VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

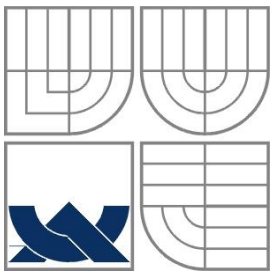
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SÉMANTICKÝ WEB V CMS SYSTÉMECH

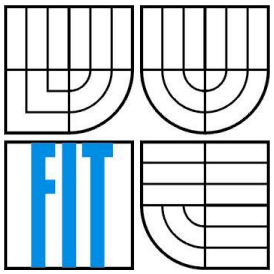
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Michal Vrána
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SÉMANTICKÝ WEB V CMS SYSTÉMECH

SEMANTIC WEB IN CONTENT MANAGEMENT SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Michal Vrána
AUTHOR

VEDOUCÍ PRÁCE — Ing. Radek Burget, Ph.D.
SUPERVISOR

BRNO 2010

Abstrakt

Tato diplomová práce se zabývá Sémantickým webem, jeho návazností na web současný a technologiemi, které ho vytvářejí. Dále se věnuje jeho současnému využití v praxi, podrobněji zkoumá nasazení v systémech pro správu webového obsahu a navrhuje sémantická rozšíření pro systém Kentico CMS.

Abstract

This master's thesis deals with the Semantic Web, its link to existing Web and the technologies, that produce it. It also deals with its current use in practice, further examines the deployment in web content management systems and proposes semantic extensions for the Kentico CMS.

Klíčová slova

Sémantický web, Web 3.0, RDF, RDFa, OWL, mikroformáty, CMS, Kentico CMS, ASP.NET, Drupal, WordPress, DublinCore, FOAF, XFN, hCard, SocialGraph API.

Keywords

Semantic web, Web 3.0, RDF, RDFa, OWL, microformats, CMS, Kentico CMS, ASP.NET, Drupal, WordPress, DublinCore, FOAF, XFN, hCard, SocialGraph API.

Citace

Vrána Michal: Sémantický web v CMS systémech, diplomová práce, Brno, FIT VUT v Brně, 2010

Sémantický web v CMS systémech

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radka Burgeta Ph. D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Vrána
18. 5. 2010

Poděkování

Rád bych zde poděkoval svému vedoucímu Ing. Radku Burgetovi Ph.D. za odborné a trpělivé vedení, své rodině a přítelkyni za podporu poskytnutou v průběhu studia a tvorby této práce.

Obsah

1 Úvod

Když v roce 1989 Tim Berners-Lee zformuloval první návrh na podobu dnes nejpoužívanější služby na Internetu, tedy WWW (World Wide Web), asi ani sám netušil, jakou revoluci tím způsobí. Předpokládal, že z hlediska zpracovávání a výměny vědeckých dokumentů a prací by se mohlo jednat o velkou věc, jen o tom bylo třeba přesvědčit i ostatní. Přimět je představit si, kam až by věci mohly dospět. Jak sám přiznává, toto bylo na celém úkolu nejtěžší [1]. Dnes o jeho myšlence nikdo nepochybuje. Ani on však zřejmě nevěděl, že web pronikne rovněž do životů stovek milionů obyčejných lidí celého světa a zásadním způsobem ho ovlivní.

Dnes se nás snaží se stejnou naléhavostí přesvědčit, abychom se posunuli o krok dále k vizi jménem „Sémantický web“. K vizi, kde jsou na webu volně dostupná data s jasně definovaným významem, která je možno dále využít k nalezení odpovědí na problémy lidstva, posunutí hranice vědění či v neposlední řadě také usnadnění běžného života.

Ve druhé kapitole této práce se podrobněji seznámíme s touto vizí, s tím, jak vznikala a čeho s ní bude možné dosáhnout, jak se změní současná podoba webu, na jakých technologiích bude tento „nový“ web postaven a které problémy by měl řešit. Dále zde bude představen pojem ontologie jako základní koncept pro popisování sémantických informací.

