PHRUB.CZ
Hlavně klid

XML
Extensible Markup Language

Mgr. Pavel Hrubý

PHRUB.CZ
Hlavně klid

Úvodní informace

- Oblasti použití XML.
- Historie XML a značkovacích jazyků HTML, SGML.
- Struktura XML aplikace.
- Přehled a použití XML editorů.
- Přehled XML parserů, DOM, SAX knihovny.


```
<?xml version="1.0" encoding="UTF-8"?>
- <Problem>
    <ProblemNumber>1000002</ProblemNumber>
    <Subject>Update antivirus on AV Servers</Subject>
    <ErrorMessage>Installation failed with error code: xxxxxx</ErrorMessage>
    <Description>The AV on the network security</Description>
    <Status>Active</Status>
    <Category>Accessibility</Category>
    <Source>Instant Message</Source>
    <OwnerTeam>Problem Management</OwnerTeam>
    <Owner>ADale</Owner>
    <Urgency>Low</Urgency>
    <Urgency>High</Urgency>
  - <Incidents>
      - <Incident>
          <IncidentNumber>50001</IncidentNumber>
          <Subject>Daily Backup Failure</Subject>
          <Symptom>Backup failed on server</Symptom>
        </Incident>
    </Incidents>
</Problem>
```

PHRUB.CZ
Hlavně klid

Oblasti použití XML

- Historie jazyka XML spadá někdy do poloviny devadesátých let.
- Roku 1996 společnost W3C začala pracovat na definování standardu tohoto jazyka.
- Tento jazyk vychází z obecnějšího značkovacího jazyka SGML (Standard Generalized MarkUp Language
- Dnes se jedná o standardní formát pro výměnu a sdílení informací
- Formát XML se používá:
  - výměna informací mezi různými databázovými systémy
  - např. formát dokumentů pro hlášení úřadům
  - elektronické publikování - zcela samostatná oblast, kde se XML využívá ke snadnému konvertování základního dokumentu do celé řady dalších formátů (soubor vhodný pro DTP, obecný PDF soubor, sada HTML stránek,...)
  - tvorba webových stránek – (kombinace jazyka HTML a XML do nového standardu XHTML)
  - E-business - XML je ideální formát výměny dat pro e-businesové transakce
  - Metadata - XML jako formát dat, kterými jsou popsána jiná data.

PHRUB.CZ
Hlavně klid

Historie XML a značkovacích jazyků HTML, SGML

SGML

- Historie značkovacích jazyků spadá do šedesátých let, kdy IBM řešila problém ukládání velkého množství strukturovaných dokumentů. Bylo potřeba vymyslet formát s dlouhou životností a potřebnou nezávislostí na programovém prostředí.
- SGML (Standard Generalized Markup Language)1 - standardní jazyk určený k formálnímu popisu struktury dokumentů.
- SGML je zcela otevřeným standardem nezávislým na platformách a aplikacích.
- SGML je metajazyk, který umožňuje definování dalších značkovacích jazyků.
- V tzv. DTD definici typu dokumentu (Document Type Definition) jsou určeny elementy - značky, které je možné v dokumentu použít, a vztahy mezi nimi.

XML

- rozšiřitelnost (definování tagů podle potřeby);
- strukturovatelnost (úprava dat do různé úrovně složitosti);
- validovatelnost (kontrola dokumentu z hlediska jeho strukturální správnosti);
- nezávislost na médiu (publikování obsahu v různých formátech);
- nezávislost na dodavateli a platformě (zpracování jakéhokoliv dokumentu za použití komerčního softwaru anebo obyčejného nástroje na zpracování textu).

