

# XML

Aleš Keprt 29.11.2005

- značkovací jazyk
- pro obecné použití → pro vytváření speciálních značkovacích jazyků

Nejčastější použití:

- výměna dat (při komunikaci)
- na internetu (společně nebo místo HTML)
  
- „Je to podobné jako HTML, ale víc obecné.“
- Vznik 1996-1997, oficiální verze 1.0 únor 1998 (W3C recommendation)
- Současná verze: 1.0 třetí vydání a 1.1 – únor 2004 (lepší unicode podpora)

Vlastnosti XML

- Určeno k reprezentaci dat ve stromové podobě
- Textová podoba (vše je text, binární data jsou trochu problém)
- Obsahem XML je: „Data proložená značkami.“
- Dobře se to edituje – stačí obyčejný textový editor
- Je to platformě nezávislé, dobře přenositelné, pro univerzální použití

Struktura XML souboru

- XML je obecné, umožňuje definovat vlastní jazyky pomocí schémat
- Schema: definuje strukturu XML souboru pro nějaké konkrétní použití
  - Schema může definovat omezení, že <narozeniny> musí obsahovat právě jeden prvek <datum>, ten zas právě jeden prvek <den>, apod.
  - Schema může definovat také datové typy, tj. povolené domény hodnot.
- Srovnej s HTML – tam je pevná struktura tagů, kterou nelze nijak pozměnit
- Někdy se z praktických důvodů používají různé „binární XML“, což jsou nestandardní úpravy XML zmenšující objem dat

## Výhody a nevýhody XML

Pro přenos dat:

- Je to formát zároveň čitelný člověkem i strojem
- Podporuje unicode, tj. všechny světové jazyky i další symboly a značky
- Dokáže reprezentovat nejběžnější datové struktury: záznamy, seznamy, stromy
  - Pozn. Odpovídá v C++ struct/class, list/array/vector, tree ☺
- Je sebedopisný – definuje zároveň jména datových položek, jejich strukturu a hodnoty
- Pevná syntaxe umožňuje jednoduché a efektivní strojové zpracování (parsování)

Jako úložiště dat:

- Je standardizován
- Hierarchická struktura je vhodná pro většinu druhů dat (i když ne všechny)
- XML data, to je obyčejný text – nevztahují se na to žádné patenty apod.
- Je platformě nezávislý, takže přežije i technologické změny počítačů

Nevýhody:

- Syntaxe XML je docela nabobtnalá – snižuje to přehlednost pro člověka a XML soubory jsou taky hodně velké (problém na síti, na kapesních počítačích atp.)
- XML strom může být libovolně hluboko vnořený, což opět zatěžuje slabší počítače s menší pamětí (+ nebezpečí stack overflow atp.)
- XML samo o sobě obsahuje jen textová data – jiné datové typy je nutno definovat zvlášť pomocí schémat, nebo zpracovávat dodatečnou konverzí textu na jiný datový typ
- Využití pro obecnější než stromové struktury je složitější
  - Např. obecné uložení objektů, relační nebo síťové databáze atp.

## Syntaxe XML

Příklad receptu (na vaření) v XML:

```
<?xml version="1.0" encoding="UTF-8"?>

<Recipe name="bread" prep_time="5 mins" cook_time="3 hours">
  <title>Basic bread</title>
  <ingredient amount="3" unit="cups">Flour</ingredient>
  <ingredient amount="0.25" unit="ounce">Yeast</ingredient>
  <ingredient amount="1.5" unit="cups">Warm Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <Instructions>
    <step>Mix all ingredients together, and knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again, place in a tin, and then bake in the oven.</step>
  </Instructions>
</Recipe>
```

1.řádek: XML deklarace – číslo verze a (volitelně) kódování znaků

Další řádky obsahují **elementy**, ty mohou být libovolně (stromově) vnořené a mohou obsahovat **atributy** a/nebo **obsah** (příklady jsou označeny barevným pozadím).

Element se obvykle skládá ze dvou **tagů**. Počáteční tag a koncový tag se liší jen lomítkem – viz příklad. Atributy mohou být připojeny k počátečnímu tagu a mají formát jméno="hodnota", hodnota je vždy v uvozovkách.

Častá chyba: HTML kód často nevyhovuje XML protože tagy nejsou stromově vnořeny.

Například toto není stromová struktura:

```
<p>Normal <em>emphasized <strong>strong emphasized</em> strong</strong></p>
```

Tag <em> tady končí dřív, než jeho vnořený tag <strong>.

