PRG036 – Technologie XML

Přednáší:

Irena Mlýnková (mlynkova@ksi.mff.cuni.cz)

Martin Nečaský (necasky@ksi.mff.cuni.cz)

LS 2011

Stránka přednášky:
http://www.ksi.mff.cuni.cz/~mlynkova/prg036/

Osnova předmětu

- Úvod do principů formátu XML, přehled XML technologií, jazyk DTD
- Datové modely XML, rozhraní DOM a SAX
- Úvod do jazyka XPath
- Úvod do jazyka XSLT
- XPath 2.0, XSLT 2.0
- Úvod do jazyka XML Schema
- Pokročilé rysy jazyka XML Schema
- Přehled standardních XML formátů
- Úvod do jazyka XQuery
- Pokročilé rysy jazyka XQuery, XQuery Update
- Úvod do XML databází, nativní XML databáze, číslovací schémata, structural join
- Relační databáze s XML rozšířením, SQL/XML

Úvod

- XML data – obecně elementy + atributy + data + speciální prvky
  - Komentáře, CDATA, instrukce pro zpracování apod.
- Správně (semi)strukturovaný / zformovaný (well-formed) XML dokument
  - XML data jsou správně uzávorkována
- Správně strukturovaný / validní / platný (valid) XML dokument
  - XML data navíc odpovídají danému modelu (XML schématu)

Definice schématu XML dat

- Schéma XML dat
  - Popis přípustné struktury XML dokumentů
  - Popis elementů a atributů, které se smí v dokumentu vyskytovat a jejich vzájemných vztahů
- Nástroje pro definici struktury:
  - DTD (Document Type Definition) – už známe
  - XML Schema – návrh W3C konsorcia
  - Schematron, RELAX, …
- Nástroje pro kontrolu validity XML dokumentů (tzv. XML validátory, XML procesory, …)

Zmatky v terminologii

- XML schéma = přípustná struktura XML dat popsaná v některém z existujících jazyků
  - DTD, XML Schema, Relax NG, Schematron, …
- XML Schema = jeden z jazyků pro definici přípustné struktury
  - „XML schéma v jazyce XML Schema“
- Existují i další varianty: XML-schema, XML-Schema, XML-schéma, XML schema, …
- V angličtině: XML schema vs. XML Schema

DTD – připomenutí

- Stále nejpoužívanější jazyk
- Důvody:
  - Jednoduchý na pochopení, použití i zpracování
  - Postačující vyjadřovací síla (?)
    - pro jednoduché aplikace
    - Extrémisti: „Proč dělat schéma? Pak ho budu muset dodržovat…“
- Problém:
  - Aplikace jsou stále složitější
  - Rostou požadavky na přesnější specifikaci přípustné struktury
  - DTD přestává stačit

V DTD lze definovat

- Elementy a atributy
- Vztahy element-podelement a element-atribut
- Přípustný obsah elementů (prázdný, textový, elementový, smíšený, ...)
- Pořadí (, |) a počet výskytů (? + *) elementů v rámci nadelementu
- (V omezené míře) datové typy, povinnost výskytu a implicitní hodnoty atributů

DTD – příklad

```
<?xml version="1.0" encoding="windows-1250"?>
<!ELEMENT zaměstnanci (osoba)+>
<!ELEMENT osoba (jméno, email*, vztahy?)>
    <!ATTLIST osoba id ID #REQUIRED>
    <!ATTLIST osoba poznámka CDATA #IMPLIED>
    <!ATTLIST osoba dovolená (ano|ne) "ne">
<!ELEMENT jméno ((křestní, příjmení)|(příjmení, křestní))>
<!ELEMENT křestní (#PCDATA)>
<!ELEMENT příjmení (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT vztahy EMPTY>
    <!ATTLIST vztahy nadřízený IDREF #IMPLIED>
    <!ATTLIST vztahy podřízení IDREFS #IMPLIED>
```

