XML

A související technologie

Doc. Ing. Radek Burget, Ph.D.
burgetr@fit.vutbr.cz

XML – Extensible Markup Language

- Serializační formát vycházející ze starších značkovacích jazyků
- Orientovaný na dokumenty
  - Odlišný způsob zápisu
- Celá řada souvisejících technologií pro zpracování XML

XML dokumenty

- Syntakticky obdobné jako v HTML
- Hierarchické zanořování XML prvků (**elementů**)
  - Každý element má *jméno* a případně *atributy*
  - Jeden **kořenový element**
- Syntaktické rozdíly oproti HTML
  - XML nedefinuje jména ani význam elementů a atributů – vše je dáno konkrétní aplikací
  - Proto všechny značky jsou párové (parser nezná jejich význam)
  - Hodnoty všech atributů musí být v uvozovkách

Příklad XML

```
{
   "firstName":"John",
   "lastName":"Smith",
   "address":{
      "streetAddress":"21 2. street",
      "city":"New York",
      "state":"NY",
      "postalCode":10021
   },
   "phoneNumbers":[
      "212 555-1234",
      "646 555-4567"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
   <address>
      <city>New York</city>
      <postalCode>10021</postalCode>
      <state>NY</state>
      <streetAddress>21 2. street</str
   </address>
   <firstName>John</firstName>
   <lastName>Smith</lastName>
   <phoneNumbers>
      <item>212 555-1234</item>
```

- Záhlaví XML není nutné pro kódování UTF-8
- Navíc kořenový element a elementy pro prvky kolekce

Příklad XML – Alternativně

```
{
   "firstName":"John",
   "lastName":"Smith",
   "address":{
      "streetAddress":"21 2. street",
      "city":"New York",
      "state":"NY",
      "postalCode":10021
   },
   "phoneNumbers":[
      "212 555-1234",
      "646 555-4567"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
   <address
      city="New York"
      postalCode="10021"
      state="NY"
      streetAddress="21 2. street"/>
   <firstName>John</firstName>
   <lastName>Smith</lastName>
   <phoneNumbers>
      <item>212 555-1234</item>
      <item>646 555-4567</item>
```