V další části jsou detailně popsány technologie pro tvorbu sémanticky anotovaného obsahu jako RDF, RDFa, OWL, mikroformáty či mikrodata. Je zde popsán také jazyk XML, který tvoří základ pro tuto anotaci.

Navazuje kapitola s příklady již dnes existujících stránek a projektů využívajících sémantické technologie. Jsou zde informace o tom, jak Google, Bing a Yahoo upravují své vyhledávací systémy, jak se mění publikování obsahu, ať už pro bloggery či velké redakce, jak Sémantický web prorůstá do sociálních sítí. Velice důležité je zpřístupnění a provázání dat v rámci iniciativy LinkedData.

Pátá kapitola popisuje CMS systémy, jejich funkce a vlastnosti. Konkretizuje členění těchto systémů do kategorií. Podrobněji se věnuje především webovým CMS. Jsou zde vybrány nejpoužívanější systémy, které jsou následně popsány z hlediska podpory sémantických prvků.

Obsahem šesté kapitoly je návrh a popis implementace sémantických rozšíření pro systém Kentico CMS. Je představena architektura tohoto systému a také vývojová platforma ASP.NET. Je zde též porovnání vytvořených rozšíření s ostatními CMS systémy.

V závěru je pak zhodnocení dosažených výsledků a můj osobní pohled na budoucnost Sémantického webu.

2 Sémantický web

Informace jsou na dnešním webu prezentovány především ve formě textu a multimediálních dat. Tato forma je sice srozumitelná pro člověka, ale způsobuje obtížné vyhledávání a strojové zpracování informací ukrytých v těchto datech. Abychom tuto situaci změnili, je zapotřebí použít systém, který by umožnil popis obsahu a práci s těmito popisy. Vzhledem k obrovskému počtu dokumentů na webu, se objevuje přirozená potřeba co nejvíce tento popis, neboli též anotaci, zautomatizovat. O to by se mohly starat právě CMS systémy.

Pro správné rozpoznání a hlavně pochopení nalezených informací, musí mít počítač k dispozici formálně definované znalosti o rozpoznávaném světě. K tomuto určení sémantiky se používají tzv. ontologie.

„*Sémantický web není odděleným webem, je to rozšíření toho stávajícího. Informace v něm budou mít jasně definovaný význam, což umožní počítačům a lidem lépe spolupracovat*."

Tim Berners-Lee [2]

2.1 Současný World Wide Web

Jak již vyplývá z uvedeného prohlášení, Sémantický web nestaví „na zelené louce". Je to rozšíření webu stávajícího, a proto je důležité si říci něco o celém jeho vývoji, technologiích a standardech, na kterých stojí.

2.1.1 Internet

Pro neodbornou veřejnost je Internet synonymem pro web. Věty typu „Najdeš to na netu" a „Najdeš to na webu" mají v češtině opravdu téměř shodný význam. Znalí však vědí, že Internet je pouze infrastrukturou pro výměnu dat, kterou využívá mnoho služeb, jako je např. email, FTP, IM[1] a v neposlední řadě také web.