DTD – příklad


```
<?xml version="1.0" encoding="windows-1250"?>
<!ELEMENT zaměstnanci (osoba)+>
<!ELEMENT osoba (jméno, email*, vztahy?)>
    <!ATTLIST osoba id ID #REQUIRED>
    <!ATTLIST osoba poznámka CDATA #IMPLIED>
    <!ATTLIST osoba dovolená (ano|ne) "ne">
<!ELEMENT jméno ((křestní, příjmení)|(příjmení, křestní))>
<!ELEMENT křestní (#PCDATA)>
<!ELEMENT příjmení (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT vztahy EMPTY>
    <!ATTLIST vztahy nadřízený IDREF #IMPLIED>
    <!ATTLIST vztahy podřízení IDREFS #IMPLIED>
```

XML Schema – přínosy (1)

- Nevyžaduje speciální syntaxi
  - XML schémata v jazyce XML Schema = XML dokumenty
  - Není nutné umět dva různé jazyky, psát / mít parser dvou různých jazyků apod.
  - Pro zpracování lze využít všech nástrojů pro práci s XML dokumenty (např. DOM, SAX, dotazovací jazyky…)
- Silná podpora datových typů
  - Sada vestavěných typů (např. boolean, date, ...)
  - Možnost definovat vlastní typy, vícehodnotové datové typy (tj. pole daného typu)

XML Schema – přínosy (2)

- Lze přesně vyjádřit přípustné počty výskytů elementů v rámci nadelementu

```
<!ELEMENT osoba (jméno, email, email?, email?, email?,
email?, vztahy?)>
```

- Lze definovat elementy se stejným názvem, ale různým obsahem
  - V DTD jsou všechny elementy definovány na stejné úrovni
- Lze definovat elementy s prázdným obsahem / elementy, které se mohou vyskytovat bez obsahu

XML Schema – přínosy (3)

- Lze (snadno) vyjádřit libovolné pořadí elementů v rámci nadelementu

```
<!ELEMENT jméno ((křestní, příjmení)|(příjmení, křestní))>
```

- Při modelování je možné opakovaně využívat již definované prvky
  - Elementy lze i v DTD
  - Datové typy, množiny elementů, množiny atributů, ...
- Mnoho objektově-orientovaných prvků
  - Dědičnost, substituovatelnost apod.
  - Pro modelování přirozené

XML Schema – přínosy (4)

- Umožňuje specifikovat unikátnost obsahu elementu, hodnoty atributu nebo jejich kombinace v rámci požadované části XML dokumentu
  - V DTD lze specifikovat pouze unikátnost hodnot atributů v rámci celého XML dokumentu
- Lze definovat tutéž věc několika způsoby
  - „Syntaktický cukr“ – příjemné pro uživatele
- Zachovává prvky jazyka DTD
  - Téměř všechny prvky jsou zachovány nebo mají ekvivalentní nebo silnější náhradu

XML Schema – přínosy = nevýhody

- Nevyžaduje speciální syntaxi
  - Jazyk je příliš „upovídaný“
  - Schémata jsou dlouhá a méně přehledná než DTD
  - Složitější schéma je „nepochopitelné“
  - Komplikovaný popis principu
    - Elementy a atributy definujeme opět pomocí elementů a atributů
- Lze definovat tutéž věc několika způsoby
  - Výhodné pro definici schématu
  - Nevýhodné pro zpracování

zam.xsd

xhtml.xsd

XML Schema – specifikace (1)

- Doporučení konsorcia W3C
- Historie:
  - 1999 – první working draft (pracovní verze)
  - květen 2001 – working draft => recommendation (doporučení = oficiální specifikace)
    - verze 1.0
  - říjen 2004 – vychází Second Edition verze 1.0
    - Především oprava chyb
    - Stávající oficiální specifikace
  - červenec 2004 – working draft verze 1.1

XML Schema – specifikace (2)

