Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky
Ústav přístrojové a řídicí techniky ČVUT v Praze, odbor automatického řízení a inženýrské informatiky
Technická 4, 166 07 Praha 6

JAZYK XSLT A SÉMANTICKÝ WEB

XSLT Language and Semantic Web

Josef Kokeš

Abstrakt: Basic semantic oriented languages and ontological schemes are discussed and transformation formulas are shown. Empiric approach to create semi-semantic Web is presented. Based on empiric approach, a simple solution of personal web pages is presented. This solution is intended as a machine-readable part of department's web portal.

Key words: XML, XSLT, sémantický web

1. Kritické zhodnocení současného vývoje

Již od roku 1996, kdy byly provedeny první pokusy o sémanticky standardizovaný popis webových zdrojů, bylo předpovídáno masové nasazení technologií, umožňujících automatizovanou excerpci a zpracování metadat obsažených na webu. Bohužel, jak se dnes ukazuje, skutečný vývoj jde cestou spíše opačnou: zatímco objem dat dostupných přes webové rozhraní exponenciálně roste, objem strojově zpracovatelných znalostí spíše stagnuje a propast mezi nimi se rozevírá.

Domnívám se, že existují dvě hlavní příčiny. První z nich – poněkud paradoxně - je významné zlepšení kvality webových vyhledávačů a indexerů. Ty jsou s to dodat výsledky s tak vysokou relevancí, v tak krátkém čase a (pro uživatele) natolik bezpracně, že velmi poklesla potřeba prohledávat web podle významu, tzn. sémanticky.

Druhou významnou příčinou nepochybně je, že pro sémanticky orientované technologie neexistují všeobecně rozšířené prostředky pro praktické použití, zejména ne pro vstup a pro výstup. Publikace a odborné práce se zaměřují spíše na rozšiřování možností stávajících jazyků a schémat, což ústí v rostoucí komplexitu. To z nich činí složité a obtížně uchopitelné nástroje, jejichž masová použitelnost je sporná.

Smyslem tohoto příspěvku je ukázat, že prvním krokem směrem k sémantickému webu by mohlo být vhodné používání zcela standardních nástrojů, jako jsou Excel a Internet Explorer.

2. Ontologie jako základ sémantického webu

Ve filosofii se ontologie chápe jako nauka (či soubor nauk) o "bytí", popřípadě jako univerzální soustava znalostí popisující objekty, jevy a zákonitosti světa. V informatice je ontologie specifikována jako "explicitní specifikace konceptualizace".

V současnosti rozeznáváme tři základní typy ontologií:

- **terminologické** – to jsou vlastně pokročilejší tezaury. Používané jsou v knihovnictví a oborech zaměřených převážně na textové informace.
- **informační** – představují rozvinutí databázových konceptuálních schémat. Zajišťují abstrakci a vyšší kontrolu integrity
- **znalostní** - reprezentace znalostí v rámci umělé inteligence. Objekty a relace mezi objekty jsou důsledně definovány pomocí formálního jazyka.

Základním prvkem sémantického webu je **konceptualizace** dat. Konceptualizací rozumíme vytvoření systému pojmů, modelujícího určitou část světa. Důležitým předpokladem sémantického webu je standardizovaný popis webových zdrojů, které ovšem mohou být prakticky jakékoliv (zvuk, video, text, obrázek, atd.). Protože konceptualizace musí být specifikována explicitně, vytváří se například pomocí vhodných „dat o datech“, neboli metadat, připojených k těmto webovým zdrojům.

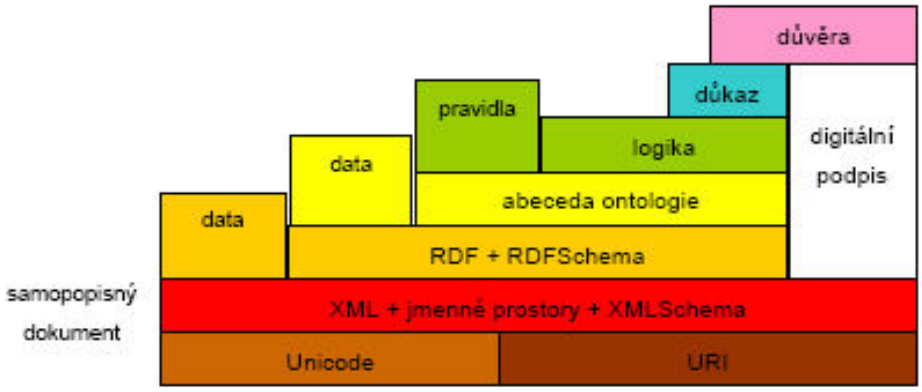
Historii pokusů o konceptualizaci webovou ontologii pomocí formálních jazyků lze podle [1] schématicky popsat takto:

