INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Realizováno za finanční podpory ESF a státního rozpočtu ČR v rámci v projektu *Zkvalitnění a rozšíření možností studia na TUL pro studenty se SVP* reg. č. CZ.1.07/2.2.00/29.0011

RELAX NG

RELAX NG

- **REgular LAnguage for XML Next Generation**
- vytvořila **OASIS** (Organization for the Advancement of Structured Information Standards)
  - původně sdružení výrobců SGML nástrojů
  - později přeorientováno na XML
  - vyvinulo desítky standardů (DocBook, OpenDocument, SAML,...)
- RELAX NG vznikl 2002

Základní principy

- postaveno na vzorech, nikoli datových typech
  - schéma je vzorem dokumentu
  - obsahuje vzory prvků, atributů a textových uzlů
- dvě alternativní syntaxe
  - XML – jmenný prostor http://relaxng.org/ns/structure/1.0
  - textová, kompaktní
  - ekvivalentní a vzájemně převoditelné

Základní vzory (1)

- **text** – neobsahuje vnořené prvky, i prázdný
  - <text/>
  - text
- **prvek** – obsahuje vzory pro svůj obsah
  - <element name="*jméno*">
    obsah
    </element>
  - element *jméno* { *obsah* }

Příklad: osoba

```
<osoba>
    <krestni>Josef</krestni>
    <prijmeni>Novák</prijmeni>
</osoba>
```

```
<element name="osoba">
    <element name="krestni">
        <text/>
    </element>
    <element name="prijmeni">
        <text/>
    </element>
</element>
```

```
element osoba {
    element krestni { text },
    element prijmeni { text }
}
```

Základní vzory (2)

- **atributy** – uvnitř prvku, nezáleží na pořadí
  - <attribute name="*jméno*"><text/></attribute>
    lze zkrátit na <attribute name="*jméno*"/>
  - attribute *jméno* { *text* }

```
<element name="osoba">
    <attribute name="id"/>
    <element name="krestni">
        <text/>
    </element>
    ...
</element>
```

```
element osoba {
    attribute id { text },
    element krestni { text },
    element prijmeni { text }
}
```

Prázdný prvek

- prvek bez obsahu
  - <element name="*jméno*">**<empty/>**</element>
  - element *jméno* { **empty** }
- pokud obsahuje atributy, lze vzor empty vynechat

Opakování (1)

- nepovinný výskyt
  - <optional> – prvky zabalené jsou nepovinné
  - ? – připojuje se za konstrukci
- libovolný počet výskytů
  - <zeroOrMore>
  - *
- alespoň jeden výskyt
  - <oneOrMore>
  - +

Příklad: nepovinné id, více jmen

```
<element name="osoba">
    <optional>
        <attribute name="id"/>
    </optional>
    <oneOrMore>
        <element name="krestni">
            <text/>
        </element>
    </oneOrMore>
    <element name="prijmeni">
        <text/>
    </element>
</element>
```

```
element osoba {
    attribute id { text }?,
    element krestni { text }+,
    element prijmeni { text }
}
```

Opakování (2)

- nelze určit konkrétní rozsah pro počet opakování (něco jako minOccurs, maxOccurs)
- musí se vyjádřit explicitně – např. 2 až 4 jména:

```
element jmeno { text },
element jmeno { text },
element jmeno { text }?,
element jmeno { text }?
```

- naštěstí je tato potřeba vzácná

Kombinování vzorů

Alternativy

- v dokumentu má být právě jeden z prvků
  - <choice>
  - |

```
<element name="stav">
    <choice>
        <element name="zenaty">
            <text/>
        </element>
        <element name="svobodny"><empty/></element>
    </choice>
</element>
```

```
element stav {
    element zenaty { text }
  | element svobodny { empty }
}
```

Zachování pořadí

- implicitně zachovává
  - lze vyjádřit pomocí <group>
  - kompaktní schéma: čárky, případně závorky
- explicitní vyjádření může být potřebné, jedná-li se např. o pevnou podskupinu mezi alternativami

