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

Vytvoření schématu

Obal schématu

- celé schéma obaluje prvek **schema** z jmenného prostoru přiděleného XML Schema:

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

- atributy umožňují určit různé vlastnosti

Připojení schématu k dokumentu

- pomocí atributů z prostoru http://www.w3.org/2001/XMLSchema-instance
- obvyklým prefixem je xsi
- atribut noNamespaceSchemaLocation pokud nepoužíváme jmenné prostory, hodnotou lokátor

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

Připojení se jmenným prostorem

- atribut schemaLocation
- hodnotou dvojice (i několik)
  URI_prostoru lokátor_schématu

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

Jak vytvářet schéma?

- obvyklé jsou tři základní metody
  - matrjoška
  - plátkování
  - slepý Benátčan
- volba závisí především na složitosti a charakteru popisovaného jazyka
- lze kombinovat (ale s mírou)

Příklad: ceník zboží

- poslouží k ilustraci jednotlivých přístupů
- definice XML prvku zbozi, který má mít při použití následující tvar:

```
<zbozi kod="1267">
    <nazev>Žízeň Killer – voda 1,5 l</nazev>
    <cena>8.50</cena>
</zbozi>
```

Matrjoška

- definice prvků se do sebe ve schématu vkládají přímo, stejně jako prvky v dokumentu
- anonymní typy
- na nejvyšší úrovni vždy jen jeden prvek
- krátké a kompaktní schéma
- pro složitější jazyky nepřehledné
- nelze opakovaně využívat dílčí definice

Matrjoška – příklad

```
<xsd:element name="zbozi" maxOccurs="unbounded">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="nazev" type="xsd:string"/>
            <xsd:element name="cena" type="xsd:decimal"/>
        </xsd:sequence>
        <xsd:attribute name="kod" type="xsd:integer"/>
    </xsd:complexType>
</xsd:element>
```

Plátkování

- všechny prvky a atributy se definují na stejné (nejvyšší) úrovni
- odkazují se na sebe pomocí ref="*jméno*"
- definované prvky/atributy lze používat opakovaně
- koncepce blízká DTD
- obsah prvku se nemůže lišit podle kontextu (rodiče)

Plátkování – příklad

```
<xsd:element name="nazev" type="xsd:string"/>
<xsd:element name="cena" type="xsd:decimal"/>
<xsd:attribute name="kod" type="xsd:integer"/>

<xsd:element name="zbozi">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="nazev"/>
            <xsd:element ref="cena"/>
        </xsd:sequence>
        <xsd:attribute ref="kod"/>
    </xsd:complexType>
</xsd:element>
```

Slepý Benátčan

- předem se (na globální úrovni) definují typy
- prvky a atributy se definují lokálně (jako v matrjošce), ovšem triviálně s využitím připravených typů
- nejflexibilnější
- typy lze používat opakovaně
- ale schéma je delší
- vhodné pro složité jazyky

Slepý Benátčan – příklad (1)

```
<xsd:simpleType name="nazevTyp">
    <xsd:restriction base="xsd:string">
        <xsd:maxLength value="50"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="cenaTyp">
    <xsd:restriction base="xsd:decimal">
        <xsd:fractionDigits value="2"/>
    </xsd:restriction>
</xsd:simpleType>
```

Slepý Benátčan – příklad (2)

```
<xsd:simpleType name="kodTyp">
    <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>

<xsd:complexType name="zboziTyp">
    <xsd:sequence>
        <xsd:element name="nazev" type="nazevTyp"/>
        <xsd:element name="cena" type="cenaTyp"/>
    </xsd:sequence>
    <xsd:attribute name="kod" type="kodTyp"/>
</xsd:complexType>

<xsd:element name="zbozi" type="zboziTyp"/>
```

Pokročilé konstrukce

Skupiny

- prvky **group** a **attributeGroup**
- umožňují definovat obvyklé kombinace (např. základní sadu atributů pro všechny prvky)
- nejsou typy, jen „kontejnery“
- použití: atribut ref="*jméno skupiny*"