- Verze 1.0:
  - Part 0: Primer http://www.w3.org/TR/xmlschema-0/
    - Není přímo normou, sada ukázkových příkladů a jejich popis
  - Part 1: Structures http://www.w3.org/TR/xmlschema-1/
    - Popis struktur jazyka
  - Part 2: Datatypes http://www.w3.org/TR/xmlschema-2/
    - Popis vestavěných datových typů jazyka
- Verze 1.1:
  - Part 1: Structures
    - http://www.w3.org/TR/xmlschema11-1/
  - Part 2: Datatypes
    - http://www.w3.org/TR/xmlschema11-2/

XML Schema – specifikace (3)

„XML Schema 1.1 is intended to be mostly compatible with XML Schema 1.0 and to have approximately the same scope, but also to fix bugs and make whatever improvements we can, consistent with the constraints on scope and compatibility.“

W3C

XML Schema – základy (1)

- XML schéma v jazyce XML Schema = správně zformovaný a validní XML dokument
  - XML deklarace, kořenový element (schema), správné uzávorkování, validita

```
<?xml version="1.0" encoding="windows-1250"?>
<schema ...>
  ... <!-- definice XML schématu --> ...
</schema>
```

- Prvky jazyka = elementy
  - Vlastnosti – podelementy / atributy
  - Definovány ve jmenném prostoru jazyka XML Schema

XML Schema – základy (2)

- Jmenné prostory:
  - Jmenný prostor jazyka XML Schema
  - Cílový jmenný prostor (targetNamespace)
  - Implicitní jmenný prostor

```
<?xml version="1.0" encoding="windows-1250"?>
<xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.mff.cuni.cz/MojeSchema"
      xmlns="http://www.mff.cuni.cz/MojeSchema">
  ... <!-- definice XML schématu --> ...
</xs:schema>
```

Připojení schématu k dokumentu (1)

- Jmenný prostor instancí XML schémat (tj. XML dokumentů)
- Jmenný prostor schématu XML dokumentu + URL souboru s XML schématem (schemaLocation)
- Implicitní jmenný prostor

```
<?xml version="1.0" encoding="windows-1250"?>
<kořenový_element
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
          "http://www.mff.cuni.cz/MojeSchema schema1.xsd"
    xmlns="http://www.mff.cuni.cz/MojeSchema">
  ... <!-- XML dokument --> ...
</kořenový_element>
```

Připojení schématu k dokumentu (2)

- XML schéma nemusí mít cílový jmenný prostor
  - Tj. specifikován atribut targetNamespace elementu schema
  - (Aktuální prostor je implicitním => není třeba definovat implicitní.)
- Jmenný prostor instancí XML schémat
- URL souboru s XML schématem (noNamespaceSchemaLocation)

```
<?xml version="1.0" encoding="windows-1250"?>
<kořenový_element
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="schema2.xsd">
  ... <!-- XML dokument --> ...
</kořenový_element>
```

Kořenový element?

- Kořenovým elementem může být libovolný globálně definovaný element
  - Definován na první úrovni, tj. jeho definice je přímým podelementem elementu schema
- Globálně definované prvky mají v XML Schema speciální významy
  - Elementy, atributy, (jednoduché i složené) datové typy, skupiny elementů, skupiny atributů
  - Elementy => kořenové elementy, substituce, ...
  - Všechny => lze opakovaně využívat, ... (viz dále)

Kořenové elementy – příklad (1)

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

  <xs:element name="osoba">
    <!-- definice obsahu -->
  </xs:element>

  <!-- definice delších elementů -->
</xs:schema>
```

Kořenové elementy – příklad (2)

```
<?xml version="1.0" encoding="windows-1250"?>
<zamestnanci
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="zam.xsd">
  <!-- obsah elementu -->
</zaměstnanci>
```