- **SHOE** (*Simple HTML Ontology Extension*) – první jazyk, který vznikal pro specifické potřeby přidání sémantiky (významu) k informacím na webových stránkách. Vyvinutý týmem J. Hendlera na University of Maryland v roce 1996. Nevýhodou je vazba objektu na konkrétní HTML stránku identifikovanou pomocí URL.
- **Ontobroker** - vznikl přibližně ve stejné době jako SHOE na univerzitě v Karlsruhe. Má stejný koncept, ale důslednou centralizaci. Předpokládá existenci centrálního serveru.
- **RDF Schema** – První ontologický jazyk orientovaný na RDF (*Resource Description Framework*) – metadatový standard konsorcia W3C. Metadata jsou data vkládána do HTML stránky a to buď do hlavičky anebo jako samostatný doplněk k dokumentu. Obsahují ontologický popis informací na stránkách. Ontologické jazyky dodávají metadatům sémantiku (význam). Tento jazyk vznikl již v r.1999, relativně nezávisle na hlavním proudu "ontologického" výzkumu, přímo na půdě W3C.
- **DAML+OIL** – v polovině roku 2000 byl zahájen projekt DAML (*DARPA Agent Mark-up Language*), sponzorovaný vojenskou institucí DARPA.. Cílem bylo vytvořit sémantický jazyk pro RDF s větší vyjadřovací silou než má RDFS. Je překonán jazykem DAML+OIL.
- **OWL** - *Ontology Web Language*. Vzniká v současné době na základě zkušeností s DAML+OIL pod hlavičkou W3C Ontology Working Group. Z praktického hlediska je významné vyčlenění minimální podmnožiny tohoto jazyka: OWL Lite - to by mělo usnadnit implementaci programových nástrojů, která byla pro plnou verzi DAML+OIL (i pro plnou verzi OWL) velmi komplikovaná.

3. Na půlce cesty

Výše popsané formální jazyky sice jsou vhodné pro popis metadat a vytváření ontologií, ale jejich praktická využitelnost je sporná. Zejména proto, že se zatím ani neustálil jeden (nebo několik málo) procesních modelů, ani nejsou k dispozici prostředky pro práci s nimi.

Domnívám se, že daleko větší šanci na úspěšné nasazení mají méně ambiciózní postupy, založené na důsledném využití už existujících a rozšířených postupů. Jedná se zejména o „samo-popisné“ dokumenty založené na schématu XML. Přestože jazyk XML sám o sobě je definován pouze na syntaktické úrovni, lze obvyklými prostředky jednoznačně určit konkrétní význam značek. Důležité je, že XML lze využít pro zápis metadat. XML vlastně definuje gramatiku a využívá zápisu pomocí UNICODE, a tak může být v takovémto dokumentu uloženo cokoliv v jakémkoliv jazyce. Je tedy možné, nad XML vytvářet nadstavby ve vyšších, tentokrát už sémantických (významových) jazycích, jako je RDF. Pozici jazyka XML ukazuje obrázek převzatý z [2]:



Obrázek 1 Pozice XML v sémantickém webu

Odtud je již velmi blízko k popisu sémantiky, významu. Například tvrzení, že Jan Novák je zaměstnancem ČVUT, může zapsáno být v jazyce RDF a odtud již automatizovaně zpracováno. Forma zápisu v RDF se ovšem může lišit podle použitých prostředků. Standardní je „predikátový zápis“, ve kterém je vytvořena trojice **subjekt→predikát→objekt**, například

```
subjekt:   ČVUT
predikát: zaměstnání
objekt:    Jan Novák
```

