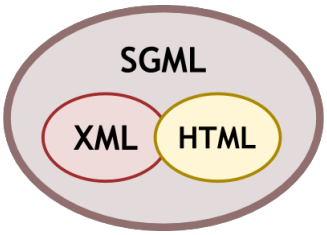
XML

- obecný jazyk SGML
- jak vypadá XML dokument, využití v praxi
- XML a layout, co to je eXtensible Stylesheet Language (XSLT styl)
- co to je definice typu dokumentu (DTD)

Obecný jazyk SGML

XML patří do skupiny tzv. **značkovacích jazyků**. Tyto jazyky umožňují označovat části textu **značkami** (tagy).



Vztah mezi SGML, XML a HTML

Předchůdcem XML byl jeden z prvních značkovací jazyků **SGML** (Standard Generalized Markup Language). Ten byl přijat v roce 1986 jako ISO norma. Asi prvním známých značkovacím jazykem byl jazyk GML, který byl vytvořen při práci na systému pro uchovávání a následné využití právních textů pro IBM. Autoři se museli vypořádat s nekompatibilitou jednotlivých systémů a programů a nejsnazší cesta vedla právě přes vytvoření nějakého obecného značkovacího jazyka. Tak vzniklo SGML.

Vlastnosti

- umožňuje definovat vlastní **značkovací jazyky** (pomocí DTD viz dále)
- umožňuje nastavit spoustu volitelných parametrů (maximální délku názvů značek, určit znaky sloužící jako oddělovače značek od textu a **mnohem mnohem** víc)
- velice obecný a složitý/komplexní

Na jeho základech stojí mnoho dalších značkovacích jazyků (HTML, XML, atd.).

Po letech používání jazyka SGML se ukázalo, že se ve skutečnosti používá jen část jeho skutečných možností. Tato část byla v roce 1996 vybrána jako základ pro nově vznikající jazyk XML.

Jak vypadá XML dokument

Na začátku každého XML dokumentu se může nacházet (*měla by, je to slušnost*) tzv. **XML deklarace**. V základu vypadá takto:

```
<?xml version="1.0" ?>
```

Deklarace sděluje programu, který bude dokument zpracovávat, některé důležité informace. V uvedeném případě se jedná o informaci, že dokument je zapsán podle XML verze 1.0 (aktuální je verze 1.1). Deklarace **musí** obsahovat alespoň informaci o použité verzi XML. Poté je možné ještě určit použité kódování:

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

Uvedené informace musí být uvedeny přesně v tomto pořadí. Když není hodnota `encoding` uvedena, je předpokládano použití výchozího kódování **ISO 10646**. Jedná se o 32-bitovou znakovou sadu, která je schopná pojmout všechny znaky dnes používaných jazyků. Je téměř shodná se standardem Unicode (zdroj).

Za XML deklarací se nechází **kořenový element**. Kořenový element je nejvyšším elementem dokumentu, všechny ostatní elementy jsou do něj vnořeny. XML dokument musí obsahovat alespoň jeden element (třeba jenom kořenový). Kořenový element je **jenom jeden**.

```
<?xml version="1.0" encoding="UTF-8" ?>
<trida name="it4" startYear="2008" endYear="2012">
        <tridniUcitel name="Ivana Šmejkalová" alias="SJ" />
        <zaci>
                <zak>
                        <jmeno>Mirek</jmeno>
                        <prijmeni>Balog</prijmeni>
                        <cislo>1</cislo>
                        <znamky year="2008">
                                <znamka lesson="tev" grade="1">
                                <znamka lesson="cjl" grade="2">
                                <znamka lesson="pvy" grade="1">
                        </znamky>

                        <znamky year="2009">
                                <znamka lesson="tev" grade="1">
                                <znamka lesson="cjl" grade="2">
                                <znamka lesson="pvy" grade="1">
                        </znamky>
                </zak>

                <zak>
                        <jmeno>Jakub</jmeno>
                        <prijmeni>Beneš</prijmeni>
                        <cislo>2</cislo>
                        <znamky year="2008">
                                <znamka lesson="tev" grade="1">
                                <znamka lesson="cjl" grade="2">
                                <znamka lesson="pvy" grade="1">
                        </znamky>
                </zak>
                <!-- atd. -->
        </zaci>
</trida>
```

Kořenovým elementem je zde element `trida`. Ten obsahuje **atributy** `name`, `startYear` a `endYear`. Do kořenového elementu jsou vnořeny další elementy - element `tridniUcitel` a element `zaci`.