- Atributy místo elementů
- Prázdný element `<address />`

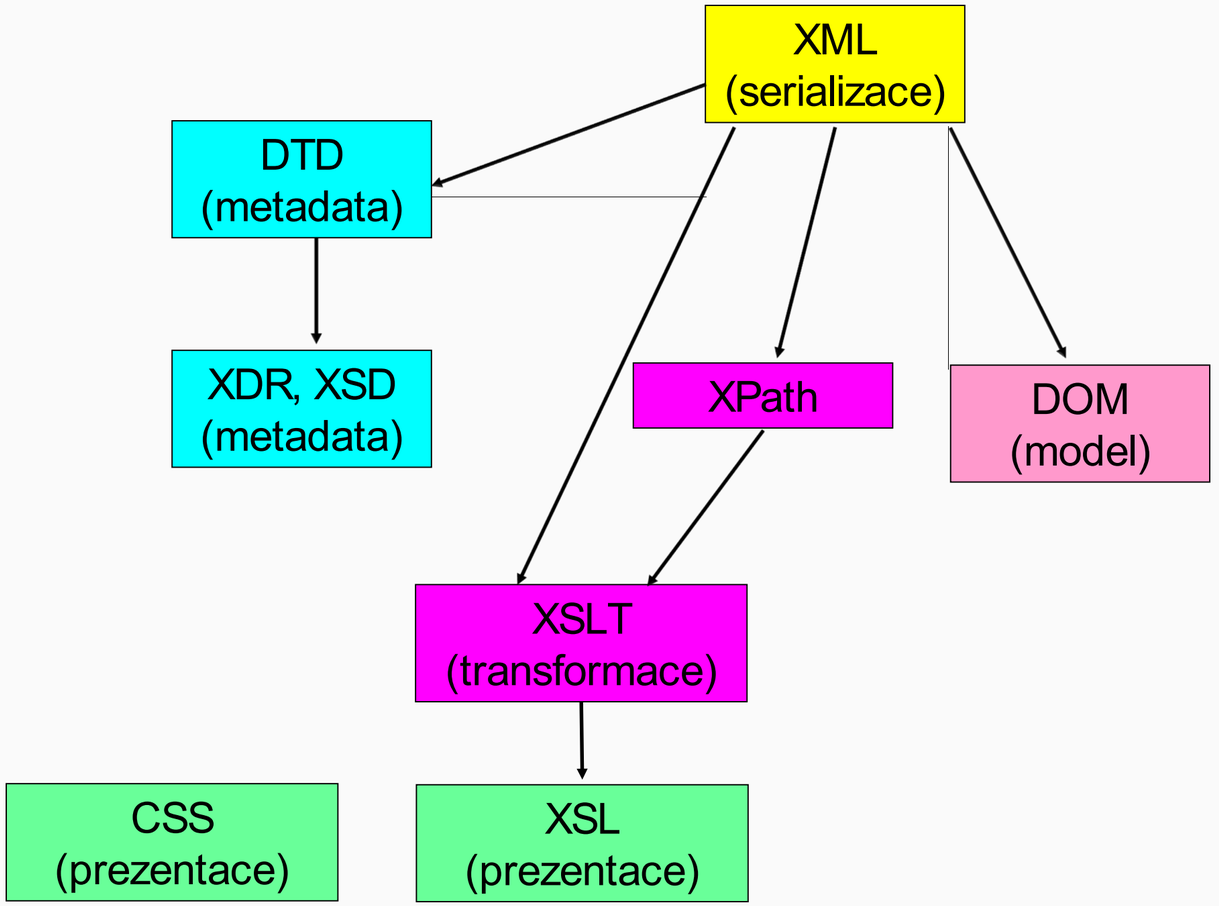
Mixed content

- Element může obsahovat směs vnořených elementů a textu

```xml
<?xml version="1.0" encoding="UTF-8"?>
<person>
   <firstName>John</firstName>
   <lastName>Smith</lastName>
   <phoneNumbers>
      <item>212 555-1234</item>
      <item>646 555-4567</item>
   </phoneNumbers>
   <description>Mr. Smith is <strong>very important</strong> and
        should be given <em>special care</em>.</description>
</person>
```

- Pořadí elementů může být významné (srov. s JSON)

Komponenty XML technologie



XML Schémata

Typová kontrola serializovaných dat

Typová kontrola

- Pro konkrétní aplikaci je možno definovat konkrétní jména značek, atributů, atd.
- Zavedením možnosti ***typového určení a kontroly*** (text, čísla, enumerace apod.) obsahu značek umožňuje
  - ***definovat***
  - ***kontrolovat***
  - ***přenášet*** obecné datové struktury.
- Tím se ze značkovacích jazyků stává ***obecný prostředek pro serializaci strukturovaných dat***

Definice typu dokumentu

- Definice typu dokumentu (DOCTYPE) definuje **schéma** pro danou aplikaci – **metadata**
- Definice DTD (Document Type Definition)
  - Starší „historická“ syntaxe
  - Samotné definice nejsou v jazyce XML
- Definice XSD (XML Schema Definition)
  - Používá XML syntaxi
  - Více možností (např. pořadí prvků, datové typy, …)
- Další alternativy (např. RelaxNG)

Příklad: XML dokument a DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
   <address>
      <city>New York</city>
      <postalCode>10021</postalCode>
      <state>NY</state>
   </address>
   <firstName>John</firstName>
   <lastName>Smith</lastName>
   <phoneNumbers>
      <item>212 555-1234</item>
      <item>646 555-4567</item>
```

```
<!ELEMENT person (address,firstName,la
description?)>
<!ELEMENT address (city,postalCode,sta
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastName (#PCDATA)>
<!ELEMENT phoneNumbers (item)+>
<!ELEMENT city (#PCDATA)>
<!ELEMENT postalCode (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT streetAddress (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT item (#PCDATA)>
```

- (prvek „description“ je nepovinný)

Příklad: Totéž s atributy

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
   <address
      city="New York"
      postalCode="10021"
      state="NY"
      streetAddress="21 2. street"/>
   <firstName>John</firstName>
   <lastName>Smith</lastName>
   <phoneNumbers>
      <item>212 555-1234</item>
      <item>646 555-4567</item>
```

```
<!ELEMENT person (address,firstName,la
description?)>
<!ELEMENT address EMPTY>
<!ATTLIST address
  city CDATA #REQUIRED
  postalCode CDATA #REQUIRED
  state NMTOKEN #REQUIRED
  streetAddress CDATA #REQUIRED>
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastName (#PCDATA)>
<!ELEMENT phoneNumbers (item)+>
<!ELEMENT description (#PCDATA)>
```

- (prvek „description“ je nepovinný)

Typy deklarací

`<!DOCTYPE ...>`	typ dokumentu
`<!- ..... ->`	komentář
`<![CDATA[ ...]]>`	sekce znakových dat
`<!ENTITY .....>`	deklarace entity
`<!ELEMENT .....>`	deklarace elementu
`<!ATTLIST .....>`	deklarace atributu