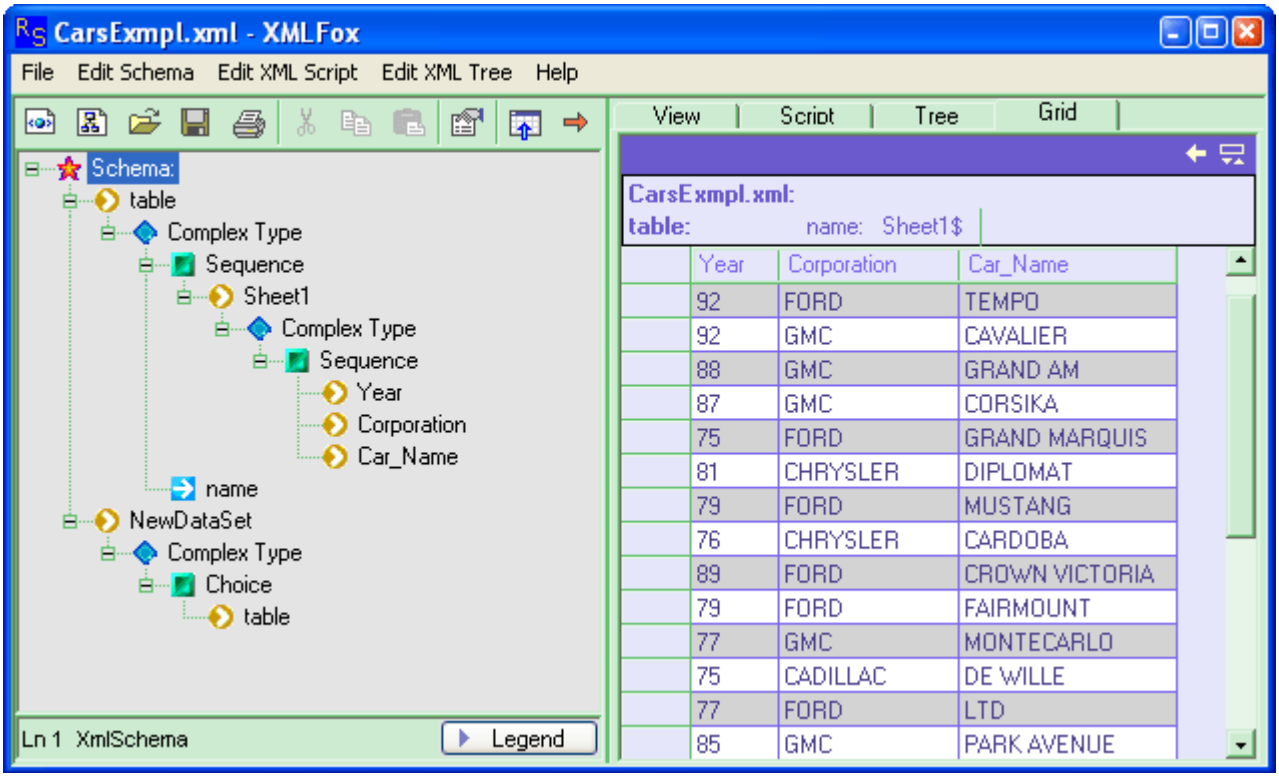
Struktura XML aplikace

- Každý XML dokument se skládá z elementů, které jsou do sebe navzájem vnořené. Elementy se v textu vyznačují pomocí tzv. tagů. Většině elementů odpovídají dva tagy počáteční a ukončovací.
- Základní struktura:

Přehled a použití XML editorů

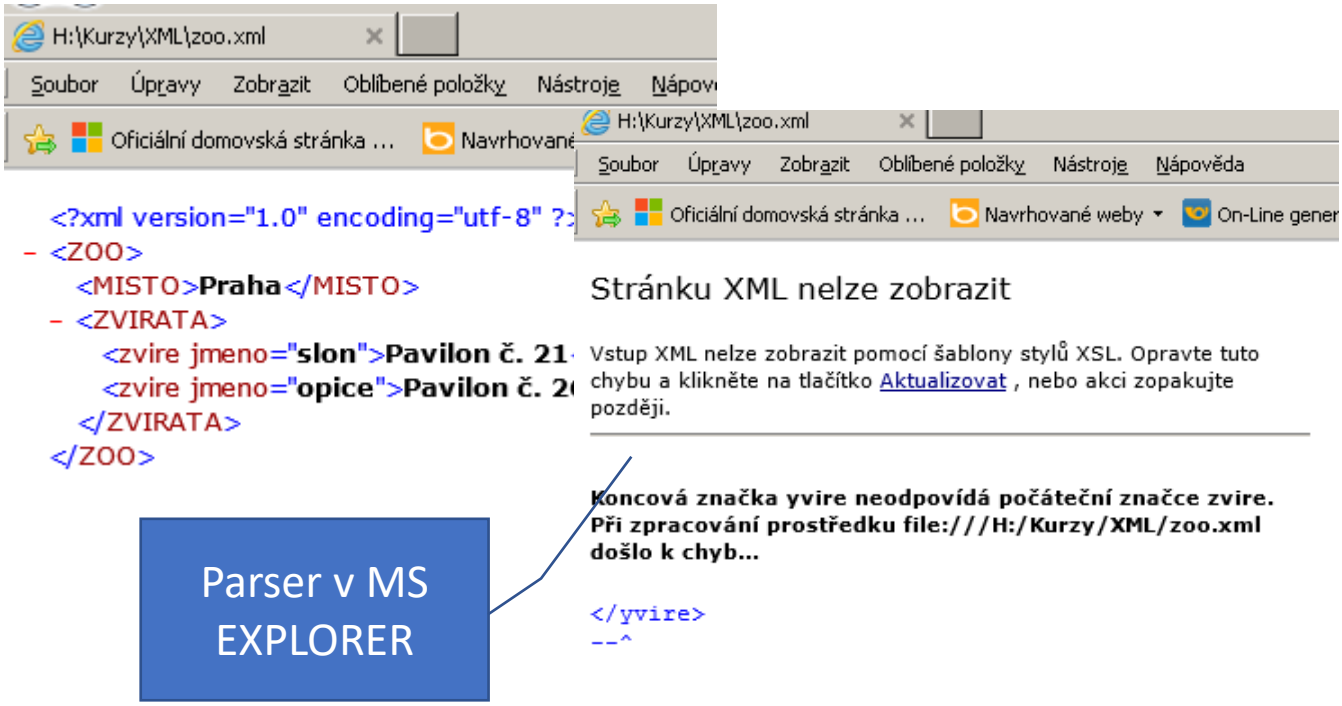
Free XML Editor

- Free XML Editor je přehledný a nenáročný editor dokumentů v XML formátu a XSD schémat.
- XML soubory je možno zobrazit v módech XML View, XML Tree, XML Grid a XML Script. Dokumenty jsou zobrazovány v přehledném rozhraní se dvěma panely.



