmlns="http://www.w3.org/1999/xhtml">
        <p>Dokumentace
          může být klidně v <a ▶
href="http://www.w3.org/TR/xhtml1/">XHTML</a>.</p>
        <ul>
          <li>a používat</li>
          <li>třeba</li>
          <li>seznamy</li>
        </ul>
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="jmeno" type="xs:string"/>
      <xs:element name="prijmeni" type="xs:string"/>
      <xs:element name="plat" type="xs:decimal">
 <xs:annotation>
   <xs:documentation>Superhrubá mzda v Kč</xs:documentation>
 </xs:annotation>
      </xs:element>
      <xs:element name="narozen" type="xs:date">
 <xs:annotation>
   <xs:documentation>Datum narození ve formátu ▶
RRRR-MM-DD</xs:documentation>
 </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Jednoduché typy

Jednoduché datové typy

- vlastní typy lze odvodit z již definovaných typů pomocí restrikce, vytvořením seznamu nebo sjednocením typů
- u většiny typů lze definovat různá integritní omezení:
  - řetězce – length, minLength, maxLength, pattern, enumeration, whiteSpace
  - číselné typy – maxInclusive, maxExclusive, minInclusive, minExclusive, totalDigits, fractionDigits, pattern, enumeration
  - binární data – length, minLength, maxLength, pattern, enumeration, whiteSpace

Vytváření vlastních typů

Vytvoření a použití typu pro měnové údaje

- maximální částka 1 milión
- přesnost na haléře

```
<cenaVýrobku>23.50</cenaVýrobku>

<!-- Vytvoření uživatelského datového typu -->
<xs:simpleType name="částka">
  <xs:restriction base="xs:decimal">
    <xs:minInclusive value="0"/>
    <xs:maxExclusive value="1000000"/>
    <xs:fractionDigits value="2"/>
  </xs:restriction>
</xs:simpleType>

<!-- Definice elementu využívající nový datový typ -->
<xs:element name="cenaVýrobku" type="částka"/>
```

Vytváření vlastních typů

Vytvoření typu pro kód měny, deklarace atributu

```
<cena měna="USD">23.50</cena>

<!-- Definice typu výčtem -->
<xs:simpleType name="kódMěny">
  <xs:restriction base="xs:string">
    <xs:enumeration value="CZK"/>
    <xs:enumeration value="EUR"/>
    <xs:enumeration value="USD"/>
  </xs:restriction>
</xs:simpleType>

<!-- Element má jednoduchý obsah (číslo) rozšířený o atribut -->
<xs:element name="cena">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="částka">
        <xs:attribute name="měna" type="kódMěny"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Lexikální a hodnotový prostor

- většina integritních omezení pracuje nad prostorem hodnot
  - různé hodnoty z XML dokumentu se převedenou na skutečnou hodnotu
  - `3.5` a `3.500` se chápe stejně, pokud to jsou čísla
  - `3.5` a `3.500` se chápe odlišně, pokud jsou v elementech typu `xs:string`
- nad lexikálním prostorem pracují vzory (pattern)
  - lexikální prostor je tvořen znaky zapsanými přímo v dokumentu XML s následně upravenými bílými znaky
    - všechny bílé znaky (konec řádky, tabulátor) jsou nahrazeny mezerou
    - více mezer je nahrazeno jedinou mezerou, mezery na začátku a na konci jsou odstraněny
    - pravidla se neaplikují na typy `xs:string` a `xs:normalizedString`
  - nejčastěji se nad ním definuje omezení pomocí regulárního výrazu
  - regulární výrazy používají perlovou syntaxi
  - příklad – DIČ: `\d{3}-\d{10}`

Komplexní typy

Komplexní typy

- modelování elementů, které obsahují další elementy nebo atributy
- lze určit pořadí elementů, jejich opakování, volitelnost atd.