+ některé další

Deklarace struktury a jejich prvků

Užívá se notace známá z **regulárních výrazů**

`()` závorkové struktury

`?` volitelnost

`+` jeden a více výskytů – je serializací **kolekce**

`*` 0 a více výskytů – je serializací **kolekce**

`,` následnost – je serializací **struktury**

`|` varianta - OR

`&` jeden z možných - AND

Typy atributů

- `CDATA` řetězec znaků
- `NMTOKEN` identifikátor (ale může začínat i číslicí)
- `NMTOKENS` seznam NMTOKEN oddělených prázdnými znaky
- `ENTITY` odkaz na entitu
- `ENTITIES` seznam entit
- `ID` unikátní identifikátor
- `IDREF`, `IDREFS` odkaz na identifikátor
- *výčet* výčet hodnot `(hodnota|hodnota|…)`

Standardní hodnota, resp. druh atributu

- není nezbytná
- `#REQUIRED` povinný atribut
- *hodnota* udává implicitní hodnotu atributu
- `#IMPLIED` (před hodnotou) nepovinný atribut
- `#FIXED` (před hodnotou) je to jediná možná hodnota

Entity

- Možnost fyzicky izolovat a samostatně ukládat libovolnou **pojmenovanou** část dokumentu – **entitu**
- Použití
  - Stejná informace se používá na několika místech
  - Informace může být nekompatibilními systémy reprezentována odlišně
  - Rozsáhlý dokument je vhodné z praktických důvodů členit
  - Informace je v jiném datovém formátu, nežli text XML (multimédia)

Klasifikace entit

- Interní a externí
- Textové a binární
- Dává nám 4 kombinace, přičemž interní binární není možná
- Interní textová, externí textová a externí binární je možná

Deklarace entity

- Entita musí být v textu definována před prvním odkazem
- Při vícenásobné definici se bere první a ostatní se ignorují

```
<!ENTITY jméno entity … >
```

```
<!ENTITY myentity "obsah entity myentity">
```

Odkaz na entitu

- `&jméno entity;`
- Např.

  `Stiskněte klávesu <<<ENTER >>>`

  se zobrazí jako

  `Stiskněte klávesu <<<ENTER>>>`

Interní textové entity

- Zde je textový obsah uzavřen v uvozovkách (nebo apostrofech)

```
<!ENTITY XML "eXtensible Markup Language">
```

```
Formát &XML; obsahuje entity
```

se zobrazí jako

```
Formát eXtensible Markup Language obsahuje entity
```

Vestavěné entity

- není třeba deklarovat, nahrazují metaznaky jazyka XML
- `<` nahrazuje <
- `>` nahrazuje >
- `&` nahrazuje &
- `&apos;` nahrazuje '
- `&quot;` nahrazuje "

Deklarace typu dokumentu

- `<!DOCTYPE jméno [ … ]>`
- musí se objevit před prvním elementem dokumentu
- Je možné použít pouze pro pojmenování a potom se část se závorkami vynechává