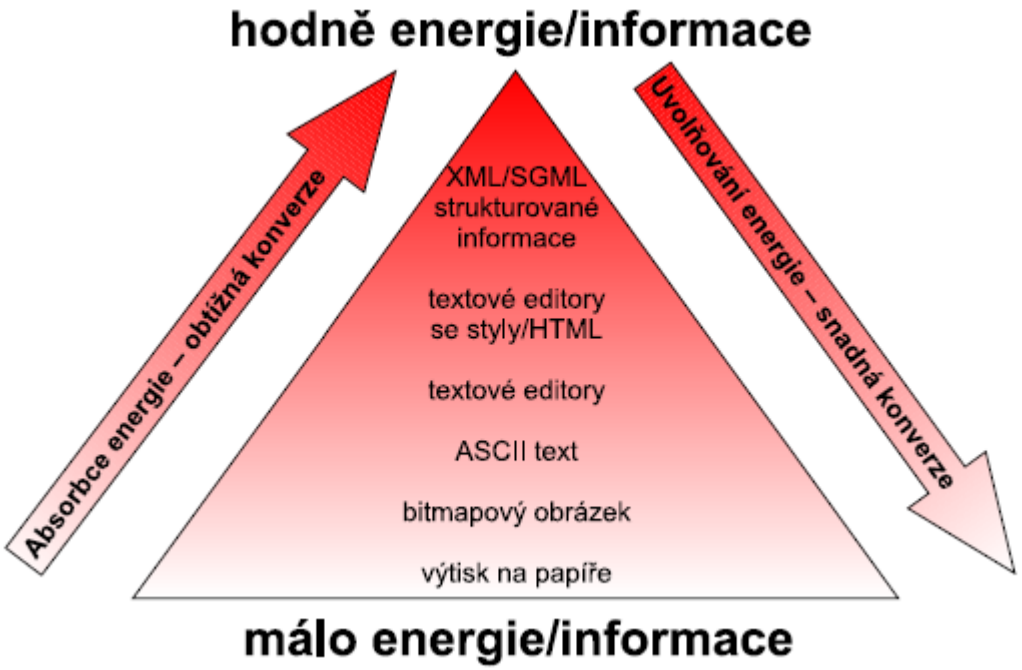
Přehled XML parserů, DOM, SAX knihovny.

- Parser je program, který kontroluje, zda je dokument správně strukturovaný. Lepší parsery zároveň kontrolují, zda dokument odpovídá danému DTD (samozřejmě jen pokud DTD pro dokument existuje).
- Parsery dnes existují ve dvou podobách. Tou první jsou knihovny pro různé programovací jazyky, které můžeme využívat v našich programech. Parser nám pak kromě detekce chyb v dokumentu umožní velmi snadné čtení dat z XML dokumentů.
- XML parsery jsou dnes běžnou součástí internetových prohlížečů.



Struktura XML

- Základní názvosloví: otevírací, uzavírací značka, atributy.
- XML entity.
- Model XML dokumentu, parent, child, sibling vztahy.
- Typy uzlů: element, atribut, text, komentář, procesní instrukce.
- XML a znakové sady, Windows 1250, ISO 8859-2, UTF-8, UTF-16.
- Správně strukturovaný dokument.



PHRUB.CZ
Hlavně klid

Základní názvosloví: otevírací, uzavírací značka, atributy.

PHRUB.CZ
Hlavně klid

XML entity.

