

# XML schémata

## Tutoriál

Jirka Kosek

KIZI a LISP, FIS VŠE Praha  
nám. W. Churchilla 4, 130 67, Praha 3  
<http://www.kosek.cz>  
<jirka@kosek.cz>

DATAKON 2003  
18.-21. října 2003

Hotel SANTON, Brno

## Agenda

- proč a nač jsou schémata
- přehled nejpoužívanějších jazyků pro popis schématu dokumentu
- základy jazyka W3C XML Schema
  - jednoduché datové typy
  - komplexní datové typy
  - jmenové prostory a validace
  - přístupy k návrhu schématu
  - pokročilé vlastnosti
  - praktické využití schémat

1

2

Úvod

Proč potřebujeme schéma dokumentů XML

### 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

Úvod

Přínosy použití schématu

### 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

XML schémata

5

XML schémata

6

Ukázky nepoužívanějších jazyků pro popis schématu dokumentu Ukázky

### Ukázkový dokument

```
<?xml version="1.0" encoding="windows-1250"?>
<zamestnanci>
  <zamestnanec id="101">
    <jmeno>Jan</jmeno>
    <prijmeni>Novák</prijmeni>
    <email>jan@novak.cz</email>
    <email>jan.novak@firma.cz</email>
    <plat>25000</plat>
    <narozen>1965-12-24</narozen>
  </zamestnanec>
  <zamestnanec id="102">
    <jmeno>Petra</jmeno>
    <prijmeni>Procházková</prijmeni>
    <email>prochazkovap@firma.cz</email>
    <plat>27500</plat>
    <narozen>1974-03-21</narozen>
  </zamestnanec>
</zamestnanci>
```

Ukázky nepoužívanějších jazyků pro popis schématu dokumentu DTD

### DTD

```
<!ELEMENT zamestnanci (zamestnanec+)>
<!ELEMENT zamestnanec (jmeno, prijmeni, email+,
  plat?, narozen)>
<!ELEMENT jmeno (#PCDATA)>
<!ELEMENT prijmeni (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT plat (#PCDATA)>
<!ELEMENT narozen (#PCDATA)>
<!ATTLIST zamestnanec
  id CDATA #REQUIRED>
```

XML schémata

7

XML schémata

8

## XML Schema

```
<?xml version="1.0" encoding="windows-1250"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="zamestnanci">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="zamestnanec"
          maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="jmeno" type="xs:string"/>
              <xs:element name="prijmeni" type="xs:string"/>
              <xs:element name="email" type="xs:string"
                maxOccurs="unbounded"/>
              <xs:element name="plat" type="xs:decimal"
                minOccurs="0"/>
              <xs:element name="narozen" type="xs:date"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:int"
              use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML schémata

9

## Relax NG

```
<?xml version="1.0" encoding="windows-1250"?>
<element xmlns="http://relaxng.org/ns/structure/1.0"
  name="zamestnanci">
  <oneOrMore>
    <element name="zamestnanec">
      <attribute name="id">
        <text/>
      </attribute>
      <element name="jmeno">
        <text/>
      </element>
      <element name="prijmeni">
        <text/>
      </element>
      <oneOrMore>
        <element name="email">
          <text/>
        </element>
      </oneOrMore>
      <optional>
        <element name="plat">
          <text/>
        </element>
      </optional>
      <element name="narozen">
        <text/>
      </element>
    </oneOrMore>
  </element>
</element>
```

XML schémata

10

## Relax NG

### Doplněno o datové typy

```
<?xml version="1.0" encoding="windows-1250"?>
<element xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-
  datatypes"
  name="zamestnanci">
  <oneOrMore>
    <element name="zamestnanec">
      <attribute name="id">
        <data type="int"/>
      </attribute>
      <element name="jmeno">
        <data type="string"/>
      </element>
      <element name="prijmeni">
        <data type="string"/>
      </element>
      <oneOrMore>
        <element name="email">
          <data type="string"/>
        </element>
      </oneOrMore>
      <optional>
        <element name="plat">
          <data type="decimal"/>
        </element>
      </optional>
      <element name="narozen">
        <data type="date"/>
      </element>
    </oneOrMore>
  </element>
</element>
```

XML schémata

11

## Relax NG

### Kompaktní syntaxe