XML Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="q
  <xs:element name="person">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element name="lastName" type="xs:string"/>
        <xs:element ref="phoneNumbers"/>
        <xs:element minOccurs="0" name="description" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
```

- Modernější alternativa DTD, používá XML zápis.

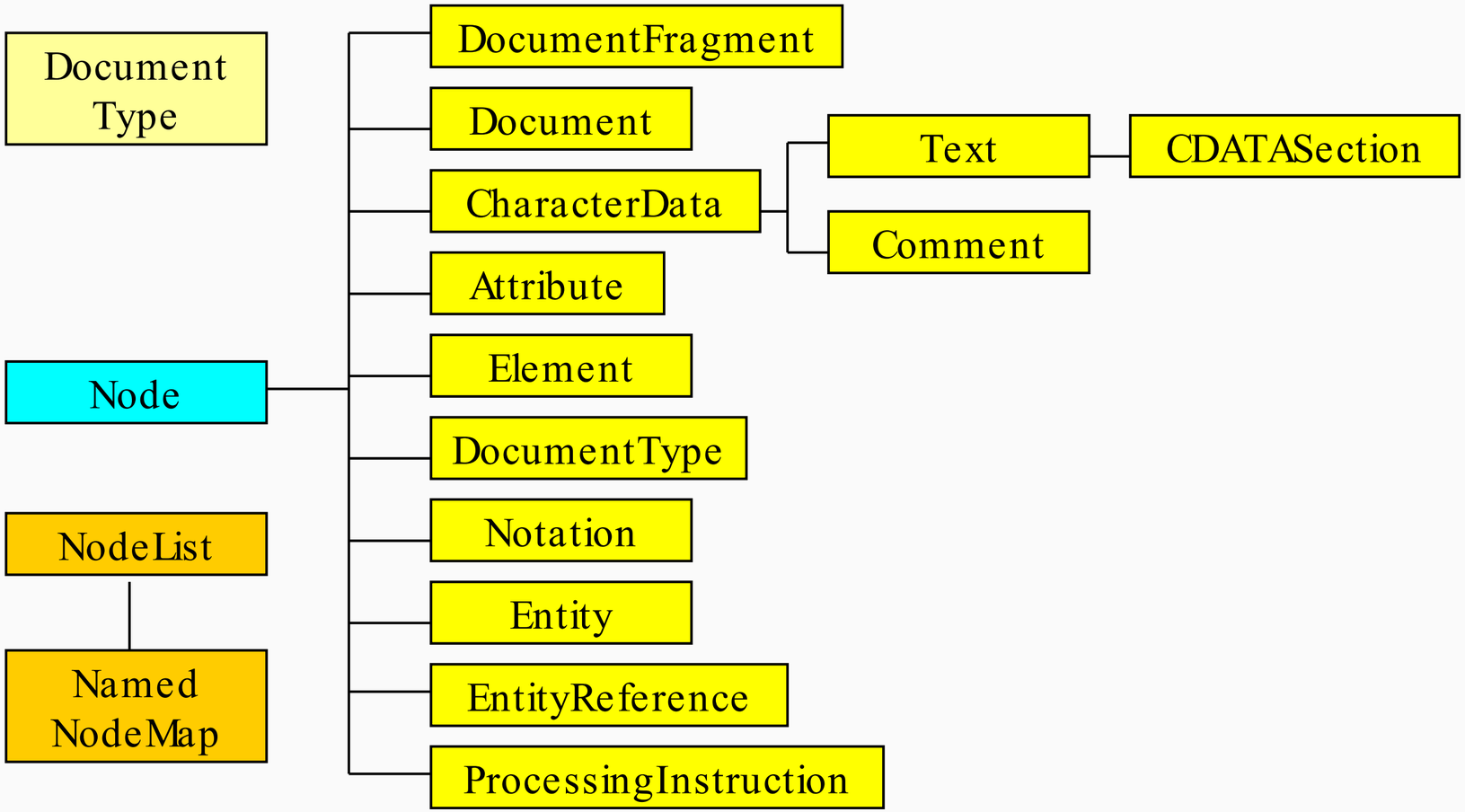
Document Object Model

Reprezentace (nejen) XML v programovacích jazycích

Úvod

- Objektová reprezentace XML (nebo HTML) dokumentu pro práci s dokumentem
  - Čtení, a změna dokumentů
- Dokument je reprezentován jako strom objektů
- W3C standard, definuje rozhraní objektů nezávisle na platformě
  - Implementace: JavaScript, Java, PHP, C++, C#, ...

Třídy DOM


Document Type
Node
NodeList
Named NodeMap
DocumentFragment
Document
CharacterData
Attribute
Element
DocumentType
Notation
Entity
EntityReference
ProcessingInstruction
Text
CDATASection
Comment

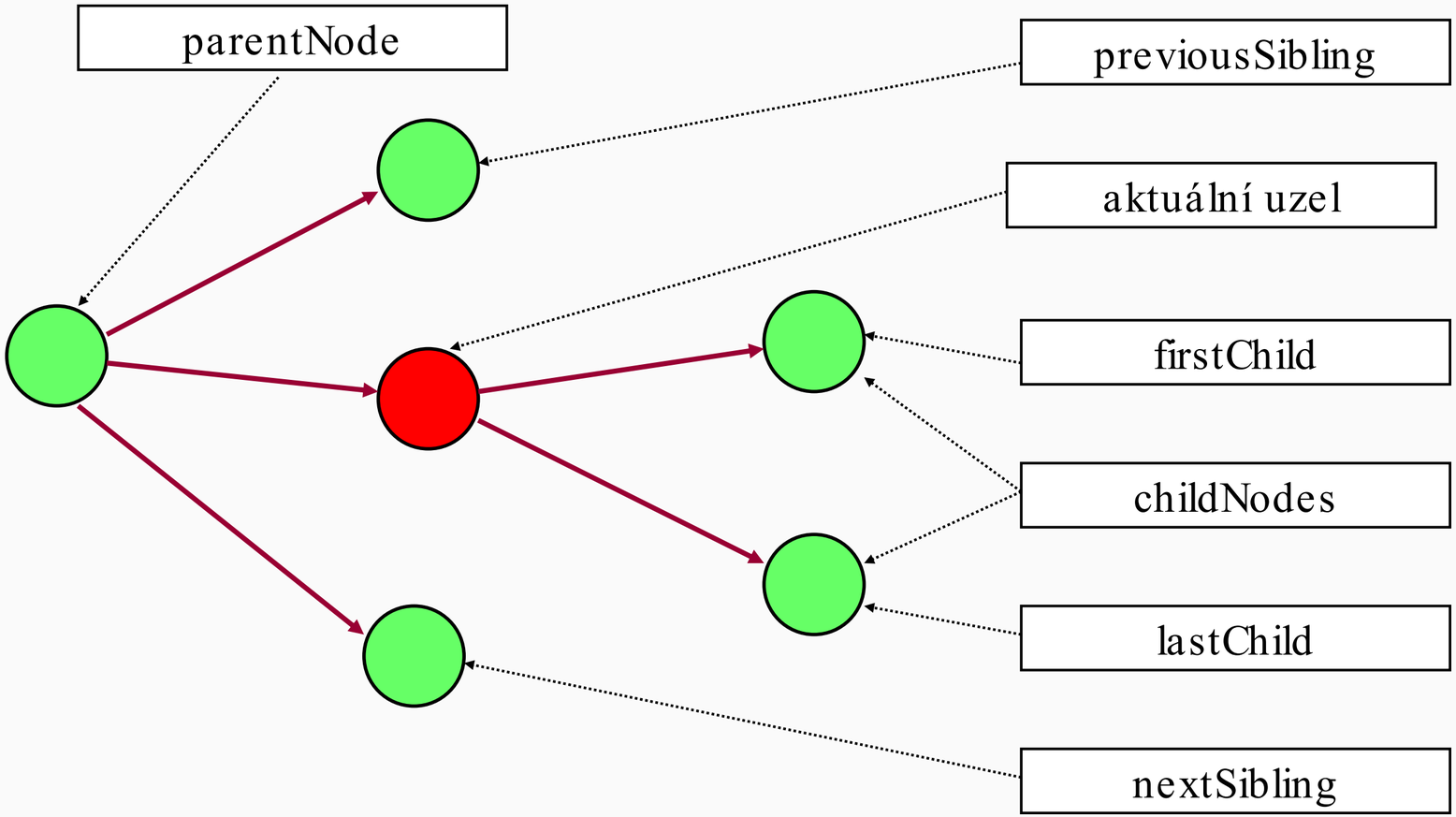
Rozhraní Node

- Je základem modelu a má velký počet následníků
- Tři kategorie metod:
  - určení charakteristik uzlu (jméno, typ, hodnota)
  - určení pozice v dokumentu (navigace), přístup k příbuzným uzlům
  - modifikace obsahu uzlu (manipulace s uzly)

Vlastnosti uzlu

- **nodeType**
- `nodeName`
- `nodeValue`
- `hasChildNodes()`
- `ownerDocument`
- `attributes`
- **textContent**