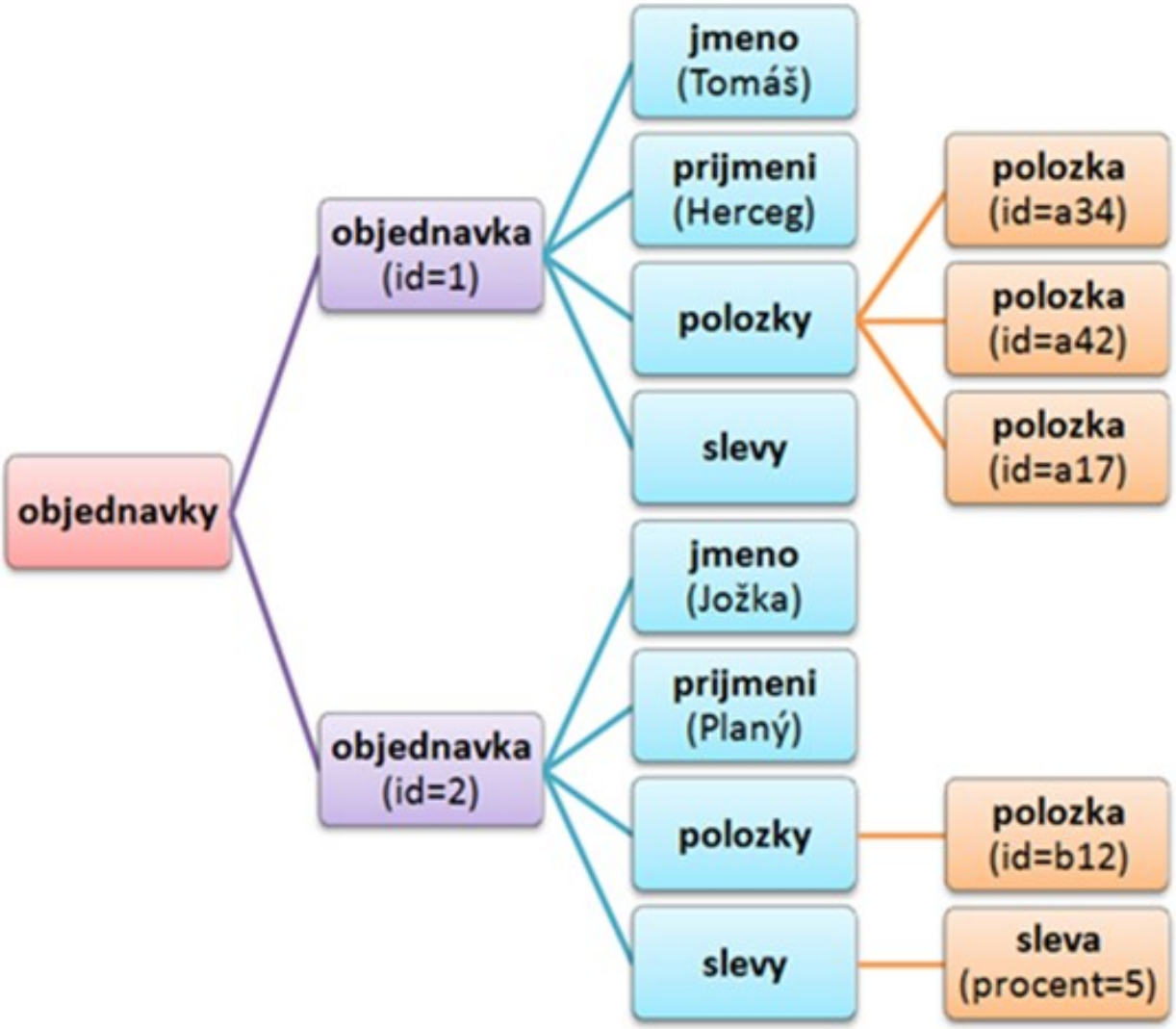
Definice entity

- Entita musí být definována ještě předtím, než ji v dokumentu použijeme.
- **Deklarace všech entit musí být uvedeny v DTD**. Je jedno, zda budou v lokální části v deklaraci typu dokumentu, nebo v externím DTD.