Tentýž vztah ovšem lze zapsat i graficky, například

Obrázek 2 Predikátový vztah

Pro nás ovšem je zajímavé, že totéž lze také vyjádřit pomocí reprezentace zapsané v XML, například takto:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:s="http://www.description.org/schema/">
  <rdf:Description about="ČVUT">
  <z:Zamestnanec >Jan Novák</z:Zamestnanec>
  </rdf:Description>
</rdf:RDF>
```

Vlastní sémantickou informaci nese tučně vytištěná část, zatímco první token odkazuje na standard konsorcia W3C, podle kterého je RDF schéma zadáno.

Jako experiment jsem realizoval jednoduchou úlohu – webové stránky pracovníků odboru – pomocí standardních prostředků a s využitím technologie XML tak, aby bylo možno nad nimi vytvářet sémanticky orientované nadstavby. To ovšem bude předmětem až dalších prací.

Cílem bylo provedení následujícího experimentu: Co nejjednoduššími prostředky a pokud možno s minimalizací chyb, pořídit znalosti o jednotlivých pracovnících. Tyto znalosti automatizovaně přepracovat do takového XML, aby v něm byla zřejmá sémantika. A konečně, vytvořit prostředky, pomocí kterých se uvedené XML soubory budou prezentovat formou osobních webových stránek.

4. Vstup a prvotní zpracování dat

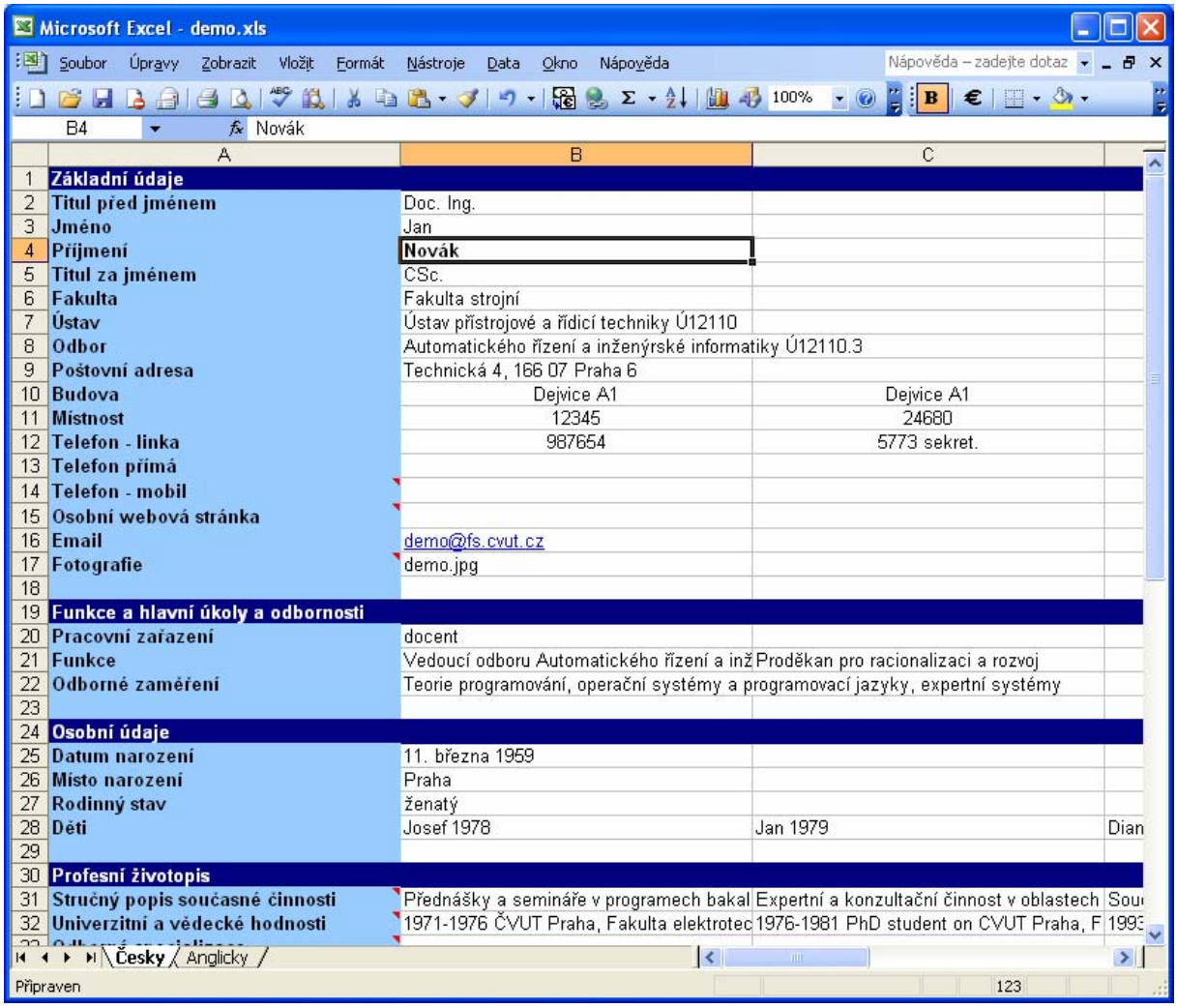
Data lze pohodlně zadávat pomocí specializovaného formuláře, pokud jej vytvoříme, přičemž můžeme současně aplikovat všechny myslitelné kontroly. Kontrolovat data hned na vstupu je nejlepší, nejspolehlivější a současně nejlevnější cestou, jak ověřit jejich bezchybnost. Všechny pokusy o následné kontroly dat jsou dražší (ve smyslu sumárních nákladů na vytvoření, validaci a následnou opravu). Pro vstup tak jednoduchých údajů, jak jsou osobní data, by ovšem programování specializovaného formuláře bylo neefektivní a navíc, takové řešení je naprosto neoperativní: jakákoliv změna je pracná a zdlouhavá.