Příklad navigačních operací



Kolekce uzlů

- má standardní operace:
  - `item(index)`
  - `length`
  - `nextNode()`
  - `reset()`

Rozhraní Document

- Reprezentuje celý dokument
- Vlastnost `documentElement` – kořenový element dokumentu
- Metody pro vyhledání elementů
  - `getElementsByTagName()`
  - `getElementById()`

Rozhraní Element

- Reprezentuje jeden element
- Metody pro vyhledání elementů
  - `getElementsByTagName()`
- Metody pro práci s atributy
  - `getAttribute()`
  - `setAttribute()`

Alternativní rozhraní

- DOM rozhraní je univerzální, ale poněkud těžkopádné v mnoha situacích
- Různé platformy poskytují různá alternativní rozhraní pro práci s XML
- Např v PHP: SimpleXML

Příklady

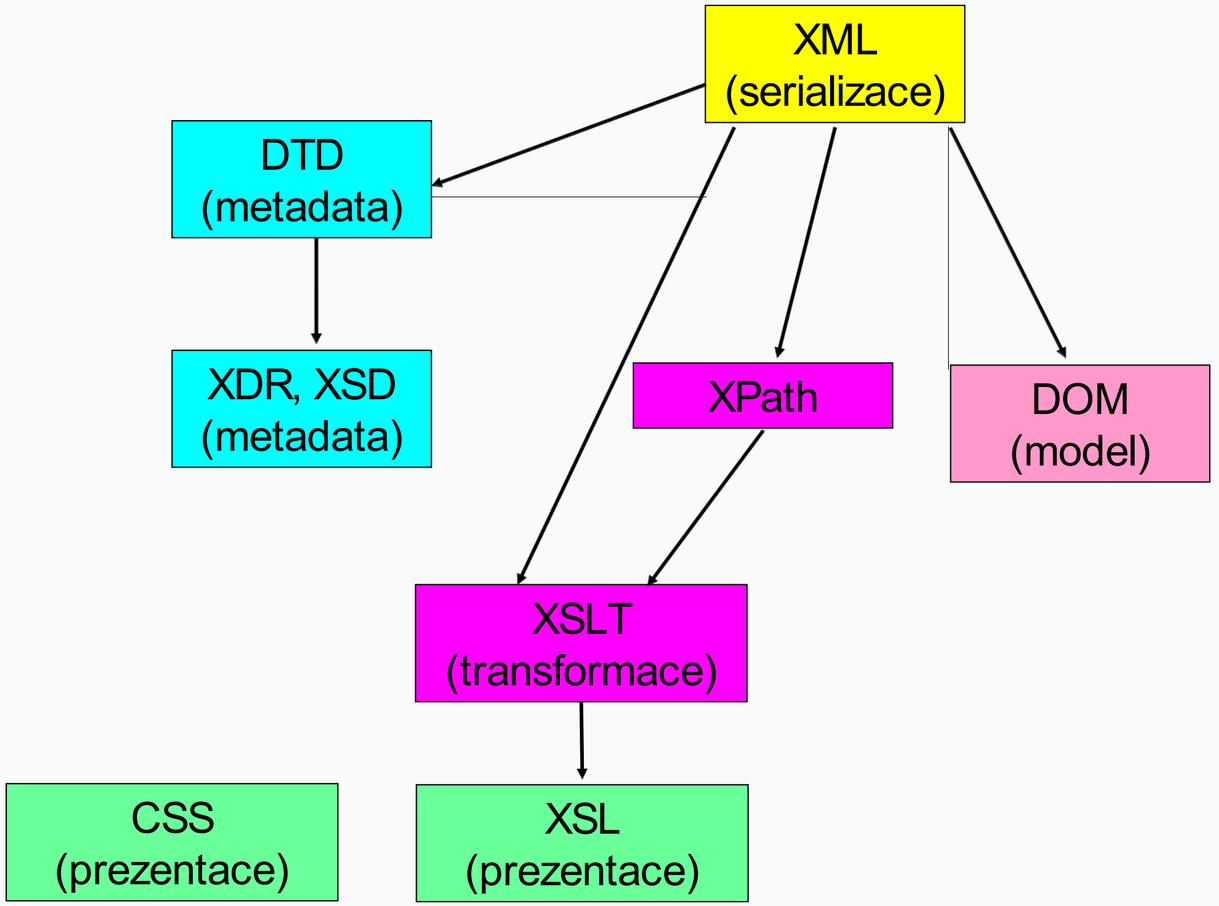
- Příklady na XML v PHP: GitHub

https://github.com/DIFS-Teaching/basic-demos/tree/master/php-xml

XPath a XSLT

Transformace a vizualizace XML dat

Komponenty XML technologie



Vizualizace XML dat

- CSS – Cascading Style Sheets
  - Stejně jako v HTML, jen v selektorech jiná jména elementů
- XSL – eXtensible Style Sheet Language
  - Xpath – adresování elementu („selektory“)
  - XSLT – XSL Transformations (transformace)
  - XSL-FO – formatting objects (popis prezentace)
    - (místo toho se ale spíše transformuje do HTML)
- (nebo formátování DOM např. v PHP)
- Příklady: https://github.com/DIFS-Teaching/basic-demos/tree/master/xml-style

XPath

- Jazyk pro výběr uzlů v XML dokumentu