Interní entita

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE clanek[
<!ENTITY program "XML editor">
<!ENTITY autor "Jan Novák">
]>
<clanek>
<nadpis> Jak se naučit XML</nadpis>
<odstavec>Je potřeba umět TAGY</odstavec>
<odstavec>A jak se strukturuje dokument</odstavec>
<odstavec>&program; je nejlepší ve své kategorii</odstavec>
<odstavec>Vypracoval &autor;</odstavec>
</clanek>
```

Model XML dokumentu, parent, child, sibling vztahy.



PHRUB.CZ
Hlavně klid

Typy uzlů: element, atribut, text, komentář, procesní instrukce.

Strom, který XML dokument vyjadřuje čítá několik různých typů uzlů:

- **Element** - Stromovou strukturu XML tvoří zejména elementy. Ty jediné totiž mohou obsahovat děti (potomky). Ostatní uzly stromu vždy tvoří listy stromu. Element může obsahovat množini atributů
- **Dokument** - Jedná se o speciální případ elementu. Neobsahuje žádné atributy, může však obsahovat URL, které odkazuje na specializovaný datový model XML určený pro tento uzel a jeho děti. pokud první výraz v XML dokumentu není <!doctype…> pak má dokument anonymní kořen.
- **Instrukce pro zpracování** (processing instruction) - Takovýto uzel je vždy listem stromu. Obsahuje pouze instrukci i. Instrukce je posloupnost nula či více znaků bez jakýchkolivomezení (nesmí začínat znaky „xml“). Instrukce se v XML dokumentech používají pro potřeby aplikací (například pro XML parser, což je program, který kontroluje správnost XML dokumentu).
- **Komentář** - Komentáře jsou na rozdíl od instrukcí určeny pro potřeby lidí programátorů, uživatelů, …). Obvykle se do nich vepisují vysvětlení elementů, atributů,… a různé poznámky.
- **Datový uzel** - Datové uzly jsou listy stromu. Jejich účel je jediný – obsahují vlastní data. Vše co v XML dokumentu není uzavřeno mezi znaky „<“ a „>“ je pokládáno za data.
- **Atributy** – atributy jsou součástí elementu.

XML a znakové sady, Windows 1250, ISO 8859-2, UTF-8, UTF-16.

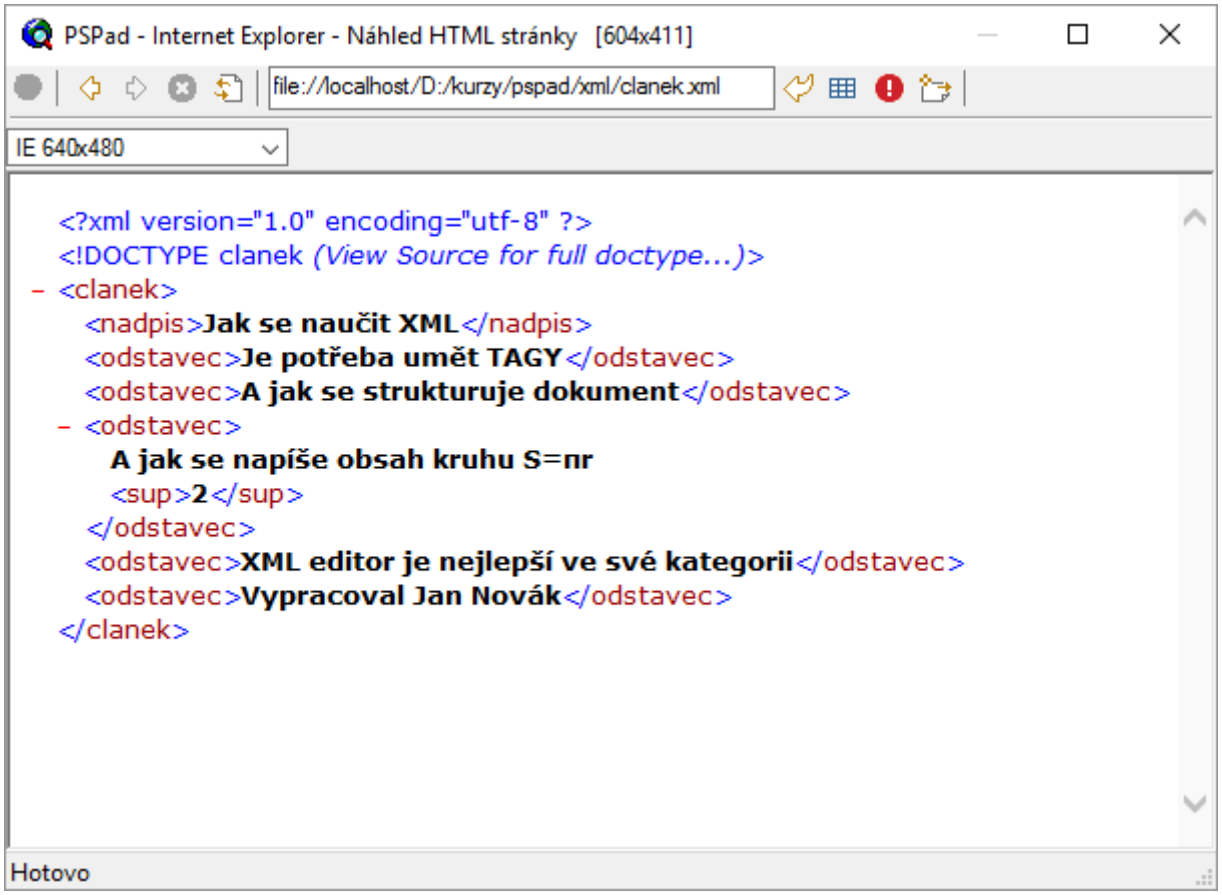
- Zatímco znaková sada definuje, jaké znaky a pod jakým číslem máme k dispozici, kódování znakové sady určuje, jak jsou jednotlivé kódy znaků převedeny na sekvenci bajtů, které znak reprezentují v paměti počítače, v souboru, při přenosu počítačovou sítí apod.
- Kódování v UNICODE – utf-8
- <?xml version="1.0" encoding="utf-8"?>
- Kódování v MS WINDOWS
- <?xml version="1.0" encoding="windows-1250"?>

- Do XML dokumentu vložit **libovolný znak pomocí znakové entity**. Znaková entita má tvar &#x«kód znaku»;, kde «kódznaku» je kód znaku ze znakové sady ISO 10646 zapsaný v šestnáctkové soustavě. Můžeme použít i tvar &#«kód znaku»;, kde je «kód znaku» zapsán v desítkové soustavě.

<odstavec>A jak se napíše obsah kruhu S=πr²
</odstavec>

Správně strukturovaný dokument.

- Strukturu dokumentu kontroluje parser zabudovaný v editorech či v prohlížečích



Návrh XML dokumentu

- Vytvoření slovní zásoby.
- Povolené znaky v XML názvech, jmenné konvence.

Vytvoření slovní zásoby.

- Výhoda ukládání dokumentů v XML spočívá především v možnosti zachytit **pomocí elementů strukturu dokumentu**.
- Pokud dokument obsahuje DTD nebo je na něj odkaz v deklaraci typu dokumentu, parser kontroluje, zda dokument odpovídá DTD. Pokud ano, říkáme, že dokument **je validní**. Pokud dokument porušuje pravidla daná v DTD, je **invalidní**. To, že je dokument invalidní, se nevylučuje s tím, že je správně strukturovaný. Zároveň také platí, že validní dokument musí být správně strukturovaný.

- DTD obsahuje deklarace čtyř typů:
  - deklarace elementů
  - deklarace atributů
  - deklarace entit
  - deklarace notací.

- Více v části DTD

PHRUB.CZ
Hlavně klid

Document Type Definition

- Interní a externí DTD.
- Instrukce ELEMENT, ATTLIST.
- Sekvence child elementů, alternativní obsah.
- Četnosti child elementů.

```
<?xml version="1.0"?><!DOCTYPE books [
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT authors (author)+>
<!ELEMENT subject (#PCDATA)>
<!ATTLIST subject class CDATA "">
<!ELEMENT book (title,authors,subject)>
<!ATTLIST book
    bookid CDATA #REQUIRED
    pubdate CDATA #REQUIRED>
]>
<books name="My books">
    <book bookid="1" pubdate="03/01/2002">
        <title>Java Web Services</title>
        <authors>
```

Interní a externí DTD.

- Tím, že naše dokumenty založíme na určitém DTD, získáme hned dvě výhody. Jednak můžeme pomocí parseru kontrolovat, zda má náš dokument strukturu odpovídající danému DTD. Druhá výhoda je patrná při použití standardních DTD jako HTML nebo DocBook k dispozici budeme mít mnoho užitečných a jednoúčelových nástrojů navržených pro konkrétní DTD.
- Pokud chceme využít výhody DTD, musíme použité DTD určit pomocí deklarace typu dokumentu (DOCTYPE), která se umísťuje na samotný začátek dokumentu, ihned za XML deklaraci. Obvykle je DTD uloženo v samostatném souboru, aby mohlo být opakovaně využíváno mnoha dokumenty.

Externí DTD má v tomto případě deklaraci ve tvaru