```
<xs:element name="kniha">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nazev" type="xs:string"/>
      <xs:element name="autor" type="xs:string"
                  minOccurs="0" maxOccurs="unbounded"/>
      <xs:choice minOccurs="1" maxOccurs="unbounded">
        <xs:element name="odstavec" type="xs:string"/>
        <xs:element name="obrazek" type="xs:base64Binary"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- sekvence za sebou jdoucích elementů – `<xs:sequence>`
- výběr jednoho z elementů – `<xs:choice>`
- nezáleží na pořadí elementů – `<xs:all>`
- vytvoření skupiny – `<xs:group>`
- lze vzájemně kombinovat
- smíšený obsah (mezi elementy se může objevit text) – `<xs:complexType mixed="true">`

Sekvence elementů

xs:sequence

- všechny elementy se musí objevit v zadaném pořadí
- počet opakování elementu lze určit pomocí `maxOccurs` a `minOccurs`
- implicitní hodnoty: `maxOccurs=1`, `minOccurs=1`
- pro nekonečno se používá hodnota `unbounded`

```
<článek>
  <nadpis>Ukázka</nadpis>
  <autor>Pepa</autor>
  <odstavec>...</odstavec>
  <odstavec>...</odstavec>
</článek>

<xs:element name="článek">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nadpis" type="xs:string"/>
      <xs:element name="autor" type="xs:string"
                  minOccurs="0"/>
      <xs:element name="odstavec" type="xs:string"
                  maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Výběr jednoho z elementů

xs:choice

- může se vyskytnout pouze jeden z uvedených podelementů

```
<osoby>
  <osoba>
    <jméno>Pepa Tuzemec</jméno>
    <RČ>681203/0123</RČ>
  </osoba>
  <osoba>
    <jméno>Pepa Cizinec</jméno>
    <pas>1234567</pas>
  </osoba>
  <osoba>
    <jméno>Pepa Rozvědčík</jméno>
    <SSN>987654321</SSN>
  </osoba>
</osoby>

<xs:element name="osoby">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="osoba" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="jméno" type="xs:string"/>
            <xs:choice>
               <xs:element name="RČ" type="xs:string"/>
               <xs:element name="pas" type="xs:string"/>
               <xs:element name="SSN" type="xs:string"/>
            </xs:choice>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- chybou je uvést více než jednu variantu:

```
  <osoba>
    <jméno>Pepa Pochybil</jméno>
```

Výběr jednoho z elementů (Pokračování)

```
    <pas>1234567</pas>
    <RČ>681203/0123</RČ>
  </osoba>
```

Elementy v libovolném pořadí

xs:all

- podobně jako `xs:sequence`, ale nezáleží na pořadí výskytu
- počet opakování může být pouze 0 nebo 1

```
<osoba>
  <jméno>Jan</jméno>
  <příjmení>Novák</příjmení>
</osoba>
<osoba>
  <příjmení>Novák</příjmení>
  <jméno>Jan</jméno>
</osoba>
<osoba>
  <titul>Ing.</titul>
  <jméno>Jan</jméno>
  <příjmení>Novák</příjmení>
</osoba>
<osoba>
  <jméno>Jan</jméno>
  <příjmení>Novák</příjmení>
  <titul>CSc.</titul>
</osoba>

<xs:element name="osoba">
  <xs:complexType>
    <xs:all>
      <xs:element name="jméno" type="xs:string"/>
      <xs:element name="příjmení" type="xs:string"/>
      <xs:element name="titul" type="xs:string"
                  minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Prázdný element

- nemá žádný obsah – text, podelementy, …
- může obsahovat pouze atributy

```
<img src="obrazek.png"/>

<!-- Zkrácená syntaxe -->
<xs:element name="img">
  <xs:complexType>
    <xs:attribute name="src" type="xs:anyURI"/>
  </xs:complexType>
</xs:element>