- Výsledkem vyhodnocení XPath výrazu je obecně **množina elementů**
  - Obdobně jako CSS selektory, ale mocnější
- Předpokládá existenci ‘kurzoru’ (ukazovátka) představujícího aktuální pozici v dokumentu
  - Absolutní i relativní „cesty“ v dokumentu
- Použitelný i na webu
  - Introduction to using XPath in JavaScript (MDN)

Základní tvar výrazu

- `krok1/krok2/krok3/...` - relativní cesta
- `/krok1/krok2/krok3/...` - absolutní cesta
- Výsledkem každého kroku je obecně **množina uzlů DOM**
- Pokud je výsledkem kroku množina uzlů, pak se další krok provádí paralelně ze všech cílů předchozího kroku

Operace pro navigaci

- Slouží k popisu cesty ve stromové struktuře dokumentu
- Mají tvar

  `osa::typ uzlu DOM`
- Osa (axis) v každém kroku určuje, kam se přemístí ukazovátko, může to být na jediný další uzel nebo i na množinu
- Za `::` může být typový výraz omezující cíl pouze na uzly jistého typu

Typový výraz

- Libovolný typ prvku je označen `node()` nebo `*`
- Nebo **jména prvků**
- Další omezení (atributy, pozice, …)

child:: potomek daného uzlu

- potomek daného uzlu se označí `child::`
- jméno funkce child je možné vynechat
- `child::address/child::city`

  je totožné s

  `address/city`

Zástupné znaky

- je možné použít znaku `*` pro libovolný typ uzlu, např. `person/*/city`
- `//` popisují všechny potomky daného jména (ve všech úrovních), např. `people//person`

Další osy