```
<!DOCTYPE clanek SYSTEM "clanek.dtd">
```

Interní DTD

DTD můžeme umístit i přímo do dokumentu pomocí následujícího zápisu

```
<!DOCTYPE clanek [
```

Vlastní DTD

```
]>
```

PHRUB.CZ
Hlavně klid

Instrukce ELEMENT, ATTLIST.

Deklarace elementu

<!ELEMENT ***název elementu*** ***obsah elementu***>

<!ELEMENT clanek (nadpis, odstavec)>

Deklarace elementu EMPTY

- Právě klíčové slovo EMPTY určuje, že element nesmí nic obsahovat. V dokumentu pak musíme psát buď
</br>, nebo zkráceně
. Není však možný zápis samotného
, protože by parser zcela marně hledal ukončovací tag </br>.

Příklad DTD

```
<?xml version="1.0" encoding="utf-8"?>
<!ELEMENT br EMPTY>
<!ELEMENT clanek (nadpis,odstavec)>
<!ELEMENT nadpis (#PCDATA)>
<!ELEMENT odstavec (#PCDATA)>
```

PHRUB.CZ
Hlavně klid

Instrukce ELEMENT, ATTLIST.

Deklarace atributů

<!ATTLIST ***název elementu*** ***deklarace atributů***>

<!ATTLIST clanek **autor CDATA …**>

- Deklarace jednotlivých atributů se skládá ze tří částí. První částí je jméno atributu. Pro jména atributů platí stejná omezení jako pro jména elementů. Atributy jsou rovněž citlivé na velikost písmen, a tak MujAtribut a MUJATRIBUT jsou dva zcela rozdílné atributy.
- Za jménem následuje typ atributu. Poslední část určuje standardní hodnotu atributu, popřípadě zda je používání atributu u daného elementu povinné.
- Deklarace se opakuje pro každý atribut, který má být u elementu k dispozici.
- Nejobecnějším typem atributu je CDATA, který umožňuje jako jeho hodnotu zadat libovolný textový řetězec.
  <!ATTLIST kniha autor CDATA ...>

Sekvence child elementů, alternativní obsah.

- V deklaraci všech atributů jsme zatím na posledním místě používali tři tečky. Na tomto místě se uvádí **standardní hodnota atributu nebo to, zda je atribut povinný.**
- Pokud je atribut povinný, uvede se za deklarací jeho typu slovo **#REQUIRED**.
- Parser při kontrole dokumentu ohlásí chyby vždy, když nebude tento atribut u elementu použit. XML editor si může zadání hodnoty atributu vynutit přímo při vložení elementu, který povinný atribut obsahuje.
- Opačným případem je situace, kdy můžeme atribut vynechat, a v tomto případě si jeho hodnotu podle svých potřeb domyslí sama konkrétní aplikace, která právě s dokumentem pracuje. V těchto případech se použije klíčové slovo **#IMPLIED**.
- Další možností je specifikování standardní hodnoty. Stačí ji uvést. Pokud chceme, aby byl obrázek zarovnán vlevo, pokud není určeno jinak, použijeme deklaraci:
<!ATTLISTobrazekzarovnani(vlevo|vpravo|doprostred)"vlevo">

Jmenné prostory

- Funkce a označování jmenných prostorů

Jmenné prostory (namespaces)

Jmenný prostor vytváří množinu jmen daného významu. Jmenné prostory umožňují používat nezávisle na sobě několik druhů značek. Můžeme tak vytvářet dokumenty, ve kterých jsou použité značky stejného jména a jiného významu.

Výchozí jmenný prostor

- deklarujeme pomocí speciálního atributu xmlns, hodnotou tohoto atributu je nejčastěji URL

Prefix jmenného prostoru

- kombinujeme-li více jmenných prostorů v dokumentu, oddělujeme je deklarovanou předponou čímž začleňujeme použité elementy do daného jmenného prostoru

Funkce a označování jmenných prostorů.

- Bývá dnes dobrým zvykem přiřadit každému nově vytvořenému značkovacímu jazyku vlastní jmenný prostor, aby jej šlo snadno identifikovat.
- Jmenný prostor, do něhož budou elementy patřit, se určuje pomocí atributu **targetNamespace** u kořenového elementu schématu.
- Z praktických důvodů se tento jmenný prostor obvykle deklaruje i jako implicitní, abychom před naše elementy nemuseli všude psát nějaký prefix.

Příklad:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://example.org/ns"
        xmlns="http://example.org/ns"
        elementFormDefault="qualified">

<xs:element name="root" type="mujTyp"/>
<xs:simpleType name="mujTyp">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:schema>
```