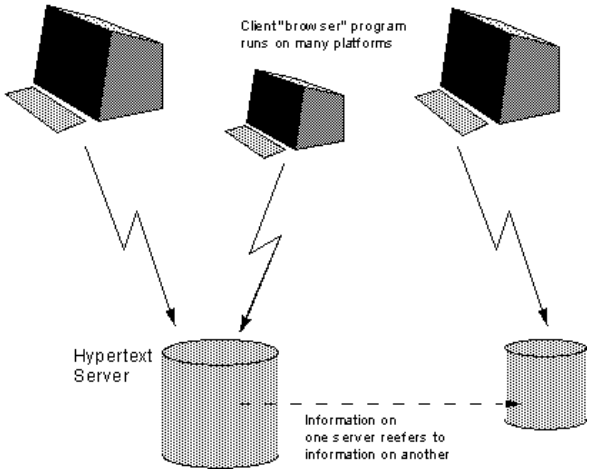
Internet vznikl z počítačové sítě ARPANET, za jejímž vznikem v roce 1969 stála agentura amerického ministerstva obrany ARPA[2]. Počet uzlů připojených do této sítě postupně rostl, vznikly fundamentální protokoly Internet Protocol (IP), Transmission Control Protocol (TCP) a Domain Name System (DNS) a vysoko-úrovňové protokoly jako FTP, SMTP, Telnet a zejména pak HTTP. Právě posledně jmenovaný je pro nás ten nejdůležitější. Slouží totiž primárně k výměně hypertextových webových dokumentů.

[1] Instant messaging – služby pro okamžitou komunikaci jako ICQ, MSN, Jabber atd.
[2] Advanced Research Projects Agency, dnes již přejmenována na DARPA

2.1.2 Vznik a základní technologie

V roce 1989 Tim Berners-Lee, v rámci svého působení ve výzkumné laboratoři CERN ve Švýcarsku, navrhnul svou první verzi WWW. Podstatou bylo použití hypertextu jako prostředku pro organizaci distribuovaného systému dokumentů. Pomocí hypertextu lze přes odkazy vzájemně provázat libovolné dokumenty nebo jejich části. Zbývalo vymyslet, jakým způsobem popisovat jejich umístění, jak dokumenty z tohoto umístění získat a také jazyk, který by je popisoval [3].

Jako mechanizmus pro adresování dokumentů vznikl sytém Uniform Resource Locator (URL). URL adresa je řetězec, který se skládá ze schématu, doménového jména, portu, specifikace souboru a možných parametrů. Schéma určuje, jaký protokol bude použit pro získaní dokumentu. Nejčastějším je právě protokol HTTP. Díky němu je možné vytvořit požadavek na určitý dokument, který se nachází na cílovém serveru. Když takový požadavek dorazí na server, je zpracován a požadovaný dokument je vrácen klientovi. Tyto servery původně Berners-Lee nazval „hypertext server“, ale dnes se používá spíše termín „web server“. Jeho původní model lze vidět na následujícím obrázku.



Obrázek 1: Původní model klient/server distribuovaného hypertext systému [4]

Obvykle je obdržený dokument na klientovi zpracováván pomocí aplikace prohlížeče, jež se stará o správnou prezentaci a vykreslení na obrazovce uživatele. Způsob, jak má být dokument vykreslen, se určuje pomocí jazyka HTML, popř. některého z jeho velmi podobných následovníků.

Jazyk HTML, neboli HyperText Markup Language, je nejdůležitějším jazykem pro tvorbu webových stránek. Jedná se o jazyk značkovací, tedy pomocí definované množiny značek, kterým se také říká tagy, umožňuje označit určitou část textu a tím jí dodat nějakou vlastnost či funkcionalitu. Takovými značkami jsou např. odkazovací tag `<a>`, umožňující provazování dokumentů, tag `<h1>` definující nadpis či `<img>` pro vložení obrázku.

Většina značek jazyka je orientovaná na vzhled či definování struktury dokumentu, avšak pomocí některých tagů, či jejich atributů, jsme schopni vyjádřit i slabou sémantickou informaci [5]. V HTML specifikaci 2.0 se objevil element `<meta>` a atribut `rel`. Meta element obsahuje ve formě klíč-hodnota metadata o zobrazované stránce. Např. ve své době se hojně využíval klíč `keywords`, který měl pomoci vyhledávačům s nalezením relevantní stránky. Z důvodu jeho masivního zneužívání vkládáním populárních, ale nerelevantních klíčových slov se jeho význam naprosto devalvoval a dnešní vyhledávací systémy ho v podstatě ignorují. Druhým zmíněným prvkem je atribut `rel` použitelný u tagů `<a>` a `<link>`. Tento atribut umožňuje popsat vztah, jaký má odkazující dokument k dokumentu odkazovanému. Pro úplnost zmíním, že existuje i atribut `rev`, který popisuje tento vztah v opačném směru.