- `descendant`, `descendant-or-self::` – potomci (včetně samotného uzlu)
- `ancestor::`, `ancestor-or-self::` – předek
- `self::` – uzel sám
- `parent::` – rodičovský uzel (totéž co `..`)
- `preceding-sibling::` – předchozí sourozenec
- `following-sibling::` – následující sourozenec

Omezující podmínka

- výsledkem cesty může být celý seznam uzlů
- filtry se používají pro další selekci
- mají tvar booleovské podmínky uzavřené do hranatých závorek
- lze jich uvést i několik za sebou `[…] […] […] […]…`
- lze užívat logické spojky `and`, `or` a `not`
- relační operátory jsou `= != >= <= < >`

Přístup k hodnotě atributu

- `attribute::` nebo zkráceně `@`
- například
  address[@city='Brno'] nebo
  address[attribute::city='Brno']

Určení pozice

- `position()=číslo` pozice, lze zkrátit pouze na číslo
- `person[position()=1]` je totéž jako `person[1]`
- `last()` určuje poslední element: `person[position() = last()]`

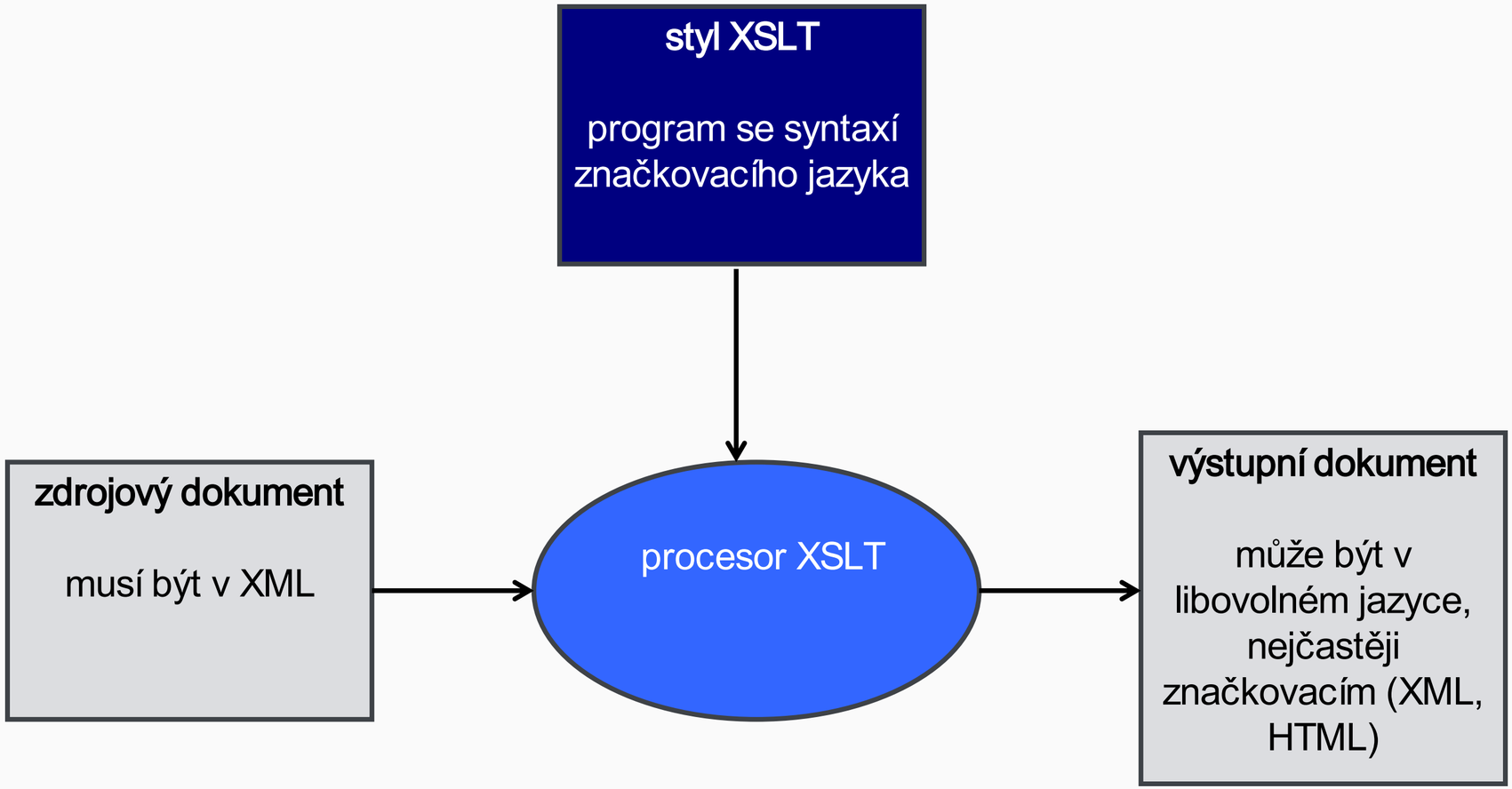
Různé typy výrazů v podmínkách

- Úplný řetězcový a číselný výraz
  - včetně reálných čísel
  - podřetězce, atd.

XSLT

XSL Transformations

Schéma transformace XSLT



Transformace

- XSLT vytváří z jednoho dokumentu obecně jiný dokument (např. XML, který může odpovídat jinému DTD)
- transformace jsou popsány šablonami – ***templates***
- šablony jsou přiřazeny elementům zdroje pomocí výrazů ***XPath***

Příklad

- `xsl:` formát (jazyk XSLT)
- bez prefixu: výstup