```
<?xml version="1.0" encoding="windows-1250"?>
<osoba
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="zam.xsd">
  <!-- obsah elementu -->
</osoba>
```

Filozofie práce s XML Schema

- Princip definice XML schématu:
  - Definice datových typů
  - Definice elementů a atributů (název + datový typ)
    - Hierarchie je dána pozicí ve schématu, popř. referencemi
- Prvky jazyka:
  - Základní – jednoduchý datový typ, složený datový typ, element, atribut, skupina elementů, skupina atributů
  - Pokročilé – omezení identity, substituční skupiny, zástupci, externí schémata, …
    - viz příště
- „Stavebnice“ – sestavování složitějších prvků z jednoduchých, odvozování, odkazování, …

Jednoduché datové typy

Úmluva: Nadále bude v příkladech vynechávána XML deklarace i element schema.

- Obsahem elementů i hodnotami atributů je vždy text
- Jednoduchý datový typ = omezení textového řetězce na množinu přípustných hodnot
- Druhy:
  - Vestavěné – předdefinované
    - viz Specifikace Part 2: Datatypes
  - Uživatelsky definované – specifikované uživatelem prostřednictvím odvozování

Vestavěné jednoduché typy

- Součástí jazyka XML Schema
- Další dělení:
  - Základní – např. string, boolean, decimal, float, date, time, … (19 typů)
  - Odvozené
    - Od typu string – řetězcové typy, např. token, language, ID, IDREF, … (12 typů)
    - Od typu decimal – číselné typy, např. integer, long, int, short, byte, positiveInteger, … (13 typů)
- Zahrnují všechny datové typy z DTD
- Příště podrobný přehled
  - Verze 1.1: Přibylo několik nových typů

Uživatelsky definované jednoduché typy

- Možnost definice vlastního jednoduchého typu
  - viz příklad
- Element simpleType – atributy:
  - name – jméno jednoduchého typu
  - final – zákaz dalšího odvozování
    - restriction, union, list, #all
- Odvozením z jiného (vestavěného / uživatelsky definovaného)
  - Restrikcí (restriction)
  - Seznamem (union)
  - Sjednocením (list)

Odvození restrikcí (1) – příklad

```
<xs:simpleType name="Porty">
  <xs:restriction base="xs:integer">
    <xs:enumeration value="111"/>
    <xs:enumeration value="21"/>
    <xs:enumeration value="80"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="NeprázdnýŘetězec" final="#all">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
```

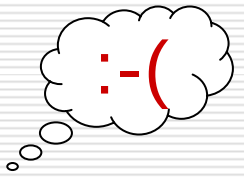
Odvození restrikcí (2) – příklad

```
<xs:element name="Číslo_portu" type="Porty"/>

<xs:element name="Název_serveru" type="NeprázdnýŘetězec"/>
```

```
<Číslo_portu>111</Číslo_portu>

<Název_serveru>kocour</Název_serveru>
```




```
<Číslo_portu>112</Číslo_portu>
<Číslo_portu>ahoj</Číslo_portu>

<Název_serveru/>
<Název_serveru>kocour.ms.mff.cuni.cz</Název_serveru>
```

Odvození restrikcí (3)

- Omezení hodnot původního typu dle daného pravidla
  - length, pattern, enumeration, maxInclusive, minInclusive, maxExclusive, minExclusive, totalDigits, fractionDigits, ...
  - Verze 1.1: minScale, maxScale
    - Min a max exponent při vyjádření ve formě mantisy a exponentu, pouze pro nově definovaný datový typ precisionDecimal
- Odvození musí pro původní typ „dávat smysl“, tj. není povoleno vše

Odvození seznamem (1)

- Seznam hodnot původního typu oddělených bílými znaky
  - Problém: seznam řetězcových typů vs. oddělovač bílý znak
- Vícehodnotové datové typy
  - Nelze odvozovat z jiných vícehodnotových typů
  - NMTOKENS, IDREFS, ENTITIES

Odvození seznamem (2) – příklad

```
<xs:simpleType name="SeznamReálnýchČísel">
  <xs:list itemType="xs:float"/>
</xs:simpleType>

<xs:element name="Teploty"
            type="SeznamReálnýchČísel"/>
```

```
<Teploty>11 12.5 10.2</Teploty>
<Teploty>-3.14 0 -1.5</Teploty>
```

Odvození sjednocením

- Sjednocení přípustných hodnot všech sjednocovaných typů