```
element zamestnanci {
  element zamestnanec {
    attribute id { text },
    element jmeno { text },
    element prijmeni { text },
    element email { text }+,
    element plat { text }?,
    element narozen { text }
  }+
}

element zamestnanci {
  element zamestnanec {
    attribute id { xsd:int },
    element jmeno { xsd:string },
    element prijmeni { xsd:string },
    element email { xsd:string }+,
    element plat { xsd:decimal }?,
    element narozen { xsd:date }
  }+
}
```

XML schémata

12

### 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>

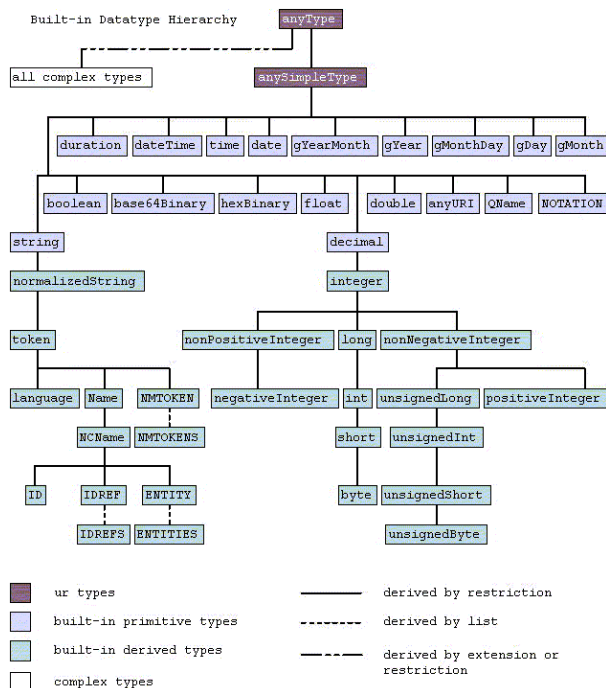
```

- všechny elementy a datové typy patří do jmenného prostoru XML schémat
- pro každý element/atribut musíme určit datový typ

### Datové typy

- lze použít pro obsah atributů i elementů
- komplexní x 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

```

<xs:element name="zamestnanec">
  <xs:annotation>
    <xs:documentation>Element slouží pro uchování důležitých
      údajů o zaměstnanci.</xs:documentation>
  </xs:annotation>
  <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:complexType>
</xs:element>

```

## 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

XML schémata

17

Jednoduché typy

Vytváření vlastních typů

## Vytváření vlastních typů

### Vytvoření a použití typu pro měnové údaje

- maximální částka 1 milión
- přesnost na haléře

```
<cenaVýrobku>23.50</cenaVýrobku>
```

```
<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>
```

```
<xs:element name="cenaVýrobku" type="částka"/>
```

XML schémata

19

## 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 perlou syntaxi
  - příklad – DIČ: `\d{3}-\d{10}`

XML schémata

18

Jednoduché typy

Vytváření vlastních typů

## Vytváření vlastních typů

### Vytvoření typu pro kód měny, deklarace atributu

```
<cena měna="USD">23.50</cena>
```

```
<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>
```

```
<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>
```

XML schémata

20

## 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:binary"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

## Modelování obsahu

- sekvence za sebou jdoucích elementů – <xs:sequence>
- výběr jednoho z elementů – <xs:choice>
- nezáleží na pořadí elementů – <xs:all>
- 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>
  <název>Ukázka</název>
  <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>
  <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

```

```

```
<xs:element name="img">
  <xs:complexType>
    <xs:attribute name="src" type="xs:anyURI"/>
  </xs:complexType>
</xs:element>
```

```
<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
- 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:

```

```

```

```

### chybné zápisy:

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

```

```

XML schémata

29

Jmenné prostory

Co to jsou jmenné prostory

```
      s vektorovým obrázkem</title>
</head>
<body>
  <p>Zajímavý obrázek:</p>
  <svg:svg width="4in" height="3in"
    xmlns:svg="http://www.w3.org/2000/svg">
    <svg:desc>This is a blue circle with
      a red outline</svg:desc>
    <svg:g>
      <svg:circle style="fill: blue; stroke: red" cx="200"
        cy="200" r="100"/>
      <svg:text x=".5in" y="2in">Hello World</svg:text>
    </svg:g>
  </svg:svg>
</body>
</html>
```

XML schémata

31

## Co to jsou jmenné prostory

- umožňují kvalifikovat elementy/atributy
- umožňují v jednom dokumentu kombinovat několik různých sad značek (např. v XHTML dokumentu je obrázek v SVG a rovnice v MathML)
- jmenný prostor je identifikován pomocí URI adresy
- podle jmenného prostoru aplikace poznají, kterým částem XML dokumentu rozumějí

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Ukázka webové stránky
      s vektorovým obrázkem</title>
  </head>
  <body>
    <p>Zajímavý obrázek:</p>
    <svg width="4in" height="3in"
      xmlns="http://www.w3.org/2000/svg">
      <desc>This is a blue circle with a red outline</desc>
      <g>
        <circle style="fill: blue; stroke: red" cx="200"
          cy="200" r="100"/>
        <text x=".5in" y="2in">Hello World</text>
      </g>
    </svg>
  </body>
</html>
```