<!-- Ekvivalent v plné syntaxi -->
<xs:element name="img">
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base="xs:anyType">
        <xs:attribute name="src" type="xs:anyURI"/>
      </xs:restriction>
    <xs:complexContent>
  </xs:complexType>
</xs:element>
```

Smíšený obsah

- mezi elementy se může objevit text
- funguje trochu odlišně než smíšený obsah v DTD
- po vynechání textu musí podelementy vyhovět definici komplexního typu

```
<odstavec>Odstavce typicky obsahují <pojem>smíšený
obsah</pojem>. Text se může střídat
s <odkaz url="http://www.kosek.cz">odkazy</odkaz>
a dalšími <pojem>elementy</pojem>.</odstavec>

<odstavec>Odstavec může obsahovat i jen text.</odstavec>

<odstavec><pojem>Nebo jen element.</pojem></odstavec>
```

```
<xs:element name="odstavec">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="pojem" type="xs:string"/>
      <xs:element name="odkaz">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="url" type="xs:anyURI"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Atributy

- jsou součástí komplexních typů
- mohou obsahovat jen jednoduché typy
- `default` – standardní hodnota atributu, doplní se v případě, že atribut chybí (není dobré používat)
- `use` – povinnost atributu
  - `optional` – nepovinný
  - `required` – povinný
  - `prohibited` – zakázaný (může se využít při odvozování typů)

```
<xs:element name="img">
  <xs:complexType>
    <xs:attribute name="src" type="xs:anyURI"
                  use="required"/>
    <xs:attribute name="alt" type="xs:string"
                  use="required"/>
    <xs:attribute name="title" type="xs:string"
                  default="Bez titulku"/>
  </xs:complexType>
</xs:element>
```

- ekvivalentní zápisy:

```
<img src="logo.gif" alt="Logo firmy"/>

<img src="logo.gif" alt="Logo firmy"
     title="Bez titulku"/>
```

- chybné zápisy:

```
<img alt="Logo" title="Logo naší firmy"/>

<img src="logo.gif"/>
```

Jmenné prostory

Globální deklarace

- globální deklarace – jsou uvedené přímo pod `xs:schema`
- globální deklarace lze využívat z jiných schémat (`xs:import`, `xs:include`)
- jmenný prostor pro globální elementy/atributy se určuje pomocí `targetNamespace`

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:cz-kosek:schemas:zamestnanci:v1.0">
```

- standardně do cílového jmenného prostoru patří jen globálně deklarované elementy

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:x-kosek:schemas:pokus"
           xmlns="urn:x-kosek:schemas:pokus">
  <xs:element name="a">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="b" type="xs:string"/>
        <xs:element name="c" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<a xmlns="urn:x-kosek:schemas:pokus">
  <b xmlns="">foo</b>
  <c xmlns="">bar</c>
</a>

<p:a xmlns:p="urn:x-kosek:schemas:pokus">
  <b>foo</b>
  <c>bar</c>
</p:a>
```

Lokální deklarace

- lokální deklarace jsou uvnitř globálních
- nelze je znovuvyužívat z jiných schémat
- u typů hovoříme o tzv. anonymních typech – nejsou pojmenované a nejde se na ně odvolat
- u lokálních deklarací elementů/atributů můžeme pomocí atributu `form` určit, zda mají patřit do cílového jmenného prostoru

```
<xs:element name="b" form="qualified" type="xs:string"/>
```

- jde nastavit i globálně pro celé schéma – `elementFormDefault`, `attributeFormDefault`

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:x-kosek:schemas:pokus"
           xmlns="urn:x-kosek:schemas:pokus"
           elementFormDefault="qualified">
  <xs:element name="a">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="b" type="xs:string"/>
        <xs:element name="c" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<a xmlns="urn:x-kosek:schemas:pokus">
  <b>foo</b>
  <c>bar</c>
</a>