```
<xs:simpleType name="NenulováCeláČísla">
  <xs:union memberTypes="xs:positiveInteger
                         xs:negativeInteger"/>
</xs:simpleType>

<xs:element name="Teplota" type="NenulováCeláČísla"/>
```

```
<Teplota>11</Teplota>
<Teplota>-3</Teplota>
<Teplota>10</Teplota>
```

Globálně vs. lokálně definované typy

```
<xs:simpleType name="TypNulaAž100">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="100"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="nula"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

Atributy (1)

- Název + jednoduchý datový typ
  - Vestavěný – hodnotou atributu type
  - Globálně definovaný – hodnotou atributu type
  - Lokálně definovaný – jako podelement

Atributy (2) – příklad

```
<xs:attribute name="Věk" type="xs:positiveInteger"/>

<xs:attribute name="Název" type="NeprázdnýŘetězec"/>

<xs:attribute name="TelefonníČíslo">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{3}-\d{6}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

```
<Osoba Věk="30"
       Jméno="H. Simpson"
       TelefonníČíslo="123-445566"/>
```

Atributy (3)

- Element attribute – atributy:
  - default – implicitní hodnota atributu není-li v XML dokumentu uveden
  - fixed – konstantní hodnota atributu
  - use – povinnost výskytu atributu
    - optional, required, prohibited (?)
- Atribut lze také definovat globálně / lokálně
  - V obou případech je pojmenován
  - Globálně – element attribute je podelementem elementu schema, lze se na něj odkazovat referencí
  - Lokálně – v rámci definice složeného typu nebo skupiny atributů, pouze lokální použití

Elementy (1)

- Název + jednoduchý / složený datový typ
  - Jednoduchý typ – element bez atributů pouze s textovým (jednoduchým) obsahem
    - Vestavěný – hodnotou atributu type
    - Globálně definovaný – hodnotou atributu type
    - Lokálně definovaný – jako podelement
  - Složený typ – ostatní varianty elementů

Elementy (2) – příklad

```
<xs:element name="Jméno" type="xs:string"/>

<xs:element name="Příjmení">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="2"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<Jméno>Marge</Jméno>
<Příjmení>Simpson</Příjmení>
```

Elementy (3)

- Element element – atributy:
  - nillable – příznak možného prázdného obsahu
  - default – implicitní hodnota elementu, je-li uveden bez obsahu (textový obsah)
  - fixed – konstantní hodnota elementu (textový obsah)
- Lze také definovat globálně / lokálně
  - V obou případech je pojmenován
  - Globálně – element element je podelementem elementu schema, lze se na něj odkazovat
    - Kořenové elementy XML dokumentů
  - Lokálně – v rámci definice složeného typu, lokální použití

Složené datové typy (1)

- Pro definici složitějších typů elementů
  - Vztahy element-podelement a element-atribut
  - Počty a pořadí elementů v rámci nadelementu
  - Verze 1.1: Možnost specifikace podmínek pro hodnoty podelementů / atributů (jazyk XPath)
- Obsahuje (viz příklad):
  - Specifikaci obsahu složeného typu
    - prázdná = složený typ elementu bez obsahu
  - Specifikaci množiny atributů
    - prázdná = složený typ elementu bez atributů
- Element complexType – atributy:
  - mixed – příznak smíšeného obsahu

Složené datové typy (2) – příklad

```
<xs:complexType name="TypAdresa">
 <!-- Specifikace obsahu -->
  <xs:sequence>
    <xs:element name="Ulice" type="xs:string"/>
    <xs:element name="ČDomu" type="xs:integer"/>
    <xs:element name="Město" type="xs:string"/>
  </xs:sequence>

  <!-- Specifikace množiny atributů -->
  <xs:attribute name="Země"  type="xs:NMTOKEN"
                             default="CZ"/>
</xs:complexType>

<xs:element name="Adresa" type="TypAdresa"/>
```

Složené datové typy (3) – příklad