HTML také často není XML, neboť některé tagy nejsou párové – to v XML nejde. Ke každému počátečnímu tagu musí být jeho ukončovací tag (a na správném místě). Pokud element neobsahuje žádný obsah, je možno počáteční a koncový tag spojit uvedením lomítka na konci počátečního. Příklad: následující dva řádky jsou v XML identické.

```
<něco></něco>
```

```
<něco/>
```

Pozor! XML dokument může obsahovat jen jeden root element! (Root = kořen, to je element na nejvyšší úrovni. Strom může mít jen jeden kořen.)

Poznámka: Neopakující se položky lze technicky vzato zapsat pomocí atributů i vnořených elementů. Použití atributů vždy vede k podstatně kratšímu kódu, proto je to doporučeno.

## Speciální znaky – escape sekvence

Některé znaky nelze zapsat normálně – buď proto, že je nemáme na klávesnici (třeba znaky cizích národních abeced), nebo proto, že by to kolidovalo s XML syntaxí (třeba levá lomená závorka <). Tyto znaky lze zapisovat pomocí tzv. escape-sequencí.

1. Entity reference
  - odkazuje jménem entity
  - je možné použít pro znaky, ale i pro celé opakující se řetězce
  - zabudované entity reference: &lt; &gt; &quot; &amp; &apos;
2. Character reference
  - odkazuje na jednotlivé znaky podle jejich čísla v unicode
  - př. &#038; je znak & (číslo 38 v Unicode/ASCII)

## Korektnost XML dokumentu

Korektní XML dokument musí splňovat dvě vlastnosti: well-formed, valid

1. Well-formed
  - Vyhovuje XML syntaxi (správně vnořený strom, párové <tagy>, atp.)
  - Parser musí nekorektní XML dokument odmítnout
2. Valid
  - Dokument vyhovuje nějakému konkrétnímu uživatelskému formátu
  - Validita se obvykle kontroluje podle schémat (zmíněno výše)

## Well-formed dokumenty

XML dokument je text, čili posloupnost znaků.

Vyžadována je podpora kódování znaků UTF-8 a UTF-16, ostatní jsou nepovinné

Well-formed dokument musí mj. splňovat tato pravidla:

- Pouze jeden root element (před ním může být hlavička XML dokumentu)
- Neprázdné elementy jsou vždy ukončeny párovým ukončovacím tagem
- Prázdné elementy mohou alternativně mít jednotagový zápis
- Všechny hodnoty atributů jsou v uvozovkách " nebo apostrofech '. Uvozovky musí být ukončeny uvozovkami, apostrof musí být ukončen apostrofem (nelze to míchat).
- Tagy mohou být vnořeny, ale nesmí se překrývat (musí to být strom).
- Dokument musí spadat do kódování znaků určeného v hlavičce. Kódování může být uvedeno v transportním protokolu (tj. mimo XML dokument). Není-li kódování uvedeno, výchozí je UTF-16 a je detekováno podle byte-order značky. Není-li byte-order značka, pak je výchozí kódování UTF-8.
- Jména elementů jsou case-sensitive.

Poznámka: Jména elementů a atributů by měla vystihovat jejich smysl, neboť to zvyšuje přehlednost a srozumitelnost dokumentu pro člověka. Může to ale také způsobovat zvětšení délky dokumentu.

## Validní dokumenty

Validní dokument je ten, který je well-formed a odpovídá nějakému schématu.

XML schéma je popis nějakého konkrétního typu dokumentu

- obvykle obsahuje především omezení toho, který element může/musí být vnořený do kterého, které elementy jsou povinné apod.

XML parsery často umějí kontrolovat validitu dokumentu proti danému schématu.

# Jazyky schémat

## DTD (Document Type Definition)

Nejstarší jazyk pro definici schémat. Je široce podporován, má ale jistá omezení:

- Nepodporuje jmenné prostory (namespace)
- Některé věci v něm prostě nejdou definovat ☺
- Samotný DTD není XML jazykem

## XSD (XML Schema Definition)

Oficiální nástupce DTD.

- Používá systém datových typů
- Umožňuje definovat přesnější pravidla pro strukturu XML dokumentů
- Samotný XSD je také XML jazykem

Existují samozřejmě i další jazyky schémat.

Poznámka: DTD i XSD umějí definovat výchozí hodnoty pro neuvedené atributy. Někdy se to může hodit. ☺

## Zobrazování XML na webu

XML soubory se zobrazují jako obyčejné textové soubory. Chceme-li je používat místo HTML, můžeme pomocí jazyků CSS nebo XSL definovat, jak má XML soubor vypadat. Čili definujeme, jak se má který element zobrazit.