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Ukázka webové stránky
```

XML schémata

30

Jmenné prostory

Globální deklarace

## 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`
- 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">
  <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>
```

XML schémata

32

## 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 pomoci atributu `form` určit, zda mají patřit do cílového jmenného prostoru
- 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>
```

XML schémata

33

Validace

Připojení schéma k dokumentu

## Připojení schéma k dokumentu

### Používáme vlastní jmenný prostor

#### XML dokument

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

nebo bez deklarace implicitního jmenného prostoru

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

#### XML schéma – dokument.xsd

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

XML schémata

35

## Připojení schéma k dokumentu

### Nepoužíváme vlastní jmenný prostor

- před validací musíme parseru sdělit, kde pro dokument najde jeho schéma
- standardně se umístění schématu zaznamená do speciálních atributů kořenového elementu
- některá prostředí umožňují používání cache na schémata, kde se schéma zaregistruje a parser jej pak sám najde

#### XML dokument

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

#### XML schéma – dokument.xsd

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

XML schémata

34

Validace

Podpora schémat v parserech

## Podpora schémat v parserech

- Xerces (<http://xml.apache.org/>)
  - součást projektu Apache
  - kód z velké části věnovalo IBM
  - open source projekt
  - platforma: Java
- MSXML4, System.Xml
  - autor: Microsoft
  - platforma: Win32, .NET
- XSV (<http://www.w3.org/2001/03/webdata/xsv>)
  - autor: University of Edinburgh
  - platforma: Python
- ... a mnoho dalších
- ukázka validace ve VS.NET, Xercesu, XML Spy

XML schémata

36

## Matrjůška

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

```
<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:sequence>
  </xs:complexType>
</xs:element>
```

## 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

```
<xs:element name="jmeno" type="xs:string"/>
<xs:element name="prijmeni" type="xs:string"/>
<xs:element name="plat" type="xs:decimal"/>

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

## Metoda slepého Benátčana

- pro všechny elementy se definují typy
- elementy jsou definovány lokálně pomocí těchto typů
- spojuje výhody předchozích dvou přístupů
- nejupovídanější a nejpracnější

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

<xs:element name="zamestnanec">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="jmeno" type="jmenoType"/>
      <xs:element name="prijmeni" type="prijmeniType"/>
      <xs:element name="plat" type="platType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

## „Best practices“ pro návrh

- každé schéma by mělo mít definováno vlastní cílový jmenný prostor (`targetNamespace`)
- všechny elementy by měly patřit do tohoto jmenného prostoru (`elementFormDefault="qualified"`)
- nepoužívat standardní ani fixní hodnoty (`default="..."`, `fixed="..."`) protože mění infoset
- konzistentní a srozumitelné názvy elementů/atributů
  - moc krátké jsou nesrozumitelné, příliš dlouhé se špatně píší
  - `<cenaVýrobku kódMěny="USD">5.99</cenaVýrobku>`
- elementy × atributy
  - atributy se nemohou opakovat
  - atributy nelze dále strukturovat
- dnes se doporučuje do atributů ukládat pouze údaje, u kterých je předem známá množina přípustných hodnot, případně se jedná o typy jako číslo nebo datum
- obecné textové řetězce není vhodné ukládat do atributů, protože v budoucnu může vzniknout požadavek na doplnění značkování (např. Ruby anotace, změna směru textu ve vícejazyčném dokumentu)

## Práce s prázdnými hodnotami (NULL)

- elementy, které mohou být prázdné, musíme takto definovat

```
<xs:element name="autor" nillable="true"
  type="xs:string"/>
```

- autor je prázdný řetězec

```
<autor></autor>
<autor/>
```

- autor má hodnotu „NULL“

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

- nepřipustné použití

```
<autor xsi:nil="true">Novák</autor>
```

- 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

## Ukázka unikátního klíče

### 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

### 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

## Databinding

- pro data s předem známou strukturou je čtení pomocí klasických API jako SAX nebo DOM nepohodlné
- ze schématu jde vygenerovat hierarchie tříd, které jsou schopné reprezentovat data uložená v XML
- třídy obsahují kód pro serializaci/deserializaci
- datové typy XML schémat se namapují na datové typy daného jazyka
- implementace – Castor, JAXB (Java), xsd (.NET)