XML Schema

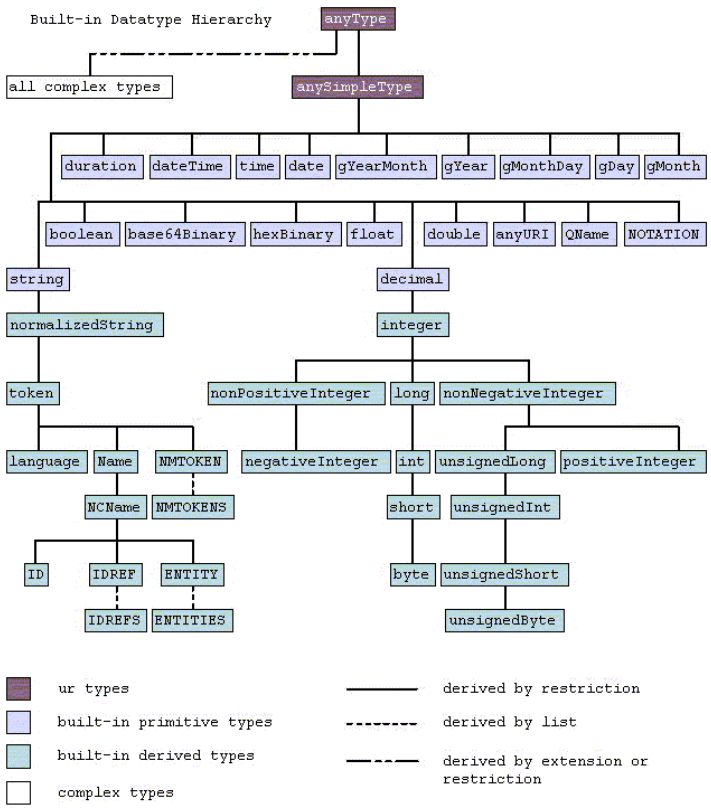
- Jednoduché a složené datové typy
- Restrikce
- Globální, lokální deklarace
- Metody návrhu XML schemat.

```
<!-- Schema Components -->
<xs:complexType name="baseComponent">
    <xs:complexContent> [29 lines]
</xs:complexType>
<xs:complexType name="componentWithFacets">
    <xs:complexContent>
        <xs:extension base="baseComponent"> [3 lines]
    </xs:complexContent>
</xs:complexType>
<xs:element name="schema">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="baseComponent">
                <xs:attribute name="type" use="required"
```

Jednoduché a složené datové typy.

Jednoduché datové typy

- XML schémata obsahují běžné základní datové typy jako textový řetězec, celá a desetinná čísla, binární data, logická hodnota, datum, čas, časový interval a několik typů převzatých z DTD pro jejich snazší konverzi do XML schémat. V mnoha případech si s těmito typy vystačíme. Naše první ukázka si například naprosto vystačila se zabudovanými typy pro textové řetězce (string), datum (date) a desetinné číslo (decimal).



PHRUB.CZ
Hlavně klid

Jednoduché a složené datové typy.

Složené datové typy

- Komplexní typy slouží k modelování struktury dokumentu, protože se mohou skládat z elementů a atributů. U elementů můžeme určit v jakém pořadí se mají vyskytovat, kolikrát se mohou opakovat, zda jsou povinné či volitelné.
- Komplexní typ se definuje pomocí elementu complexType. Podobně jako u jednoduchých typů můžeme definovat komplexní typ samostatně, aby pak šel používat opakovaně pro různé elementy. Následující příklad ukazuje, jak definovat elementy odberatel a dodavatel tak, že mají stejný obsah, právě pomocí společně použitého uživatelsky definovaného komplexního typu subjektType.

Příklad

```
<xs:element name="clanek">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="nazev" type="xs:string"/>
   <xs:element name="autor" type="xs:string"
            minOccurs="0" maxOccurs="unbounded"/>
   <xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="odstavec" type="xs:string"/>
    <xs:element name="obrazek" type="xs:base64Binary"/>
   </xs:choice>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

PHRUB.CZ
Hlavně klid

Restrikce

- Nejpoužívanějším typem odvození nového typu je bezesporu restrikce. Na existující datový typ můžeme najednou aplikovat několik integritních omezení a tak zúžit přípustné hodnoty.

```
<xs:simpleType name="PSČType">
  <xs:restriction base="xs:string">
    <xs:length value="6"/>
  </xs:restriction>
</xs:simpleType>
```

PHRUB.CZ
Hlavně klid

Globální, lokální deklarace.

- Za globální se považují deklarace provedené na nejvyšší úrovni schématu, přímo v elementu schema.
- Globálně deklarované elementy a datové typy mají v některých ohledech speciální chování.
- Dokument vyhovující schématu může začínat libovolným globálním elementem.
- Ve většině schémat má proto smysl deklarovat jako globální pouze ten element, který chceme použít jako obálku okolo celého dokumentu XML.

Příklad

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="urn:x-k:schemas:faktura:1.0"
        xmlns="urn:x-k:schemas:faktura:1.0"
        elementFormDefault="qualified">
 <xs:element name="faktura">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="odberatel" type="subjektInfoTyp" />
    <xs:element name="dodavatel" type="subjektInfoTyp" />
    <xs:element ref="polozka" minOccurs="1" maxOccurs="unbounded" />
   </xs:sequence>
   <xs:attribute name="cislo" type="cisloFakturyTyp" use="required" />
   <xs:attribute name="vystaveni" type="xs:date" use="required" />
   <xs:attribute name="splatnost" type="xs:date" use="required" />
   <xs:attribute name="vystavil" type="xs:string" />
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Metody návrhu XML schemat.

- Postupem času se vyvinulo několik přístupů k tomu, jak správně vystavět schéma. Problém, který je potřeba rozhodnout, spočívá ve výběru, kdy používat lokální, a kdy globální elementy, jak moc bude schéma rozšiřitelné a použitelné v dalších schématech.

Ukázkový dokument XML