Skupiny – příklad (1)

```
<xsd:group name="prvkyZbozi">
    <xsd:sequence>
        <xsd:element name="nazev" type="nazevTyp"/>
        <xsd:element name="cena" type="cenaTyp"/>
    </xsd:sequence>
</xsd:group>

<xsd:attributeGroup name="spolecneAtr">
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="style" type="xsd:string"/>
</xsd:attributeGroup>
```

Skupiny – příklad (2)

```
<xsd:complexType name="zboziTyp">
    <xsd:group ref="prvkyZbozi"/>
    <xsd:attributeGroup ref="spolecneAtr"/>
    <xsd:attribute name="kod" type="kodTyp"/>
</xsd:complexType>
```

Unikátní hodnoty

- lze požadovat, aby hodnoty určitého prvku či atributu (nebo jejich kombinace) byly v dané části dokumentu unikátní
- prvek **unique**
  - místo výskytu: kde má být jednoznačné
  - selector: v čem hledat prvek/atribut
  - field: který prvek/atribut má být jednoznačný
  - identifikace vychází z XPath

Příklad – unikátní kódy zboží

```
<xsd:element name="cenik">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="zbozi" type="zboziTyp"
                                maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="unikatniKod">
        <xsd:selector xpath="zbozi"/>
        <xsd:field xpath="@kod"/>
    </xsd:unique>
</xsd:element>
```

Klíče a jejich použití

- definice klíče: prvek **key**
  - odpovídá unique, povinný
- použití: prvek **keyref**
  - atribut refer – jméno definovaného klíče
  - vnořené prvky selector a field určují prvek/atribut, jehož hodnotou má být hodnota daného klíče

Příklad klíče (1)

- příklad: zboží může být součástí jiného – nepovinný prvek soucast, jehož hodnotou je kód jiného zboží

```
<xsd:complexType name="zboziTyp">
    ...
    <xsd:element name="soucast" type="kodTyp"
                 minOccurs="0" maxOccurs="unbounded"/>
</xsd:complexType>
```

Příklad klíče (2)

```
<xsd:element name="cenik">
    <xsd:complexType>...</xsd:complexType>
    <!-- atribut kod je klíčem, musí být jednoznačný -->
    <xsd:key name="unikatniKod">
        <xsd:selector xpath="zbozi"/>
        <xsd:field xpath="@kod"/>
    </xsd:key>
    <!-- hodnotou soucast musí být existující kód zboží -->
    <xsd:keyref name="odkazNaKod" refer="unikatniKod">
        <xsd:selector xpath="zbozi"/>
        <xsd:field xpath="soucast"/>
    </xsd:keyref>
<xsd:element>
```

<soucast> ve <zbozi> musí obsahovat platnou hodnotu klíče unikatniKod

Jmenný prostor jazyka

- atribut **targetNamespace** kořenového prvku schema definuje jazykový prostor pro definované prvky
- vhodné též definovat jako implicitní prostor
- problém: do jmenného prostoru patří jen prvky nejvyšší úrovně, nikoli prvky vnořené
- řešení: atribut **elementFormDefault="qualified"** určí, že všechny prvky schématu jsou kvalifikované (zařazené do jmenného prostoru)
- analogicky **attributeFormDefault** (ale atributy bývají bez jmenného prostoru)

Příklad – jmenný prostor

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.tul.cz/satrapa/pokus1"
    xmlns="http://www.tul.cz/satrapa/pokus1"
    elementFormDefault="qualified">
... definice typů a skupin ...
<xsd:element name="cenik">
    ... definice prvků ...
</xsd:element>
</xsd:schema>
```

Skládání schémat

- rozložení složitého schématu, knihovny prvků
- <xsd:**include** schemaLocation="*lokátor*"/>
  se nahradí textem z daného lokátoru („opíše“ text)
- <xsd:**redefine** schemaLocation="*lokátor*">
  předefinované součásti
  </xsd:redefine>
  vloží text z lokátoru, ale předefinuje jeho vybrané součásti podle svého těla

Import schémat

- pro schémata, která mají mít svůj vlastní prostor
- prvek **import**, atributy
  - namespace – jmenný prostor vkládaného schématu
  - schemaLocation – lokátor souboru obsahujícího schéma
- např. chci vložit schéma pro MathML