XML schémata

45

XML schémata

46

Praktické využití schémat

Databinding

## Databinding

### Ukázka schématu

```
<xs:element name="faktura">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="odberatel" type="subjektInfoTyp" />
      <xs:element name="dodavatel" type="subjektInfoTyp" />
      <xs:element ref="polozka" minOccurs="1"
        maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="cislo" type="xs:string"
      use="required" />
    <xs:attribute name="vystaveni" type="xs:date"
      use="required" />
    <xs:attribute name="splatnost" type="xs:date"
      use="required" />
    <xs:attribute name="vystavil" type="xs:string" />
  </xs:complexType>
</xs:element>

<xs:complexType name="subjektInfoTyp">
  <xs:sequence>
    <xs:element name="nazev" type="xs:string" />
    <xs:element name="adresa" type="xs:string" />
    <xs:element name="ico" type="xs:string" />
    <xs:element name="dic" type="xs:string" />
  </xs:sequence>
</xs:complexType>

<xs:element name="polozka">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="popis" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="cena" type="xs:decimal" />
      <xs:element name="dph" type="xs:decimal" />
      <xs:element name="ks" type="xs:positiveInteger" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML schémata

47

Praktické využití schémat

Databinding

```
minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>
</xs:element>
```

XML schémata

48

## Databinding

### Ukázka rozhraní vygenerovaných tříd

```
public class faktura {
    public subjektInfoTyp odberatel;
    public subjektInfoTyp dodavatel;
    public polozka[] polozka;
    public string cislo;
    public System.DateTime vystaveni;
    public System.DateTime splatnost;
    public string vystavil;
}

public class subjektInfoTyp {
    public string nazev;
    public string adresa;
    public string ico;
    public string dic;
}

public class polozka {
    public string popis;
    public System.Decimal cena;
    public System.Decimal dph;
    public string ks;
}
```

## Databinding

### Práce s dokumentem

```
// stream pro čtení dat ze souboru
StreamReader reader = new StreamReader("faktura.xml");

// serializátor založený na třídě vygenerované ze schématu
// pomocí: xsd /c faktura.xsd
XmlSerializer serializer = new
    XmlSerializer(typeof(faktura));

// do objektu f se deserializuje celý dokument faktury
faktura f = (faktura)serializer.Deserialize(reader);

// pomocné proměnné
decimal suma = 0;
decimal sumaDPH = 0;

// sečtení je hračka, nemusíme se starat ani o datové typy
foreach (polozka p in f.polozka)
{
    suma += p.cena;
    sumaDPH += p.cena * (p.dph/100);
}

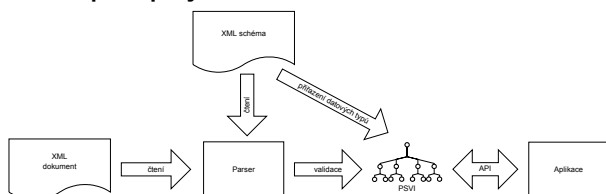
// výpis statistiky
System.Console.WriteLine("Celkem Kč: " + suma);
System.Console.WriteLine("Celkem DPH: " + sumaDPH);
```

- dokument je samozřejmě R/W, takže jej můžeme v paměti i vytvořit, upravovat a pak uložit

## Práce s PSVI

- infoset = abstraktní datový model dokumentu XML
- PSVI = Post Schema Validation Infoset
  - otypovaný infoset dokument
  - vznikne přiřazením datových typů na základě validace oproti schématu
  - při čtení přes API můžeme pracovat přímo s typovými hodnotami
  - bez PSVI nám API může vrátit jen textové řetězce
  - nad PSVI pracují moderní dotazovací jazyky jako XPath 2.0 nebo XQuery

### Parser zpřístupňující PSVI



## Odkazy

- stránky W3C (<http://www.w3.org/XML/Schema>) – specifikace a další odkazy
- XSV (<http://www.ltg.ed.ac.uk/~ht/xsv-status.html>)
- Topologi Schema Validator (<http://www.topologi.com/products/validator/>)
- Trang (<http://www.thaiopensource.com/relaxng/trang.html>) – nástroj na konverzi mezi schématy
- XmlSpy ([http://www.altova.com/products\\_ide.html](http://www.altova.com/products_ide.html)) – XML editor včetně grafického editoru schémat
- XML Schema and RelaxNG Tutorial (<http://zvon.org/xxl/XMLSchemaTutorial/Output/index.html>)
- RELAX NG (<http://books.xmlschemata.org/relaxng/>) – volně dostupná kniha od Erica van der Vlisty
- [cz.comp.lang.xml](http://cz.comp.lang.xml) – česká diskusní skupina o XML