Proto byla dána přednost tomu, vytvořit v tabulkovém procesoru (v našem případě Excel) jednoduchou tabulku, do které se údaje zapisují. Vzor tabulky byl rozeslán všem pracovníkům a po vyplnění byly všechny vyplněné tabulky umístěny do společného adresáře. Příklad vyplněného úseku tabulky je na obrázku na následující stránce.

Jak je vidět, jednotlivé údaje jsou řazeny do skupin (označeny nadpisem na tmavém pozadí). V daném řádku je vždycky ve sloupci A otázka (resp. sémantický význam), ve sloupci B je příslušná odpověď. Kde to má význam, jsou povolené vícenásobné odpovědi, které se zapisují do dalších sloupečků C, D,… atd. Výhodou vstupu přes tabulkový kalkulátor je, že je všeobecně rozšířený a obecně dobře zvládaný, tzn. respondentům práce s ním nečiní problémy. Navíc není velký problém v něm doplnit kontroly a validace.

Na tomto místě bych rád poznamenal, že Excel sám o sobě také umí vytvořit soubor XML. Můžeme se o tom přesvědčit z menu *Soubor-Uložit jako*, když vybereme *Typ souboru* „Datové soubory ve formátu XML“. Bohužel je třeba říci, že výsledný XML soubor je vytvořen z hlediska grafického formátování, tzn. prostřednictvím *tagů* XML obsahuje

všechny formátovací značky tak, aby se správně zobrazil jak v Excelu, tak i v jiných prostředích. Sémantika tam ale není nijak vyjádřena, a proto se nám tento způsob převodu na XML nehodí.



	A	B	C	
1	**Základní údaje**			
2	**Titul před jménem**	Doc. Ing.		
3	**Jméno**	Jan		
4	**Příjmení**	**Novák**		
5	**Titul za jménem**	CSc.		
6	**Fakulta**	Fakulta strojní		
7	**Ústav**	Ústav přístrojové a řídicí techniky Ú12110		
8	**Odbor**	Automatického řízení a inženýrské informatiky Ú12110.3		
9	**Poštovní adresa**	Technická 4, 166 07 Praha 6		
10	**Budova**	Dejvice A1	Dejvice A1	
11	**Místnost**	12345	24680	
12	**Telefon - linka**	987654	5773 sekret.	