```
<Adresa>
  <Ulice>Malostranské nám.</Ulice>
  <ČDomu>25</ČDomu>
  <Město>Praha 1</Město>
</Adresa>
```

```
<Adresa Země="SK">
  <Ulice>Šafárikovo nám.</Ulice>
  <ČDomu>6</ČDomu>
  <Město>Bratislava 16</Město>
</Adresa>
```

Složené datové typy (4)

- Lze také definovat globálně / lokálně
  - Podobně jako u jednoduchých typů
- Typy obsahu složeného typu:
  - s jednoduchým obsahem (simpleContent)
  - posloupnost elementů (sequence)
  - výběr z elementů (choice)
  - množina elementů (all)
  - modelová skupina (group)
  - se složeným obsahem (complexContent)

Jednoduchý obsah (1)

- Obsahem elementu je jednoduchý typ + atributy
- Odvození:
  - Rozšířením (extension)
  - Restrikcí (restriction)

```
<xs:complexType name="Typ">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Podtyp" type="xs:string"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Jednoduchý obsah (2)

- Rozšíření – přidání atributů k jednoduchému typu / složenému typu s jednoduchým obsahem
- Restrikce – stejně jako u jednoduchých typů

```
<xs:complexType name="TypAuta">
  <xs:simpleContent>
    <xs:restriction base="Typ">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="VW"/>
      <xs:enumeration value="BMW"/>
      <xs:attribute name="Podtyp" type="xs:string"/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

Jednoduchý obsah (3)

```
<xs:element name="Dopravní_prostředek" type="Typ"/>
<xs:element name="Auto" type="TypAuta"/>
```

```
<Dopravní_prostředek
Podtyp="čopr">motorka</Dopravní_prostředek>

<Auto Podtyp="TT">Audi</Auto>
```

Posloupnost elementů (1)

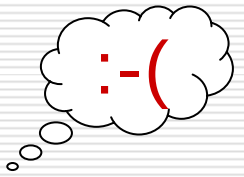
- Obsahem jsou všechny specifikované prvky v daném pořadí

```
<xs:complexType name="Osoba">
  <xs:sequence>
    <xs:element name="Jméno"     type="xs:string"
                                 maxOccurs="5"/>
    <xs:element name="Příjmení" type="xs:string"/>
    <xs:element name="DatumNar" type="xs:date"/>
    <xs:element name="Poznámka" type="xs:string"
                                 minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:ID"/>
</xs:complexType>
```

Posloupnost elementů (2)

```
<xs:element name="Účastník" type="Osoba"/>
```

```
<Účastník Id="1234">
  <Jméno>Karel</Jméno>
  <Příjmení>Němec</Příjmení>
  <DatumNar>1848-07-04</DatumNar>
  <Poznámka>Náčelník</Poznámka>
</Účastník>
```




```
<Účastník Id="4567">
  <DatumNar>1850-12-12</DatumNar>
  <Jméno>Vojtěch</Jméno>
  <Příjmení>Šofr</Příjmení>
</Účastník>
```

Výběr z elementů (1)

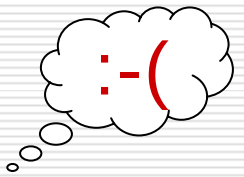
- Obsahem je jeden ze specifikovaných prvků s daným přípustným počtem výskytů

```
<xs:complexType name="TypCeník">
  <xs:choice>
    <xs:sequence>
      <xs:element name="CenaBezDPH" type="xs:string"/>
      <xs:element name="PlnáCena" type="xs:string"/>
    </xs:sequence>
    <xs:element name="SníženáCena" type="xs:string"/>
    <xs:element name="StudentskáCena" type="xs:string"/>
  </xs:choice>
</xs:complexType>
```

Výběr z elementů (2)

```
<xs:element name="Cena" type="TypCeník"/>
```

```
<Cena>
  <CenaBezDPH>1170Kč</CenaBezDPH>
  <PlnáCena>1500Kč</PlnáCena>
</Cena>
<Cena>
  <StudentskáCena>1000Kč</StudentskáCena>
</Cena>
```