Příklad

```
<element name="sluzba">
    <choice>
        <element name="url"/>
            <text/>
        </element>
        <group>
            <element name="adresa">
                <text/>
            </element>
            <element name="port">
                <text/>
            </element>
        </group>
    </choice>
</element>
```

- služba je určena jménem nebo adresou a portem

```
element sluzba {
    element url { text }
    | ( element adresa { text },
        element port { text } )
}
```

Libovolné pořadí

- pokud připouštíme vzory v libovolném pořadí
  - obalit prvkem <interleave>
  - oddělovat znakem &

```
<element name="kniha">
 <element name="nazev"/><text/></element>
 <element name="autor"/><text/></element>
 <interleave>
    <element name="isbn"><text/></element>
    <element name="cena"><text/></element>
 </interleave>
</element>
```

```
element kniha {
    element nazev { text },
    element autor { text },
    (    element isbn { text }
     &  element cena { text } )

}
```

Prolínání je silnější

- lze definovat několik prvků (či jejich skupin s definovaným pořadím), které se pak mohou navzájem prolínat
- např. definici a { (x,y) & (p,q) } vyhoví
  <a><x/><y/><p/><q/>
  <a><x/><p/><q/><y/>
  <a><p/><x/><q/><y/> ...
  (<x> musí být před <y> a <p> musí být před <q>, ale vzájemná pozice těchto dvojic je libovolná)

Příklad – komentáře

```
<element name="kniha">
 <interleave>
  <group>
   <element name="nazev"/><text/></element>
   <element name="autor"/><text/></element>
   <element name="isbn"><text/></element>
  </group>
  <zeroOrMore>
   <element name="koment"><text/></element>
  </zeroOrMore>
 </interleave>
</element>
```

```
element kniha {
 (
  ( element nazev { text },
    element autor { text },
    element isbn { text }
  )
 &
  element koment { text }*
 )
}
```

Smíšený obsah

- lze pomocí <interleave> kde jednou z variant je <text/> (nemusí se mu dávat opakování, vyhovuje libovolnému počtu uzlů)
- celkem časté – prvek <mixed> (resp. mixed { ... }) nemusí mít vnořený <text/>
  - problém: obsahuje-li více prvků, automaticky kolem nich vytváří <group>
  - řešení: zabalit je do <interleave> (nebo nepoužívat <mixed>)

Příklad: <p> s vnořeným textem, a

```
<element name="p">
 <mixed>
  <interleave>
   <zeroOrMore>
    <element name="em"/>
     <text/>
    </element>
   </zeroOrMore>
   <zeroOrMore>
    <element name="strong"/>
     <text/>
    </element>
   </zeroOrMore>
  </interleave>
 </mixed>
</element>
```

```
element p {
  mixed {
    element em { text }*
    & element strong { text }*
  }
}
```

```
element p {
  element em { text }*
  & element strong { text }*
  & text
}
```

Datové typy

Datové typy (1)

- sám má jen dva typy:
  - **string**
  - **token**
- oba reprezentují textový řetězec
- token před validací normalizuje prázdné místo (mezery, tabulátory, konce řádků)

Datové typy (2)

- zpravidla se vkládají externí knihovny datových typů, nejčastěji z XML Schema
- **XML syntaxe**
  - prvek <data>, typ určují atributy:
  - type – datový typ
  - datatypeLibrary – knihovna, z níž pochází; XML Schema http://www.w3.org/2001/XMLSchema-datatypes; dědí se

Příklad: kniha

```
<element name="kniha"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
    <element name="nazev">
        <data type="string"/>
    </element>
    ...
    <element name="cena">
        <data type="decimal"/>
    </element>
</element>
```

Datové typy (3)

- kompaktní syntaxe
  - pomocí datatypes se definuje prefix knihovny typů
  - ten se pak používá v definicích
  - pro XML schema předdefinován prefix xsd

```
datatypes xs = "http://www.w3.org/2001/XMLSchema-datatypes"

element kniha {
    element nazev { xs:string },
    ...
    element cena { xs:decimal }
}
```

Omezení datových typů

- pomocí parametrů
- **XML syntaxe:**
  - prvek <param name="*jméno*">*hodnota*</param>
  - atribut name udává jméno parametru (typ omezení)
  - obsah prvku určuje hodnotu daného omezení
- **kompaktní syntaxe:**
  - ve složených závorkách za typem
  - ve tvaru *jméno*="*hodnota*"