CSS (Cascading Style Sheets) je starší záležitost, používá se obvykle pro upřesnění vzhledu klasických HTML souborů. Jazyk CSS není XML jazykem, ale existují na to dobré editory. Použití CSS spolu s XML dává podobné výsledky jako použití CSS s HTML.

```
<?xml-stylesheet type="text/css" href="styl.css"?>
```

XSL (Extensible Stylesheet Language) / XSLT (XSL Transformation) je věc vytvořená přímo pro XML – umožňuje definovat transformaci XML souboru do jiného formátu. Tentokrát transformujeme XML do HTML – a to doslova! Čili výsledek XSL transformace lze uložit jako HTML soubor.

```
<?xml-stylesheet type="text/xsl" href="styl.xsl"?>
```

XSL je lepší než CSS a obojí lze dělat přímo na klientském počítači v HTML prohlížeči.. ☺

## Programové zpracování XML dokumentů

Nikdy nepište vlastní parser! Použijte nějakou hotovou knihovnu.

W3C definuje dva základní standardy: DOM a SAX. Obojí je platformě nezávislé.

### DOM (Document Object Model)

- Zpřístupňuje XML dokument ve formě objektů (v objektově–orientovaném jazyce).
- Umí dokumenty načítat, měnit i ukládat
- S jakýmkoliv well–formed XML dokumentem lze pracovat pomocí DOM
- Celý dokument je vždy uložen ve formě objektového stromu v paměti
- U velkých dokumentů velmi zatěžuje paměť (a je to i pomalé)

## SAX (Serial Access Parser API / Simple API for XML)

- Alternativa k DOM
- Umí XML dokumenty pouze číst (neumí měnit či zapisovat)
- Nepřechovává celý dokument v paměti → vhodné pro velké dokumenty
- Dokument čte vždy celý postupně sekvenčně
- Načtení XML dokumentu pomocí SAX je obvykle rychlejší než pomocí DOM

DOM i SAX lze používat ve Visual C++ přes MSXML.

<http://msdn.microsoft.com/XML/>

## XML v .NET Frameworku

Dotnet obsahuje knihovnu pro práci s XML v jmenném prostoru System.Xml.

Podporované prvky jsou:

- XML 1.0
- XML Namespaces
- DTD
- XSD Schemas
- XPath výrazy
- XSL transformace
- DOM Level 1 Core, DOM Level 2 Core

## Příklad

```
<?xml version="1.0" encoding="Windows-1250"?>

<tournament>

  <name>Testovací turnaj</name>
  <id>tt</id>
  <placepts>4 3 2 1</placepts>

  <players>
    <player number="1" id="ak" name="Bla Blabla" />
    <player number="3" id="dk" name="Asd Asdasd" />
    <player number="4" id="is" name="Iii Iiiiiiiiiiii" />
  </players>

  ...

</tournament>
```

## Načtení XML souboru do DOM stromu

```
XmlDocument doc = new XmlDocument();
doc.Load("soubor.xml");
```

## Načtení hráčů do objektů pomocí Xpath výrazu

```
foreach(XmlNode node in
doc.DocumentElement.SelectNodes("players/player")) {
```

```
Player p = new Player(  
    int.Parse(node.Attributes["number"].Value),  
    node.Attributes["id"].Value,  
    node.Attributes["name"].Value);  
players.Add(p);  
}
```

## Literatura

- Existuje tisíc knih o XML (hlavně volte nějaká dobrá vydavatelství – Microsoft Press, O'Reilly apod.)
- Na webu je i tisíc XML tutoriálů – na Googlu zadáte XML → 679 milionů odkazů ☺
- Všechny pojmy jsou vysvětleny ve Wikipedii
- Všechny standardy jsou srozumitelně popsány na [www.w3.org](http://www.w3.org)
- Stránka Microsoftu: <http://msdn.microsoft.com/XML/>
- Použijte nápovědu MSDN u Visual Studia ☺

---

© Mgr. Aleš Kepřt, Ph.D., 2005

Vytvořeno pro potřeby přednášky na UP Olomouc. Tento text není určen pro samostudium, ale jen jako vodítko pro přednášku, takže jeho obsah se může čtenáři zdát stručný, nekompletní či možná i chybný. Použití je povoleno dle vlastní libosti, ale jen na vlastní nebezpečí. ☺ V případě dalšího šíření je NUTNO uvádět původního autora a odkaz na původní dokument. Komentáře můžete posílat e-mailem autorovi (adresu najdete přes Google).