Každý element je tvořen **uvozovací** a **koncovou značkou** (tagem) a svým obsahem. Koncový tag obsahuje před jménem tagu lomítko (`</tag>`). Uvozovací tag může obsahovat atributy (`name="it4"`, `startYear="2008"`, apod.)

```
<mujTag>Obsah elementu</mujTag>
```

Element nemusí obsahovat žádný obsah.

```
<mujTag></mujTag>
```

Tento zápis je ale zbytečně zdlouhavý a komplikovaný, proto nám XML umožňuje zapsat element bez obsahu tímto způsobem:

```
<mujTag />
```

Zde je uvozovací a koncový tag spojen do jednoho. Třída bez žáků by se tak zapsala takto:

```
<trida name="Prázdná třída" />
```

Tagy se **nesmí** křížit. Toto je špatný zápis:

```
<a><b></a></b>
```

Správně má být:

```
<a><b></b></a>
```

Jazyk XML je velice přísný v oblasti správného zápisu. Stačí jedna drobná chyba v zápisu a zpracování dokumentu skončí chybovou zprávou. Tímto se liší např. od HTML, kde prohlížeče spoustu chyb ignorují a snaží se uhodnout, jak to mělo být správně. Pokud je dokument správně napsán (odpovídá syntaxi (správnému XML zápisu)), říkáme, že se jedná o **well-formed** dokument.

Součástí dokumentu můžou být také instrukce pro nějaký jiný program. Můžeme tak do dokumentu zařadit odkaz na styl definující zobrazení v prohlížeči, formátovacímu programu můžeme naznačit, kde má zalomit stránku, nebo vložit do dokumentu kód některého skriptovacího jazyka. Tyto instrukce XML parser ignoruje, předá je nadřazené aplikaci záleží na ní, zda je nějak využije. Syntaxe je následující:

```
<?«identifikátor» «data»?>
```

Identifikátor je identifikátor externího programu, např. `php`. Můžeme tak do dokumentu vložit kód jazyka PHP, který nám do dokumentu vypíše dnešní datum.

```
<dokument>
    <datum>Dnešní datum je <?php echo date("d.m.Y");?></datum>
    <para>Nějaké další informace</para>
</document>
```

Pomocí instrukcí je možno připojit k XML dokumentu i externí styl:

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

Do dokumentu také můžeme vložit libovolný znak pomocí tzv. entit. Znaková entita má tvar `&#x«kód znaku»;`, kde «kódznaku» je kód znaku ze znakové sady ISO 10646 (Unicode – UCS) zapsaný v šestnáctkové soustavě. Můžeme použít i tvar `&#«kód znaku»;`, kde je «kód znaku» zapsán v desítkové soustavě.

Využití v praxi

XML lze považovat za poměrně univerzální formát, který nalézá uplatnění v mnoha oblastech.

- **B2B, business-to-business aplikace** (výměna informací mezi obchodními partnery v elektronickém formátu)
- **Inteligentní webové stránky** (Možnost definice vlastních značek, které přesně vyznačí význam jednotlivých částí stránky, má pozitivní efekt na přesnost a rychlost vyhledávání informací. Obecný model umožňuje vývoj stránek pro mnoho zařízení s rozdílnými schopnostmi PC, mobilní telefony, WebTV nebo třeba herní konzole.)
- **Metadata** (Pro vyhledávání, ale hlavně pro klasifikaci dokumentů, je užitečné o nich znát co nejvíce metadat.)
- **Elektronické publikování** (Často potřebujeme jeden dokument v několika různých formátech jako tištěnou knihu, sadu provázaných webových stránek nebo hypertextovou příručku na CD-ROMu. Flexibilita stylových jazyků také umožňuje, aby se z jednoho zdroje generovalo několik druhů dokumentů s různým obsahem, některé údaje v technické dokumentaci jsou například tajné a vytisknou se pouze pro potřeby firmy, přičemž zákazníkům se ze stejného XML dokumentu vygeneruje okleštěná verze.)
- **Univerzální datový formát** (Pro ukládání složitějších parametrů programu se nehodí jednoduché způsoby typu INI soubor. Navíc je poměrně komplikované psát program, který bude načítat jednotlivé parametry z konfiguračního souboru. Pokud by si program ukládal data v XML, mohl by je programátor velice snadno číst pomocí některé z knihoven pro práci s XML.)