13	**Telefon přímá**			
14	**Telefon - mobil**			
15	**Osobní webová stránka**			
16	**Email**	demo@fs.cvut.cz		
17	**Fotografie**	demo.jpg		
18				
19	**Funkce a hlavní úkoly a odbornosti**			
20	**Pracovní zařazení**	docent		
21	**Funkce**	Vedoucí odboru Automatického řízení a inž	Proděkan pro racionalizaci a rozvoj	
22	**Odborné zaměření**	Teorie programování, operační systémy a programovací jazyky, expertní systémy		
23				
24	**Osobní údaje**			
25	**Datum narození**	11. března 1959		
26	**Místo narození**	Praha		
27	**Rodinný stav**	ženatý		
28	**Děti**	Josef 1978	Jan 1979	Dian
29				
30	**Profesní životopis**			
31	**Stručný popis současné činnosti**	Přednášky a semináře v programech bakal	Expertní a konzultační činnost v oblastech	Sou
32	**Univerzitní a vědecké hodnosti**	1971-1976 ČVUT Praha, Fakulta elektrotec	1976-1981 PhD student on CVUT Praha, F	1993

Obrázek 3 Příklad vyplnění tabulku v Excelu

Já jsem namísto toho vytvořil jednoduchý program v jazyce JAVA, jehož účelem je číst jednotlivé řádky datového souboru Excel a tyto řádky přepracovat do formy XLS souboru. V tomto souboru jsou jednotlivé údaje o zaměstnancích hierarchicky strukturovány pomocí značek (*tagů*), jejichž význam je zřejmý.

Částečný výpis XML souboru, který odpovídá ukázce na obrázku „Příklad vyplnění tabulku v Excelu“, je uveden na následující stránce. Je z něj vidět, že všechny relevantní údaje jsou umístěny mezi *tagy* XML, které definují jejich sémantický význam. Současně je zřejmé, že *tagy* vytvářejí hierarchickou stromovou strukturu, jejímž kořenem je *<pracovník>* a která shrnuje všechny nashromážděné údaje o pracovníkovi. Údaje o jednom pracovníkovi jsou v jednom souboru XML.

Sestavení RDF schématu z takto strukturovaného souboru už je jen formální záležitostí a z nedostatku času jsem se jím nezabýval.