<p:a xmlns:p="urn:x-kosek:schemas:pokus">
  <p:b>foo</p:b>
  <p:c>bar</p:c>
</p:a>
```

Validace

Připojení schéma k dokumentu

Nepoužíváme vlastní jmenný prostor

- před validací musíme parseru sdělit, kde pro dokument najde jeho schéma
- možnosti určení schématu:
  - ruční výběr schématu
  - automatická připojení schématu podle jmenného prostoru/jména kořenového elementu/…
  - odkaz na schéma je uveden přímo v dokumentu XML

    lze využit speciální atributy

    některé novější nástroje podporují instrukci `<?xml-model ?>`

Příklad 6. XML dokument – `xsd/faktura.xml`

```
<faktura xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="faktura.xsd"

  …

</faktura>
```

Příklad 7. XML schéma – `xsd/faktura.xsd`

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

 <xs:element name="faktura">

 ...
```

Připojení schéma k dokumentu

Používáme vlastní jmenný prostor

- pro každý jmenný prostor můžeme určit, kde se pro něj najde schéma
- atribut `xsi:schemaLocation` obsahuje seznam uspořádaných dvojic jmenný prostor a umístění schématu

Příklad 8. XML dokument – `xsd/fakturans.xml`

```
<faktura xmlns="urn:x-kosek:schemas:faktura:1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:x-kosek:schemas:faktura:1.0 fakturans.xsd">

  …

</faktura>
```

nebo bez deklarace implicitního jmenného prostoru

```
<f:faktura xmlns:f="urn:x-kosek:schemas:faktura:1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:x-kosek:schemas:faktura:1.0 fakturans.xsd">

  ...

</f:faktura>
```

Příklad 9. XML schéma – `xsd/fakturans.xsd`

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="urn:x-kosek:schemas:faktura:1.0"
            xmlns="urn:x-kosek:schemas:faktura:1.0"
            elementFormDefault="qualified">

  <xsd:element name="faktura">

 …
```

Podpora schémat v parserech

- Xerces[4]
  - součást projektu Apache
  - open source projekt
  - platforma: Java
  - spuštění ve validačním režimu

```
xerces -v -s dokument.xml
```

- MSXML4, System.Xml
  - autor: Microsoft
  - platforma: Win32, .NET
- XSV[5]
  - autor: University of Edinburgh
  - platforma: Python
- xmllint (součást libxml2)[6]
  - open source implementace v C dostupná pro Unix i Windows
  - neúplná podpora datových typů
  - `xmllint --schema schema.xsd --noout dokument.xml`
- … a mnoho dalších

[4] http://xml.apache.org/
[5] http://www.w3.org/2001/03/webdata/xsv
[6] http://xmlsoft.org/

Přístupy k návrhu schématu

Struktura schématu

- schémata nabízejí mnoho různých možností pro popsání stejné struktury XML
- můžeme je libovolně kombinovat
- v praxi je však dobré držet se jednotného a konzistentního přístupu k tvorbě schématu

Příklad 10. Ukázkový dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<zamestnanec>
  <jmeno>Jan</jmeno>
  <prijmeni>Novák</prijmeni>
  <adresa>
    <ulice>Dlouhá 2</ulice>
    <město>Praha 1</město>
    <psč>110 00</psč>
  </adresa>
  <plat>34500</plat>
</zamestnanec>
```

Matrjóška

- globální je jen jeden element
- špatné možnosti znovupoužití
- schéma je krátké a kompaktní

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="zamestnanec">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="jmeno" type="xs:string"/>
        <xs:element name="prijmeni" type="xs:string"/>
        <xs:element name="adresa">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ulice" type="xs:string"/>
              <xs:element name="město" type="xs:string"/>
              <xs:element name="psč" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="plat" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Salámová kolečka

- všechny elementy jsou globální
- dohromady se vše složí pomocí odkazů
- dokument může začínat libovolným elementem
- všechny elementy lze znovupoužívat
- jeden element nemůže mít dva různé modely obsahu podle kontextu