Konkrétní použití XML:

- **XHTML** – XML alternativa jazyka HTML.
- **RSS** – Rodina XML formátů, sloužící pro čtení novinek na webových stránkách.
- **SMIL** – Synchronized Multimedia Integration Language, popisuje multimedia pomocí XML.
- **MathML** – Mathematical Markup Language je značkovací jazyk pro popis matematických vzorců a symbolů pro použití na webu.
- **SVG** – Scalable Vector Graphics je jazyk pro popis dvourozměrné vektorové grafiky, statické i dynamické (animace).
- **Jabber** – Protokol pro Instant messaging.
- **SOAP** – Protokol pro komunikaci mezi Webovými službami.
- **Office Open XML, OpenDocument** – Souborový formát určený pro ukládání a výměnu dokumentů vytvořených kancelářskými aplikacemi.
- **a mnoho dalších** (formát pro ukládání TV programu, formáty pro popis 3D scén, popis editace 2D grafiky, konfigurační soubory atd.)

XML a layout

Pomocí XML značek vyznačujeme v dokumentu význam jednotlivých částí textu. Říkáme toto je název výrobku, tohle zase telefonní číslo a tohle je číslo našeho účtu. Dokumenty obsahují mnohem více informací, než kdyby se používalo prezentační značkovaní tohle je tučným písmem Arial o velikosti 12 bodů zarovnané vlevo.

V mnoha případech potřebujeme XML dokument zobrazit na nějakém běžném médiu na obrazovce, na papíře. V tomto případě už samozřejmě chceme přesně ovlivnit, jak se obsah jednotlivých značek zobrazí. XML samo o sobě žádné takové prostředky nenabízí. Existuje však naštěstí hned několik stylových jazyků, které umožňují definovat, jak se mají jednotlivé elementy zobrazit. Souboru pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu, se říká **styl**.

Výhodou je, že jeden styl můžeme aplikovat na mnoho dokumentů stejného typu. Dosáhneme tak jednotného formátování. Zároveň můžeme na jeden dokument aplikovat několik různých stylů. Jedním stylem vygenerujeme postscriptový soubor pro naše DTP studio, druhým HTML kód pro zařazení na naše webové stránky a třetím třeba jen obsah dokumentu, který pošleme mailem šéfovi.

Stylových jazyků je mnoho, mezi nejznámější patří asi kaskádové styly (CSS). Ty lze použít pouze pro jednoduché formátování, které dobře poslouží pro zobrazení dokumentu na obrazovce v XML editoru nebo v prohlížeči. Pro náročnější aplikace slouží jazyk **XSL** (eXtensible Stylesheet Language).

Co to je eXtensible Stylesheet Language (XSLT styl)

XSL zahrnuje 3 jazyky:

1. XSL Transformace (XSLT): XML jazyk k převádění XML dokumentů,
2. XSL Formátovací Objekty (XSL-FO): XML jazyk pro specifikaci vizuálního formátování XML dokumentů,
3. XML Path jazyk (XPath, česky „jazyk pro cesty v XML“): jazyk k adresování částí XML dokumentu, který

ale sám není XML jazykem. Je používán například v XSLT.

Transformace **XSLT** (eXtensible Stylesheet Language Transformations) slouží k převodům zdrojových dat ve formátu XML do libovolného jiného požadovaného formátu, nejčastěji HTML, jiného XML nebo libovolných jiných datových struktur.

K provedení transformace jsou potřeba dva soubory:

- První soubor obsahuje zdrojová data, která budou transformována. Struktura tohoto souboru vyjma obecných vlastností XML není blíže specifikována.
- Druhý soubor obsahuje vzorec pro transformaci a musí být napsán v jazyce XSL.