Příklad: XML syntaxe

```
<element name="kontakt"
    xmlns="http://relaxng.org/ns/structure/1.0"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
    <element name="jmeno">
        <data type="string">
            <param name="maxLength">50</param>
        </data>
    </element>
    <element name="telefon">
        <data type="token">
            <param name="pattern">\d{3}\s*\d{3}\s*\d{3}</param>
        </data>
    </element>
</element>
```

Příklad: kompaktní syntaxe

```
element kontakt {
    element jmeno {
        xsd:string { maxLength = "50" }
    },
    element telefon {
        xsd:token { pattern ="\d{3}\s*\d{3}\s*\d{3}" }
    }
}
```

- parametry pro omezení vycházejí z XML Schema, kromě enumeration a whiteSpace

Výčtový typ

- standardní výběr, dostupné hodnoty:
  - <value>*hodnota*</value>
  - "*hodnota*"

```
<element name="cena">
    <attribute name="mena">
        <choice>
            <value>CZK</value>
            <value>USD</value>
            <value>EUR</value>
        </choice>
    </attribute>
    <data type="decimal"/>
</element>
```

```
element cena {
    attribute mena {
        "CZK"
        | "USD"
        | "EUR" },
    xsd:decimal
}
```

Zakázané hodnoty

- některé hodnoty lze vyloučit:
  - <except>*hodnota*</except>
  - – (*hodnota*)

```
<element name="promenna">
    <data type="string">
        <except>
            <choice>
                <value>begin</value>
                <value>end</value>
            </choice>
        </except>
    </data>
</element>
```

```
element promenna {
    xsd:string –
    (( "begin" | "end" ))
}
```

Seznamy

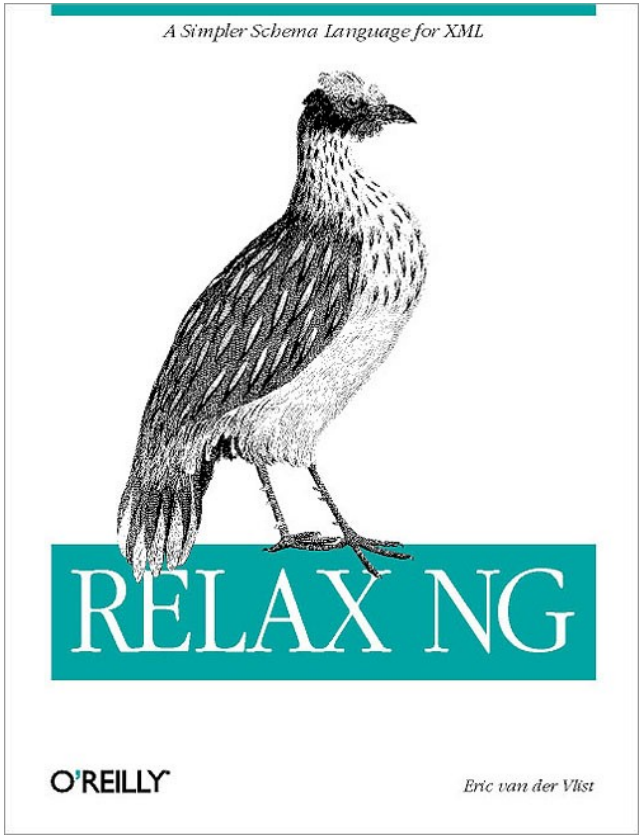
- hodnoty oddělované prázdným místem
  - <list>*vzor pro obsah*</list>
  - list { *vzor pro obsah* }
- raději méně – odporuje duchu XML

```
<element name="posloupnost">
    <list>
        <oneOrMore>
            <data type="decimal"/>
        </oneOrMore>
    </list>
</element>
```

```
element posloupnost {
    list { xsd:decimal+ }
}
```

Doporučená četba

- Eric van der Vlist: *Relax NG*
  O'Reilly, 2003



dostupná volně na
http://books.xmlschemata.org/relaxng/