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="jmeno" type="xs:string"/>
  <xs:element name="prijmeni" type="xs:string"/>
  <xs:element name="ulice" type="xs:string"/>
  <xs:element name="město" type="xs:string"/>
  <xs:element name="psč" type="xs:string"/>
  <xs:element name="plat" type="xs:decimal"/>

  <xs:element name="adresa">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ulice"/>
        <xs:element ref="město"/>
        <xs:element ref="psč"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="zamestnanec">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="jmeno"/>
        <xs:element ref="prijmeni"/>
        <xs:element ref="adresa"/>
        <xs:element ref="plat"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Metoda slepého Benátčana

- pro všechny elementy se definují typy
- elementy jsou definovány lokálně pomocí těchto typů
- spojuje výhody předchozích dvou přístupů
- nejupovídanější a nejpracnější
- správný název je „žaluziová metoda“ (Venetian Blind)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="jmenoType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="15"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="prijmeniType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="20"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="uliceType">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <xs:simpleType name="městoType">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <xs:simpleType name="pscType">
    <xs:restriction base="xs:token">
      <xs:pattern value="[0-9]{3} [0-9]{2}"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="platType">
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0"/>
    </xs:restriction>
```

Metoda slepého Benátčana (Pokračování)

```
  </xs:simpleType>

  <xs:complexType name="adresaType">
    <xs:sequence>
      <xs:element name="ulice" type="uliceType"/>
      <xs:element name="město" type="městoType"/>
      <xs:element name="psč" type="pščType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="zamestnanecType">
    <xs:sequence>
      <xs:element name="jmeno" type="jmenoType"/>
      <xs:element name="prijmeni" type="prijmeniType"/>
      <xs:element name="adresa" type="adresaType"></xs:element>
      <xs:element name="plat" type="platType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="zamestnanec" type="zamestnanecType"/>

</xs:schema>
```

Cvičení

Validace oproti XML schématu

1. vytvořte schéma popisující dokument objednávky `cviceni/objednavka.xml`
2. ve schématu se snažte používat datové typy a integritní omezení
3. připojte schéma k dokumentu pomocí atributu `noNamespaceSchemaLocation` ve jmenném prostoru `http://www.w3.org/2001/XMLSchema-instance`
4. upravujte schéma tak dlouho, dokud dokument nejde zvalidovat

Pokročilé vlastnosti

Práce s prázdnými hodnotami (NULL)

- elementy, které mohou být prázdné, musíme takto definovat

```
<xs:element name="cena" nillable="true"
            type="xs:decimal"/>
```

- nevalidní zápis, nevyhovuje definici datového typu

```
<cena></cena>
<cena/>
```

- cena má hodnotu „NULL“

```
<cena xsi:nil="true"></cena>
<cena xsi:nil="true"/>
```

- nepřípustné použití

```
<cena xsi:nil="true">123</cena>
```

- nelze použít pro atributy, pouze pro elementy

Zajištění jedinečnosti hodnot

- nad libovolnou množinou elementů a atributů lze definovat unikátní klíč
- v jednom dokumentu XML můžeme mít několik klíčů
- definice klíče pomocí XPath výrazů
- typy klíčů
  - `xs:unique` – zajistí unikátnost hodnot, ale hodnoty mohou chybět
  - `xs:key` – zajistí unikátnost hodnot, hodnoty musí být vždy uvedeny
  - `xs:keyref` – cizí klíč

Ukázka unikátního klíče

Příklad 11. Osobní číslo je jedinečné