```
<Cena>
  <PlnáCena>1500Kč</PlnáCena>
  <CenaBezDPH>1170Kč</CenaBezDPH>
</Cena>
<Cena>
  <PlnáCena>1500Kč</PlnáCena>
</Cena>
```

Množina elementů (1)

- Obsahem jsou specifikované elementy v libovolném pořadí

```
<xs:complexType name="TypKniha">
  <xs:all>
    <xs:element name="Název"  type="xs:string"/>
    <xs:element name="Autor"  type="xs:string"/>
    <xs:element name="ISBN"   type="xs:string"
                              minOccurs="0"/>
  </xs:all>
</xs:complexType>
```

Množina elementů (2)

```
<xs:element name="Kniha" type="TypKniha"/>
```

```
<Kniha>
  <Název>Babička</Název>
  <Autor>Božena Němcová</Autor>
  <ISBN>123-456-789</ISBN>
</Kniha>
<Kniha>
  <Autor>Jára Cimrman</Autor>
  <Název>Dědeček</Název>
</Kniha>
```

Množina elementů (3)

- Verze 1.0: maxOccurs elementů i celé množiny max. 1
  - Chceme maxOccurs > 1?
    - Idea: Kombinace choice a maxOccurs > 1
    - Může vést na nedeterministický datový model
      - Specifikace nepovoluje, ale může existovat parser, který ano
- Verze 1.1: maxOccurs elementů může být > 1
  - Obecně stále není povoleno vše (libovolný nedeterministický model), ale pravidla se uvolnila

Modelová skupina (1)

- Obsahuje posloupnost / množinu / výběr z elementů
- Element group
- Vždy deklarována globálně (a pojmenována)
  - Opakované využití typů obsahů (pomocí referencí)
- Reference obecně:
  - Deklarujeme stejným elementem jako odkazovaný prvek, místo atributu name atribut ref
  - Stejný princip lze použít pro modelové skupiny, elementy, atributy i skupiny atributů (viz dále)
    - V případě elementů a atributů pouze pro globálně definované

Modelová skupina (2)

```
<xs:group name="SpolečnéElementy">
  <xs:sequence>
    <xs:element name="Název" type="xs:string"/>
    <xs:element name="Autor" type="xs:string"/>
    <xs:element name="Datum" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:complexType name="TypKniha">
  <xs:sequence>
    <xs:group   ref="SpolečnéElementy" minOccurs="0"/>
    <xs:element name="ISBN"       type="xs:string"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Modelová skupina (3)

```
<xs:element name="Kniha" type="TypKniha"/>
```

```
<Kniha>
  <Název>Dědeček</Název>
  <Autor>Jára Cimrman</Autor>
  <Datum>2006-02-30</Datum>
  <ISBN>987-654-321</ISBN>
  <Vydavatel>DJC</Vydavatel>
</Kniha>
```

Poznámka: Reference a elementy

```
<xs:element name="Název">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:complexType name="Kniha">
  <xs:sequence>
    <xs:element ref="Název"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Složený obsah (1)

- Odvozování nových typů z již existujících
- Restrikcí (restriction) – nový typ je podmnožinou původního
  - např. omezením počtu výskytů elementů, omezením přípustných hodnot použitých jednoduchých typů, odstraněním atributů (use="prohibited")...
- Rozšířením (extension) – vytvořený typ obsahuje původní i nový typ (tj. jeho prvky) v tomto pořadí
  - Obdoba dědičnosti

Složený obsah (2) – příklad

```
<xs:complexType name="Publikace">
  <xs:sequence>
    <xs:element name="Název" type="xs:string"/>
    <xs:element name="Autor" type="xs:string"
                             maxOccurs="unbounded"/>
    <xs:element name="Vydána" type="xs:gYear"/>
  </xs:sequence>
</xs:complexType>
```

Složený obsah (3) – příklad restrikce