```
<xsl:template match="emph">
        <in-seq font-weight="bold">
            <xsl:apply-templates>
        </in-seq>
</xsl:template>
```

Šablony

- tělo definice stylu tvoří nejméně jedna šablona

```
<template match="výraz XPath">
    …
</template>
```

- výrazem se určují uzly DOM, které jsou danou šablonou upravovány

Průchod stromem hierarchie

- V šabloně je nutné explicitně určit, zda se mají zpracovávat i podstromy
  `<apply-templates select="výraz Xpath"/>`
- Je nutné zajistit, aby existovala šablona pro libovolný element `match="*"`, případně pro kořenový element `match="/"`
- Element `<apply-templates/>` je možno použít i vícekrát a potom se podstrom zpracovává vícekrát (asi vždy s jiným select)

Proces transformace

- Udržuje se seznam uzlů ke zpracování
- Na počátku obsahuje pouze kořenový uzel zdrojového stromu DOM
- Zpracováním seznamu zdrojových uzlů se na výstupu vytváří fragment výstupního textu nebo stromu
- Při zpracování je první prvek ze seznamu zdrojových uzlů vyjmut a stává se aktuálním uzlem (*current node*).
- Pro aktuální uzel jsou nalezena všechny šablony, který s ním lze sjednotit (*to match*)

Proces transformace

- Z nich je zvolena (instanciována) ta nejvhodnější (např. podle pořadí nebo priority).
- Výsledek instanciace zkopírován do výstupního textu nebo stromu.
- Šablona může obsahovat instrukce, které umísťují další zdrojové DOM uzly ke zpracování do seznamu, např.

`xsl:apply-templates` (následníci)

`xsl:for-each` (určeni výrazem XPath)

Výstup z transformace

- Nejčastěji se používá vkládání výstupních elementů jazyka HTML (vizualizace XML transformací na HTML)
- Libovolný text vložený do šablony bude rovněž součástí výstupu

value-of

```
<xsl:value-of
  select = "xpath-expression"
  disable-output-escaping = "yes or no"
/>
```

- Přenos zadané hodnoty na výstup
- Totéž pomocí `{xpath-expression}`

Použití transformace pro vizualizaci

- Transformace do HTML (nejběžnější)
- Transformace do XSL-FO
  - Jazyk popisující fyzický vzhled dokumentu (opět XML)
    - Rozměry stránky, jednotlivé bloky, řádky textu, jejich vizuální vlastnosti, …
  - Následně zpracování procesorem FO
    - Výstup do PDF, PostScript, PCL, SVG, …
- Příklad: https://github.com/DIFS-Teaching/basic-demos/tree/master/xml-style

Implementace XSLT

- `xsltproc` (téměř v každém Unixu)
- PHP
  - `XSLTProcessor`
- Java
  - `javax.xml.transform.Transformer`
  - Apache Xalan, Saxon
- C
  - `libxslt`
- **Každý webový prohlížeč**

A to je vše!

Dotazy?