```
<xsd:import
    namespace="http://www.w3.org/1998/Math/MathML"
    schemaLocation="schema/mathml.xsd"/>
```

Dokumentace

- lze používat XML komentáře
- většina konstrukcí může obsahovat (na začátku) prvek **annotation** obsahující popis jejich určení
- potomci:
  - **documentation** – pro lidského čtenáře
  - **appinfo** – pro zpracovávající aplikaci
- existují nástroje pro generování dokumentace (např. xnsdoc, XSDDoc)

Příklad dokumentace

```
<xsd:element name="cenik">
    <xsd:annotation>
        <xsd:documentation xml:lang="cs">
        Kořenový prvek obsahující ceník zboží.
        </xsd:documentation>

        <xsd:documentation xml:lang="en">
        Root element containing the price list.
        </xsd:documentation>
    </xsd:annotation>
    ... definice obsahu prvku ...
</xsd:element>
```

Zastupující typy

- ve schématu lze stanovit, že určité typy mohou zastoupit jiný typ
- oficiálně: vznikne **substituční skupina**
  - **hlava skupiny**: prvek zastupitelný ostatními členy skupiny; neobsahuje žádné speciální konstrukce
  - syntaxe: zastupující prvky ve své definici obsahují atribut substitutionGroup="hlava_skupiny"
  - zastupující prvek musí být stejného nebo odvozeného typu jako hlava

Příklad 1

- místo prvku <zbozi> se v datech může vyskytovat i <ware> nebo <artikel>
- pokud je typ stejný, nemusí se uvádět:

```
<xsd:element name="zbozi" type="zboziTyp"/>
<xsd:element name="ware" substitutionGroup="zbozi"/>
<xsd:element name="artikel" substitutionGroup="zbozi"/>
```

Příklad 2

- zastupující typ může být odvozen od zastupovaného

```
<xsd:element name="cd" type="cdTyp"
                       substitutionGroup="zbozi">
<xsd:complexType name="cdTyp">
  <xs:complexContent>
    <xs:extension base="zboziTyp">
      <xs:sequence>
        <xs:element name="interpret" type="interpretTyp"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Abstraktní typy

- atribut **abstract="true"** definuje prvek jako abstraktní – nepoužívá se přímo, musí být vždy zastoupen jiným
- příklad:

```
<xsd:element name="jm-vzor" type="xsd:string"
        abstract="true"/>
<xsd:element name="jmeno" type="xsd:string"
        substitutionGroup="jm-vzor"/>
<xsd:element name="prijmeni" type="xsd:string"
        substitutionGroup="jm-vzor"/>
```

XML Schema 1.1

XML Schema 1.1

- přijato 2012
- přidána validační pravidla (inspirovaná Schematronem)
- odstraněna řada omezení pro <all>
- <redefine> zavrženo, nahrazeno obecnějším <override>
- nové typy
- řada dalších rozšíření

Validační pravidla

- lze přidat k definici typu a určit **dodatečná omezení pro jeho hodnoty** – typicky jejich vzájemné vztahy
- **<assert>** pro složené typy, **<assertion>** pro jednoduché
- atribut test obsahuje pravdivostní XPath výraz (bude později); pokud je vyhodnocen jako nepravdivý, je dokument nevalidní – stejně jako při jakémkoli jiném porušení definice jazyka

Příklad: Interval

```
<xsd:element name="interval">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="from" type="xsd:decimal"/>
            <xsd:element name="to" type="xsd:decimal"/>
        </xsd:sequence>
        <xsd:assert test="from le to"/>
    </xsd:complextype>
</xsd:element>
```

hodnota ve <from> musí být menší nebo rovna hodnotě <to>