```
<zamestnanci>
  <zamestnanec oc="1164">
    <jmeno>Procházka Karel</jmeno>
    <sef>2021</sef>
  </zamestnanec>
  <zamestnanec oc="1168">
    <jmeno>Novotná Alena</jmeno>
    <sef>2021</sef>
  </zamestnanec>
  <zamestnanec oc="1230">
    <jmeno>Klíma Josef</jmeno>
    <sef>1168</sef>
  </zamestnanec>
  <zamestnanec oc="1564">
    <jmeno>Pinkas Josef</jmeno>
    <sef>2021</sef>
  </zamestnanec>
  <zamestnanec oc="2021">
    <jmeno>Kládová Adéla</jmeno>
  </zamestnanec>
</zamestnanci>

...
<xs:element name="zamestnanci">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="zamestnanec" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:unique name="oc_je_unikatni">
    <xs:selector xpath="zamestnanec"/>
    <xs:field xpath="@oc"/>
  </xs:unique>
</xs:element>
...
```

Referenční integrita

- definuje se jako cizí klíč, který musí ukazovat na nějaký existující klíč
- cizí klíč musí být definován na stejné nebo vyšší úrovni než klíč

Příklad 12. Šéf každého zaměstnance existuje

```
<xs:element name="zamestnanci">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="zamestnanec" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:key name="osobni_cislo">
    <xs:selector xpath="zamestnanec"/>
    <xs:field xpath="@oc"/>
  </xs:key>

  <xs:keyref name="sef_je_existujici_oc" refer="osobni_cislo">
    <xs:selector xpath="zamestnanec"/>
    <xs:field xpath="sef"/>
  </xs:keyref>

</xs:element>
```

- xs:keyref může ukazovat i na xs:unique

Objektově orientované rysy

- možnost odvozování (dědění) nových typů od již existujících
- substituční skupiny (podtřídy)
- abstraktní datové typy (nejde je použít v instanci)
- můžeme zablokovat další dědění typů

„Best practices“ pro návrh

Jmenné konvence

- každé schéma by mělo mít definováno vlastní cílový jmenný prostor (`targetNamespace`)
- všechny elementy by měly patřit do tohoto jmenného prostoru (`elementFormDefault="qualified"`)
- konzistentní a srozumitelné názvy elementů/atributů
  - moc krátké jsou nesrozumitelné, příliš dlouhé se špatně píší
  - `<cenaVýrobku kódMěny="USD">5.99</cenaVýrobku>`

Jemnost značkování

- značkovat má cenu jen ty údaje, které v dohledné budoucnosti využijeme
- snaha o příliš „dokonalé“ značkování většinou komplikuje a prodražuje získávání vstupních dat
- pro většinu aplikací málo značkování:

```
<adresa>Nábřeží Edvarda Beneše 4
Praha 1
118 01</adresa>
```

- pro většinu aplikací zbytečně mnoho značkování:

```
<adresa>
  <ulice>
    <typ>Nábřeží</typ>
    <název>Edvarda Beneše</název>
    <čo>4</čo>
  </ulice>
  <město>
    <název>Praha</název>
    <část>1</část>
  </město>
  <psč>118 01</psč>
</adresa>
```

- rozumný kompromis pro většinu aplikací:

```
<adresa>
  <ulice>Nábřeží Edvarda Beneše 4</ulice>
  <město>Praha 1</město>
  <psč>118 01</psč>
</adresa>
```

Elementy vs. atributy

Opakovaná hodnota

- atributy stejného jména se u jednoho elementu nemohou opakovat
- špatně:

```
<osoba tel1="123456789" tel2="987654321" tel3="111222333">
  …
</osoba>
```

- správně:

```
<osoba>
  …
  <tel>123456789</tel>
  <tel>987654321</tel>
  <tel>111222333</tel>
</osoba>
```

Elementy vs. atributy

Strukturované hodnoty

- hodnotu atributy nelze dále členit
- údaje, u kterých v budoucnu může vyvstat potřeba dalšího členění, by měly být rovnou v elementech
- špatně:

```
<osoba jméno="Jan Novák">
  …
</osoba>
```

- správně:

```
<osoba>
  <jméno>Jan Novák</jméno>
  …
</osoba>
```

v budoucnu lze snadno rozšířit na:

```
<osoba>
  <jméno>
    <křestní>Jan</křestní>
    <příjmení>Novák</příjmení>
  </jméno>
  …
</osoba>
```

Elementy vs. atributy

Volný text

- text v přirozeném jazyce by měl být vždy v elementu
- pro takový text můžeme určit jeho jazyk pomocí atributu `xml:lang`
- do elementu lze vkládat další podelementy např. pro vyznačení Ruby anotací, změnu směru textu apod.
- ukázka Ruby anotace

text anotace Ruby → とう きょう
základní text → 東 京

```
<ruby>
  <rb>東</rb>
  <rt>とう</rt>
  <rb>京</rb>
  <rt>きょう</rt>
</ruby>
```

Elementy vs. atributy

Shrnutí

- použitím pouze elementů nic nezkazíme
  - zápis může být delší než u atributů
  - v některých schémových jazycích lze hůře popisovat omezení
- atributy můžeme použít
  - pro údaje, které se neopakují
  - pro údaje, které se dále nestrukturují
  - pro údaje, u kterých dopředu známe obor hodnot (nejde tedy o texty v přirozeném jazyce) – čísla, datumy, hodnoty z číselníku

Modelování vztahů

- vztahy (1:1, 1:N, M:N) je možné řešit podobně jako v relačních databázích a nebo zanořováním elementů, případně kombinací
- pokud to situace dovoluje, je zanořování zcela přirozená metoda pro vyjádření vztahu
- ukázka vztahu 1:N řešeného zanořením

```
<faktura>
  <dodavatel>…</dodavatel>
  …
  <položka>…</položka>
  <položka>…</položka>
  <položka>…</položka>
  …
</faktura>
```

Kombinování schémat

Kombinování schémat

- každý schémový jazyk má trošku jiné vyjadřovací schopnosti
- pro důkladnou validaci je potřeba několik schémat zkombinovat
- nejčastější kombinace
  - Relax NG + Schematron
  - W3C XML Schema + Schematron
- validace komponovaných dokumentů

Validace komponovaných dokumentů

- dokument se skládá z několika různých sad značek
- každá sada je ve vlastním jmenném prostoru (např. XHTML a SVG)
- NVDL (Namespace-based Validation Dispatching Language) – jazyk pro tvorbu meta-schémat popisujících validaci komponovaných dokumentů

```
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0">
  <namespace ns="http://www.w3.org/1999/xhtml">
    <validate schema="xhtml.xsd"/>
  </namespace>
  <namespace ns="http://www.w3.org/2000/svg">
    <validate schema="svg.rng"/>
  </namespace>
</rules>
```

Další zdroje informací

Odkazy

- stránky W3C[7] – specifikace a další odkazy
- XSV[8]
- Topologi Schema Validator[9]
- Trang[10] – nástroj na konverzi mezi schématy
- MSV[11] – validátor s podporou mnoha různých schémových jazyků
- oXygen[12] – XML editor včetně grafického editoru schémat
- XmlSpy[13] – XML editor včetně grafického editoru schémat
- XML Schema and RelaxNG Tutorial[14]
- RELAX NG[15] – volně dostupná kniha od Erica van der Vlista
- XML schémata[16] – tutoriál v češtině

[7] http://www.w3.org/XML/Schema
[8] http://www.ltg.ed.ac.uk/~ht/xsv-status.html
[9] http://www.topologi.com/products/validator/
[10] http://www.thaiopensource.com/relaxng/trang.html
[11] https://msv.dev.java.net/
[12] http://www.oxygenxml.com/
[13] http://www.altova.com/products_ide.html
[14] http://zvon.org/xxl/XMLSchemaTutorial/Output/index.html
[15] http://books.xmlschemata.org/relaxng/
[16] http://www.kosek.cz/xml/schema/