```
<?xml version="1.0" encoding="UTF-8"?>
<zamestnanec>
  <jmeno>Jan</jmeno>
  <prijmeni>Novák</prijmeni>
  <adresa>
    <ulice>Dlouhá 2</ulice>
    <město>Praha 1</město>
    <psč>110 00</psč>
  </adresa>
  <plat>34500</plat>
</zamestnanec>
```

PHRUB.CZ
Hlavně klid

Metody návrhu XML schemat.

Schéma ve stylu matrjóška –zamestnanec-matrjoska.xsd

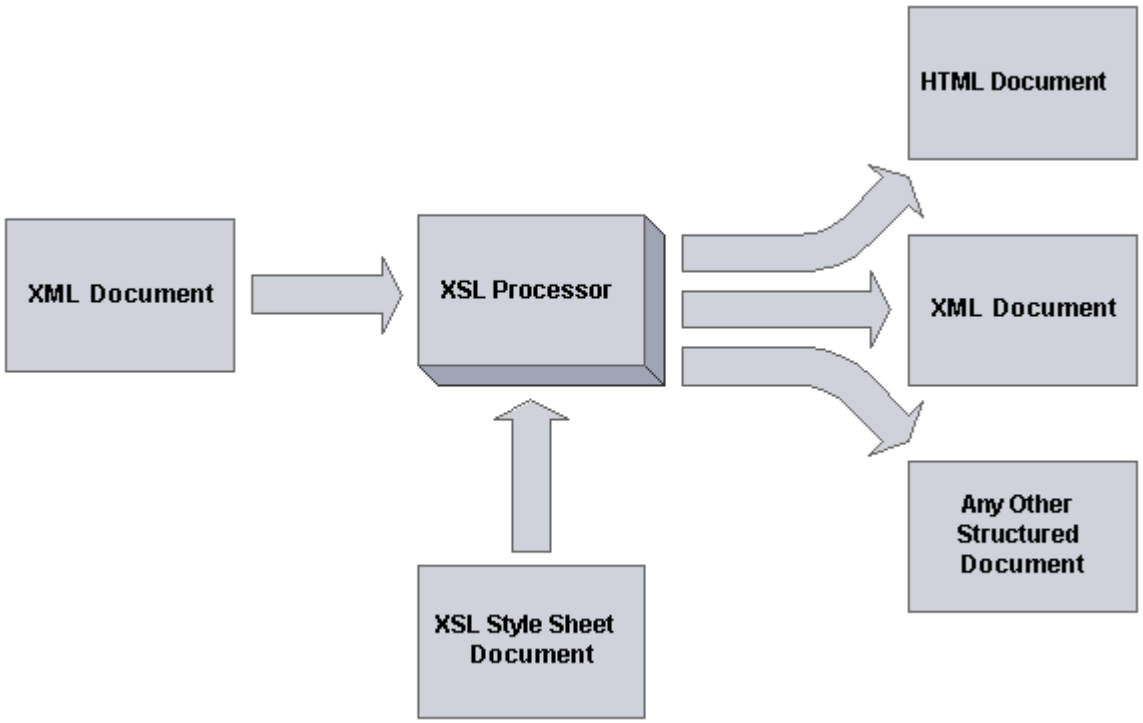
```
?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="zamestnanec">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="jmeno" type="xs:string"/>
    <xs:element name="prijmeni" type="xs:string"/>
    <xs:element name="adresa">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="ulice" type="xs:string"/>
       <xs:element name="město" type="xs:string"/>
       <xs:element name="psč" type="xs:string"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="plat" type="xs:decimal"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Připojení schematu k dokumentu

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

Prezentování XML obsahu

- XML a CSS styly
- XSL transformace



PHRUB.CZ
Hlavně klid

XML a CSS styly.

- Pokud chceme, aby se při zobrazování dokumentu používal určitý styl, musíme to samozřejmě jednotlivým aplikacím sdělit. Používá se k tomu speciální instrukce pro zpracování xml-stylesheet. Vždy musíme určit URL adresu, na které je uložen styl, a typ použitého stylového jazyka.

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

- Instrukci pro připojení stylu musíme uvádět na začátku dokumentu.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="mujstyl.xsl" type="text/xsl"?>
<dokument>
  ...
</dokument>
```

- Tomuto zápisu rozumí většina prohlížečů podporujících XML a některé editory. Stylů lze připojit k jednomu dokumentu dokonce více najednou. Lze například určit, který styl se použije pro tisk, a který při zobrazení na obrazovce.

XSL transformace a XPath.

- XSL (eXtensible Stylesheet Language) vznikl jako univerzální stylový jazyk, který by měl nabízet funkčnost všech ostatních existujících stylových jazyků a dále ji rozšířit. Samotná syntaxe jazyka je samozřejmě založena na XML, takže pro zpracování stylu lze použít všechny nástroje, které umějí s XML pracovat.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="utf-8"/>
<xsl:template match="/">
 <html>
  <head>
   <title>Moudrá přísloví</title>
  </head>
  <body>
   <xsl:apply-templates/>
  </body>
 </html>
</xsl:template>
<xsl:template match="citat">
 <xsl:apply-templates/>
 <hr/>
</xsl:template>
<xsl:template match="text">
 <p><xsl:apply-templates/></p>
</xsl:template>
<xsl:template match="autor">
 <p align="right">&#x2014; <em><xsl:apply-templates/></em></p>
</xsl:template>
</xsl:stylesheet>
```

PHRUB.CZ
Hlavně klid

Zdroje informací

- http://www.kosek.cz/clanky/xml/xml-uvod.html
- https://www.systemonline.cz/clanky/zaklady-xml-a-jeho-prakticke-vyuziti.htm
- http://webserver.ics.muni.cz/bulletin/articles/180.html
- https://www.interval.cz/clanky/xml-pro-kazdeho/
- https://www.root.cz/serialy/xml-extensible-markup-language/