```
<xs:complexType name="PublikaceSJednímAutorem">
  <xs:complexContent>
    <xs:restriction base="Publikace">
      <xs:sequence>
        <xs:element name="Název" type="xs:string"/>
        <xs:element name="Autor" type="xs:string"/>
        <xs:element name="Vydána" type="xs:gYear"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Složený obsah (4) – příklad restrikce

```
<xs:element name="Kniha" type="Publikace"/>
<xs:element name="Kniha1" type="PublikaceSJednímAutorem"/>
```

```
<Kniha>
  <Název>XML technologie</Název>
  <Autor>Irena Mlýnková</Autor>
  <Autor>Jaroslav Pokorný</Autor>
  <Autor>Karel Richta</Autor>
  <Autor>Kamil Toman</Autor>
  <Autor>Vojtěch Toman</Autor>
  <Vydána>2006</Vydána>
</Kniha>
```

```
<Kniha1>
  <Název>Dědeček</Název>
  <Autor>Jára Cimrman</Autor>
  <Vydána>2006</Vydána>
</Kniha1>
```

Složený obsah (5) – příklad rozšíření

```
<xs:complexType name="TypKniha">
  <xs:complexContent>
    <xs:extension base="Publikace">
      <xs:sequence>
        <xs:element name="Vydavatel" type="xs:string"/>
        <xs:element name="ISBN"      type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Složený obsah (6) – příklad rozšíření

```
<xs:element name="Kniha" type="TypKniha"/>
```

```
<Kniha>
  <Název>XML technologie</Název>
  <Autor>Irena Mlýnková</Autor>
  <Autor>Jaroslav Pokorný</Autor>
  <Autor>Karel Richta</Autor>
  <Autor>Kamil Toman</Autor>
  <Autor>Vojtěch Toman</Autor>
  <Vydána>2006</Vydána>
  <Vydavatel>Karolinum</Vydavatel>
  <ISBN>80-246-1272-0</ISBN>
</Kniha>
```

Složený obsah (7)

- Element complexType – atributy:
  - abstract – příznak abstraktního datového typu
    - Nelze přiřadit žádnému elementu
    - Je třeba nejprve odvodit nový typ
  - final – pro daný typ zakazuje další odvozování
    - restriction, extension, #all

Skupina atributů (1)

- Obsahuje množinu atributů
  - Obdoba modelové skupiny pro elementy
- Vždy deklarována globálně (a pojmenována)
  - Opakované využití množiny atributů (opět pomocí referencí)
  - Stejný princip lze použít pro globálně definované atributy
- Element attributeGroup

Skupina atributů (2) – příklad

```
<xs:attributeGroup name="SpolečnéAtributy">
  <xs:attribute name="Vypůjčena" type="xs:boolean"/>
  <xs:attribute name="Id"        type="xs:ID"/>
</xs:attributeGroup>

<xs:complexType name="TypKniha">
  <xs:sequence>
    <xs:element name="Název"     type="xs:string"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
  <xs:attributeGroup ref="SpolečnéAtributy"/>
</xs:complexType>
```

Skupina atributů (3) – příklad

```
<xs:element name="Kniha" type="TypKniha"/>
```

```
<Kniha Vypůjčena="true" Id="1234">
  <Název>XML technologie</Název>
  <Vydavatel>Karolinum</Vydavatel>
</Kniha>
```

Poznámka: Reference a atributy

```
<xs:attribute name="Vypůjčena">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="ano"/>
      <xs:enumeration value="ne"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

<xs:complexType name="Kniha">
  <xs:sequence>
    <xs:element name="Název"     type="xs:string"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
  <xs:attribute ref="Vypůjčena"/>
</xs:complexType>
```

Shrnutí

- Definice XML schématu:
  - Definice jednoduchých + složených datových typů
  - Přiřazení datových typů atributům / elementům
- Opakovaně využívané prvky definovány globálně + využití referencí
  - Elementy, atributy, modelové skupiny a skupiny atributů

Konec…