Na konci roku 1990 dokončil Berners-Lee práci na protokolu HTTP, jazyku HTML, prvním prohlížeči[3], webovém serveru a webové stránce. Položil tak základní stavební kameny pro to, aby web mohl začít fungovat jako opravdu celosvětová síť. A skutečně, v první polovině 90. let se web začíná rozšiřovat. Vznikají programy a prohlížeče pro různorodé systémy, připojují se další univerzity a vědecká pracoviště. V roce 1994 vzniká standardizační organizace W3C[4], která má dohlížet na další vývoj webu. Obrovský růst zažívá WWW v druhé polovině 90. let, kdy se komercializuje – vstupují do něj privátní, komerční firmy za účelem se prezentovat pomocí vlastních webových stránek a samozřejmě na něm obchodovat. To všechno přináší překotný vývoj. Stávající specifikace HTML jsou nedostačující pro atraktivní stránky. Výrobci prohlížečů si přidávají vlastní proprietární značky a rozchází se ve způsobu vykreslování prvků stránky. W3C nestíhá a nové verze HTML přichází na svět se zpožděním. Dá se říci, že z této doby se vzpamatováváme dodnes, přinesla ovšem i poučení, že je třeba dodržovat standardy, zachovávat vzájemnou kompatibilitu a na vývoji spolupracovat [3].

2.1.3 Od Webu 1.0 k Webu 3.0

Již v prvním desetiletí po svém příchodu změnil web způsob, jak spolu lidé komunikují, obchodují, sdílejí a předávají si informace a znalosti. To vše navíc jednoduše, rychle, poměrně levně a nehledě na geografické vzdálenosti. Toto období lze charakterizovat tím, že informace plynou obecně z jednoho místa, jímž může být webová stránka či server, k mnoha uživatelům. Jsou jasně rozdělené role, kdo obsah tvoří a kdo ho konzumuje. Z dnešního pohledu to můžeme označit termínem „Web 1.0“.

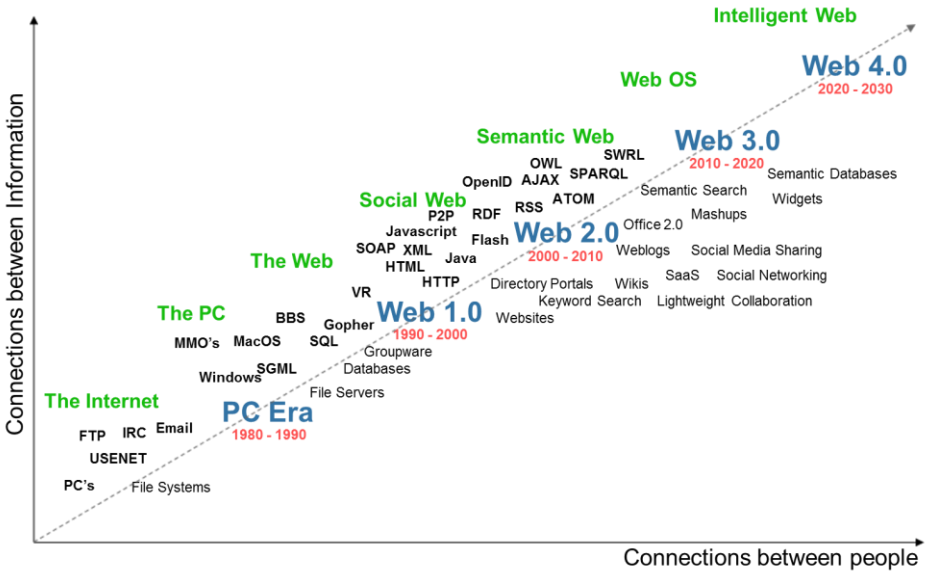
V průběhu let a dalšího rozšiřování webu se začíná měnit i uvažování lidí o něm. Ve vyspělých zemích již lidé, obzvláště mladší generace, považují Internet a web za samozřejmost, tráví na něm velké procento svého času, jak pracovního, tak i soukromého. Dle studie Wave 4 z roku 2009 [6] čítá aktivní internetová společnost 625 mil. lidí. Stále častěji a více komunikují s ostatními přes fóra,