Odkazování v XML dokumentu

XML umožňuje vytváření odkazů v rámci jednoho dokumentu i mezi dokumenty navzájem. Nabízí však mnoho možností nad rámec odkazů, které známe z HTML. Můžeme vytvářet i vícesměrné odkazy, které spojují několik dokumentů dohromady. Užitečná je i možnost uložení odkazů zcela mimo dokumenty, kterých se týkají. Tímto způsobem lze vytvářet různé anotace a komentáře k již existujícím stránkám.

Co to je definice typu dokumentu (DTD)

~~DTD (Document Type Definition) je jedna z důležitých vlastností, které převzalo XML z jazyka SGML. Umožňuje definovat vlastní sadu značek (elementy a atributy), které budou XML dokumentu k dispozici. V definici můžeme určit, kde se dané elementy a atributy mohou nacházet a jaký mohou mít obsah či hodnotu. Jestliže XML dokument odpovídá všem pravidlům v určitém DTD, říkáme, že je dokument **validní**.~~

~~DTD např. umožňuje určit, že element `trida` smí obsahovat pouze elementy `tridniUcitel` a `zaci` a že jeho jedinými povolenými atributy jsou `name`, `startYear` a `endYear`. Dále můžeme určit povolené atributy pro element `tridniUcitel` a také říct že se v elementu `trida` může vyskytovat pouze jednou. A nakonec určíme, co smí obsahovat element `zaci`.~~

~~Jedná se v podstatě o pravidla pro daný XML dokument. DTD nám umožňuje říct, co smí XML dokument obsahovat a v jaké podobě.~~

Definice typu dokumentu (DTD) říká, které elementy a atributy můžeme v dokumentu použít. Navíc je zde definováno, v jakých vzájemných vztazích mohou být jednotlivé elementy použity. DTD je tedy užitečný nástroj, který nám umožní hlídat, zda mají naše dokumenty správnou strukturu. Ve světě se používá mnoho DTD, které vyhovují různým požadavkům. Mezi jedno z nejznámějších patří například DTD DocBook, které definuje elementy a atributy vhodné pro značkování technické dokumentace.

DTD se hodí pro popis formátů, které se používají především pro textové dokumenty. Neobsahuje však nástroje pro kontrolu různých typů dat jako čísla, měnové údaje, údaje o datu a čase. To je přitom velice důležité pro aplikace, které si pomocí XML posílají data spíše databázového charakteru. Pro tyto potřeby existuje několik dalších jazyků, umožňujících určit správné schéma dokumentu.

K jakému DTD se náš XML dokument hlásí oznámíme použitím tzv. **DOCTYPE**, hned za XML deklarací. Pro

opakované použití bývá DTD uloženo v samostatném souboru.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE «kořenový element» SYSTEM "«URL»">
```

- kořenový element - jméno elementu, ve kterém bude obsažen celý document, v našem případě `trida`
- URL - adresa nebo jméno souboru, ve kterém je DTD uloženo

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE trida SYSTEM "denifinice-tridy.dtd">
```

Můžeme také DTD zapsat přímo do XML dokumentu:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE kniha [
« ... DTD ... »
]>
```

Syntaxe DTD

DTD obsahuje:

- deklarace elementů
- deklarace atributů
- deklarace entit

Deklarace elementů

- **prázdný element** - nemůže obsahovat text ani další elementy
  - v HTML např. `br` pro nový řádek a `hr` pro vložení horizontální čáry <!ELEMENT br EMPTY>☐<!ELEMENT hr EMPTY>
- **element obsahující další elementy** - např. HTML element `<html>` může obsahovat elementy `head` a `body` <!ELEMENT html (head, body)>
  - pokud chceme vyjádřit, že element může obsahovat jenom jeden z uvedených použijeme `|` <!ELEMENT potomek (dcera | syn)>
  - článek vždy obsahuje název, ale nemusí obsahovat autora <!ELEMENT clanek (nazev, autor?)>
  - chceme vyjádřit, že se nějaký element může vyskytovat opakovaně, minimálně však jednou (např. kniha obsahuje minimálně jednu kapitolu) <!ELEMENT kniha (kapitola+)>
  - vraťme se k příkladu se článkem - nyní může článek obsahovat více autorů, nebo také žádného (tj. `0` až N autorů) - použijeme hvězdičku <!ELEMENT clanek (nazev, autor*)>
- **element smí obsahovat pouze text a ne další elementy** - použijeme speciální slovo `#PCDATA` <!ELEMENT em (#PCDATA)>

Deklarace atributů

V XML může mít každý element libovolné množství atributů. Atributy se většinou používají pro připojení různých metainformací k elementům. Deklarace atributů pro element má poměrně jednoduchý tvar:

```
<!ATTLIST «jméno elementu» «deklarace atributů»>
```

Deklarace:

```
<!ATTLIST kniha autor CDATA ...>
```

Nám umožní používat u elementu `kniha` atribut `autor`:

```
<kniha autor="Čapek">
```

Deklarace entit

V XML můžeme předdefinovat libovolné množství entit. Každá bude mít vyhrazen svůj identifikátor.

Např.

```
<!ENTITY quot "&#34;">
```

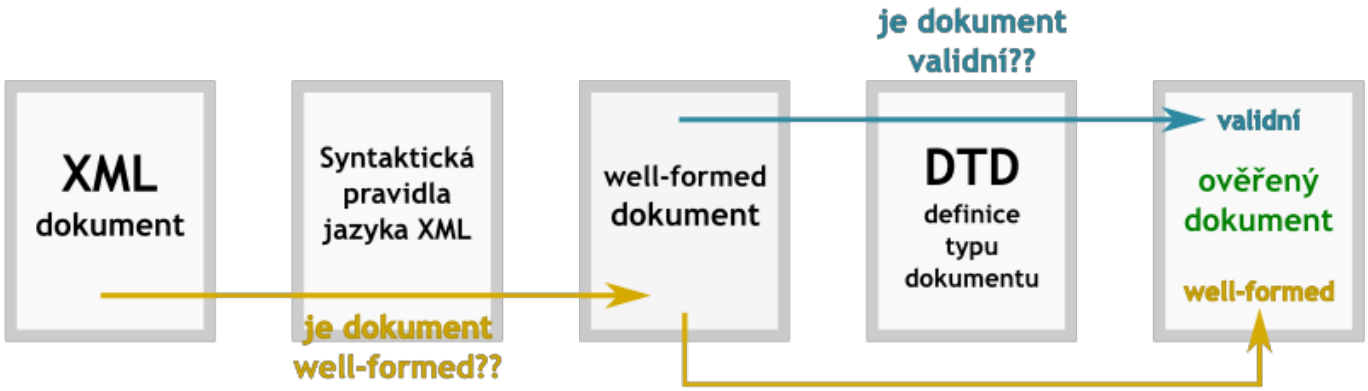
nebo

```
<!ENTITY logo SYSTEM "/obr/logo.eps" NDATA "EPS">
```

Takto definovanou entitu nemůžeme přímo vložit do textu dokumentu a ani by to nemělo smysl. Můžeme na ní však odkázat pomocí atributu, který má typ `ENTITY` nebo `ENTITIES`. Od roku 1999 má naše firma logo, které je na obrázku `<picture name="logo"/>`

Kontrola dokumentu podle DTD

Jak už bylo řečeno, pokud XML dokument odpovídá struktuře popsané pomocí DTD, říkáme o XML dokument, že je **validní**, jinak je **invalidní**. Validita dokumentu je tzv. 2 stupeň správnosti dokumentu. Prvním stupněm je tzv. syntaktická správnost, tj. zda-li je dokument napsán bez syntaktických chyb (neuzavřené tagy a hodnoty, překřížené tagy, překlepy v názvech elementů, atributů, apod.). Pokud je dokument syntakticky správný, říkáme, že je tzv. **well-formed**.



Pro zpracování XML dokumentu se používá tzv. **parser**. Ten načítá XML dokument a poskytuje nám snadný způsob, jak přečíst data z XML dokumentu, a zároveň kontroluje jeho syntaktickou správnost a v případě přítomnosti DTD i validitu dokumentu. Parser bývá k dispozici ve dvou provedeních:

- knihovny pro různé programovací jazyky, které můžeme použít v našich programech
- jednoduché programy, kterým na vstup předáme XML dokument a na výstupu obdržíme případný přehled chyb v dokumentu.

Můžeme také využít parser zabudovaný v prohlížeči, který XML podporuje (většina webových prohlížečů, apod.).