```
<pracovnik>
 <cz>
  <identifikace>
   <titul_pred>Doc. Ing.</titul_pred>
   <jmeno>Jan</jmeno>
   <prijmeni>Novák</prijmeni>
   <titul_za>CSc.</titul_za>
   <fakulta>Fakulta strojní</fakulta>
   <ustav>Ústav přístrojové a řídicí techniky Ú12110</ustav>
   <odbor>Automatického řízení a inženýrské informatiky Ú12110.3</odbor>
   <adresa>Technická 4, 166 07 Praha 6</adresa>
   <budova>Dejvice A1</budova>
   <budova>Dejvice A1</budova>
   <mistnost>12345</mistnost>
   <mistnost>24680</mistnost>
   <linka>987654</linka>
   <linka>5773 sekret.</linka>
   <email>demo@fs.cvut.cz</email>
   <fotografie>demo.jpg</fotografie>
  </identifikace>
  <odbornost>
   <zarazeni>docent</zarazeni>
        . . . . .
```

5. Výstupní transformace

Na první pohled by se mohlo zdát, že takto provedený soubor XML je popřením toho, co jsem tvrdil v úvodu. Nikdo sice nemůže zpochybnit, že tento soubor je strojově dobře čitelný (a pochopitelný), dokonce v něm lze jednoduše provádět i změny a opravy (protože to je ryze textový soubor). Ale z výše uvedené tabulky je také zřejmé, že jeho čitelnost a uchopitelnost lidským uživatelem je nízká. Jednoduše řečeno, text je nepřehledný a pro lidi špatně čitelný.

Naštěstí existuje poměrně snadná cesta, jak tento problém vyřešit. Tato cesta se nazývá **XSL**, *EXtensible Stylesheet Language*, tedy Rozšiřitelný jazyk pro šablony stylů.

Moderní webové programování se už dávno nespokojuje se statickými webovými stránkami, na kterých by se zobrazovaly ručně zapsané texty a obrázky. Pro moderní web je charakteristické, že odděluje formu od obsahu. **Obsah** webových stránek se dnes různými způsoby generuje, například z databází, jiných webových stránek, na základě interaktivní spolupráce s uživatelem a podobně. To vše s využitím moderních programovacích nástrojů, například PHP, JAVA, ASP a dalších. Naproti tomu **forma** webových stránek, jejich vzhled, je vesměs určena pomocí nějaké šablony. Zpočátku se používaly kaskádové styly, což je jednoduchá forma šablony, dnes jich existuje značné množství.

Pro moderní webové stránky je charakteristické, že na serveru se vygeneruje datový obsah stránky, připojí se k němu informace o formátování (například formou šablon) a obojí se odešle do internetového prohlížeče. Na straně klienta pak to je internetový prohlížeč, ve kterém se data naformátují podle příslušné šablony stylu (a také podle vlastností a nastavení prohlížeče) a výsledek se uživateli prezentuje na obrazovce.

XSL je jedním z mnoha jazyků, ve kterých lze popsat šablony stylů pro webové stránky. Uživatele to nemusí příliš zajímat, to je práce webového programátora. Zajímavé a důležité ale je, že součástí XSL také mohou být **transformace**. Transformace popisují, jakým způsobem se mají ze vstupního souboru XML vyextrahovat data a jak se mají naformátovat, aby je bylo možno prezentovat uživateli. Jazyk, který spojuje XSL a transformace, se souhrnně označuje **XSLT**.

Pro náš účel je důležité, že je možno poměrně snadno psát stylové šablony, které nejen určují, jak se má příslušná webová stránka zobrazit a jak má vypadat, ale současně také říkají, jakým způsobem se do ní mají načíst data z příslušného XML souboru. Přesněji řečeno, je to obráceně: XML soubor může obsahovat odkaz na XSLT šablonu, která říká, jak se tento soubor má zobrazit a zpracovat.

Pro ilustraci uvedu několik základních značek (*tagů*), které jsem použil k vytvoření šablony *styl.xsl* a shrnu je do následující tabulky:

`<xsl:template match="/">`	otevírací závorka šablony určuje, na které prvky se bude šablona aplikovat (zde je „/“, což znamená na všechny prvky). Touto závorkou musí začínak takdá šablona.
`</xsl:template>`	uzavírací závorka šablony je párová k otevírací závorce a každá šablona musí touto závorkou končit.
`<xsl:if test="výraz">`	otevírací závorka pro test znamená, že pokud je pravdivá hodnota výrazu „výraz“, provede se všechno, co následuje až po uzavírací závorku testu
`</xsl:if>`	uzavírací závorka testu
`<xsl:for-each select="určení">`	otevírací závorka cyklu určuje, že všechno co je mezi touto otevírací závorkou a příslušnou zavírací závorkou, se zopakuje pro každý jednotlivý element z „určení“
`</xsl:for-each>`	uzavírací závorka cyklu
`<xsl:value-of select="určení"/>`	namísto této závorky se do weboví stránky vloží data, definovaná hodnotou „určení“

Podobných značek existuje velké množství, ale jak vidno, jazyk XSLT je poměrně jednoduchý.

Obrázek 4 Příklad aplikace XSLT šablony na soubor XML

S využitím jazyka XSLT jsem vytvořil šablonu *styl.xsl*, kterou spolu se souborem *demo.xml* a s několika obrázky přikládám na CD. Aplikací šablony *styl* na soubor *demo* vznikne webová stránka se standardizovaným vzhledem, jak ukazuje obrázek.

Poděkování

Tento článek vznikl v rámci projektu „Systémy pro management znalostí a universitních informací“, dílčí část „Pilotní implementace vybraného produktu CMS pro řízení FS“, akce číslo 070572103.

Literatura

[1] Hanyáš, P.: Sémantický web. Webové stránky http://www.hanyas.net/seweb/index.php
[2] Hradský, J.: Jazyk OWL a sémantický web. Webové stránky http://www.hradsky.name/skola/bc/bp_utf.html#ch01