[3] Jmenoval se WolrdWideWeb
[4] Wold Wide Web Consortium

blogy, chaty nebo wiki-systémy. Vznikly nové weby, které začaly masivně využívat internetové komunity. Došlo k posunu v tom, jakým způsobem lidé web používají. Tim O'Raily to v roce 2004 pojmenoval jako „Web 2.0" [7]. Ačkoliv dodnes neexistuje přesná definice co Web 2.0 je, je jasné, že nejdůležitější složkou takového webu jsou jeho uživatelé, kteří sami generují obsah. Místo „webu dokumentů" je zde jakýsi „web lidí". Pokud mezi těmito uživateli dochází k sociální interakci a mohou definovat vzájemné vztahy, často se potom o takových webech hovoří jako o sociálních sítích. Mezi ty patří například Facebook, MySpace, Orkut a další. Je důležité zdůraznit, že Web 2.0 není náhradou za původní web, jde pouze o jeho rozšíření se zachováním původních prvků.

Netrvalo dlouho a objevil se termín „Web 3.0". Za ten je obecně považován právě Sémantický web. Paradoxem je, že svou představu o sémantickém webu prezentoval sám Berners-Lee již v roce 2001 článkem [2] v časopise Scientific American, tedy ještě předtím než vůbec O'Raily přišel s termínem Web 2.0. Věci se však dostávají více do pohybu až poslední dobou. Co si pod tímto pojmem představuje sám Berners-Lee? Citujme z původního článku: „Sémantický web přidá strukturu ke smysluplnému obsahu stránek. Vytvoří prostředí, kde softwaroví agenti procházející stránku za stránkou mohou bez váhání vykonat složité úkoly zadané uživatelem."

Progresivní vývoj webu a jeho technologie znázorňuje i následující diagram, na kterém jsou zachyceny éry webu s ohledem na bohatost sociálních vazeb a propojenost informací a technologií.



Obrázek 2: Vývoj webu [8]

2.2 Vyhledávání

Přesto, že je web poměrně mladý (sotva 20 let), je na něm obrovské množství informací. Mnozí dokonce v nadsázce říkají, že je na něm všechno, ovšem problémem je to najít. V počátečních dobách, kdy ještě neexistovalo tak mnoho stránek, byly weby tříděny do různých kategorií a lidé je pak v tzv. adresářích či katalozích mohli snadno nalézt. Toto třídění probíhalo zprvu manuálně, tudíž poměrně kvalitně. Ovšem počet stránek začal následně téměř exponenciálně růst a toto třídění nebylo již dále možné. Hlavní slovo v hledání informací na webu převzaly a dodnes mají vyhledávače. Ty většinou pracují na principu vyhledávání podle klíčových slov, tedy naleznou stránky obsahující zadané slovo. Jejich roboti procházejí celý web a stránky si indexují. Tyto indexy dnes obsahují miliardy stránek. Problém by se tedy dal přeformulovat z „jak nalézt stránku“ na „jak nalézt relevantní stránku“.

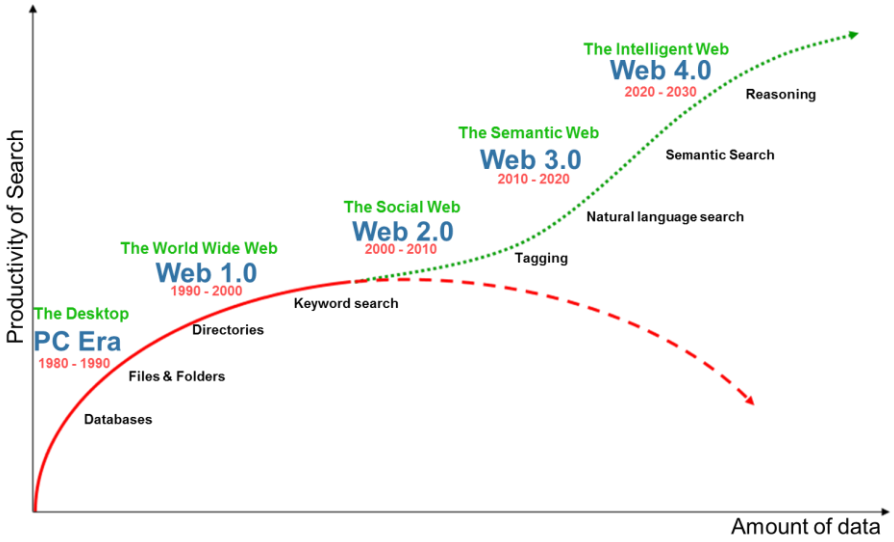
Pro stroje je úkol dolování informací z webových stránek značně obtížný přesto, že jejich obsah je velmi často generován ze strukturované databáze. Tato struktura se však při převodu do prezentační podoby vytrácí. Při vyhledávání mohou nastat situace, kdy po zadání dotazu dostaneme třeba i desítky milionů výsledků. To je pochopitelně těžko využitelné, relevance většiny z nich je malá. Popř. může nastat situace, že vyhledávač nenajde žádný nebo žádný relevantní výsledek. To může být způsobeno tím, jak jsme zadali klíčová slova. Vyhledávače jsou citlivé na to, v jaké podobě jsou slova zadána. Např. uvažovat synonyma k zadanému slovu jim činí problém. Další poměrně obvyklou situací je, že k získání požadované informace je potřeba prohledat několik různých zdrojů a z nich celou informaci teprve zkompletovat [5].

Díky technologickému pokroku v různých statistických, pravděpodobnostních a heuristických algoritmech, dokázaly vyhledávače udržet krok s růstem webu a umožňují nalézt poměrně relevantní stránky pro nespecifické dotazy, avšak informaci, kterou člověk skutečně hledá, si v nalezených stránkách musí stále najít sám. V poslední době se však zdá, že toto by se mohlo alespoň částečně změnit. Konkurenční prostředí nutí firmy přicházet s novými inovacemi také v této oblasti. Např. vyhledávač Bing[5] společnosti Microsoft se deklaruje jako „decision engine“ nebo WolframAplha[6] jako „computational knowledge engine“. U Bingu jde o to, že se snaží přímo do výsledků vyhledávání vkládat různé doplňující informace, které mají uživateli usnadnit hledání a pomoci mu „se rozhodnout“. Těmito informacemi mohou být obrázky, hodnocení produktů, srovnání cen, dokáže odpovědět na to, jaké je bude počasí, jak moc zatížená je doprava ve městě, nebo kdy letí jaké letadlo. Google si chce samozřejmě udržet vedoucí roli na poli vyhledávání a tak rychle zareagoval podobnými možnostmi. WolframAlpha pracuje trochu jinak, ostatně má i jiný cíl. Vezme uživatelův dotaz a snaží se mu co nejlépe a v kontextu porozumět a zpracovat ho v rámci velkého množství

[5] http://www.bing.com
[6] http://www.wolframalpha.com/

modelů a algoritmů z různých, povětšinou vědeckých oborů. Výsledkem je přímo odpověď na původní otázku popř. hlášení, že dané otázce nerozumí, nebo nedisponuje potřebnými daty. Jako příklad otázek, na které umí odpovědět, si můžeme uvést např. „Jaký je HDP Německa“, „Porovnej HDP Francie a Německa v minulém roce“, „Jaká je průměrná délka života“ atd. Otázky se však nekladou v zcela přirozeném jazyce.

Tomuto se samozřejmě nedá říkat Sémantický web. Nevyužívá se v masové míře jeho technologií, alespoň zatím ne. Jde spíše o využití dat z různých portálů a jejich API. Službám, které se takto navzájem propojují, aby rozšířili svou funkcionalitu, se také říká „mash-upy“. Důležitým výsledkem je ovšem obecná poptávka po takovýchto dostupných datech. Jak se již ukázalo, získávání těchto informací „shora“, tedy složitými technikami umělé inteligence a pokusy o porozumění přirozenému jazyku, nevede k úspěchu. Je třeba začít „zdola“ přidáváním sémantiky k obsahu webu tak, aby byl snadno zpracovatelný stroji. To je cesta Sémantického webu, Webu 3.0. Na následujícím diagramu je možné vidět předpoklad, jak by se měl změnit způsob, jakým na webu vyhledáváme.



Obrázek 3: Budoucnost vyhledávání na webu [9]

Jelikož se tato část týká vyhledávání, dovolím si zde ještě menší zmínku o dvou inovativních produktech, které krátce po svém uvedení na trh neunikly pozornosti obřích společností Google a Apple a byly odkoupeny. Jedná se o „social search engine“ Aardvark[7], který pracuje na principu, že se zeptáte na libovolnou otázku, systém sám vyhodnotí, který z vašich kontaktů je nejlépe způsobilý k odpovědi a tomu otázku položí. Druhým projektem je Siri[8], což je aplikace pro iPhone, která si klade za cíl být „virtuálním osobním asistentem“, kterému mluveným slovem řeknete jaký úkol (z omezené množiny) vykonat a on to zařídí díky tomu, že zná kontext a sémantiku dat.

[7] http://vark.com/
[8] http://siri.com/

2.3 Vize sémantického webu

Představme si vizi Sémantického webu v její plné šíři tak, jak ji uvedl Berners-Lee ve svém průvodním článku. Volný překlad z [2].

„Petrovi hraje v pokoji hudba a do toho mu zvoní telefon. V momentě kdy ho zvedne, vyšle mobil do všech domácích zařízení příkaz pro ztlumení hlasitosti. Volá jeho sestra Lucie z ordinace doktora: „Máma potřebuje prohlídku u specialisty a taky zařídit návštěvy u fyzioterapeuta. Asi jednou za čtrnáct dní. Nechám svého agenta zařídit schůzku.“ Petr souhlasí.

V ordinaci doktora Lucie instruovala svého sémantického agenta přes kapesní počítač. Agent okamžitě zjistil informace o předepsané léčbě od agenta doktora, vyhledal několik fyzioterapeutů, vybral ty spadající pod mámino pojištění, kteří jsou ve vzdálenosti do 20 mil od jejího domu a mají výborné hodnocení. Agent pokračoval hledáním vhodného času schůzky. Automaticky začal zjišťovat, kdy mají všechny strany volné místo ve svých kalendářích.

Za pár minut představil nalezený plán. Ale Petrovi se nelíbil, musel by s mámou jezdit přes celé město v době dopravní špičky. Dal tedy pokyn svému agentovi, ale tentokrát stanovil přísnější parametry hledání. Lučin agent, na základě plné důvěry, poskytl Petrovu agentovi veškeré doposud získané informace.

V okamžiku zde byl nový plán – bližší místo a dřívější čas. Ale byla zde dvě upozornění. První z nich říká, že Petr bude muset přeplánovat dvě ze svých méně důležitých schůzek. Ověřil si, že to není problém. Druhé upozornění se zmiňovalo o tom, že pojišťovna nemá nalezeného fyzioterapeuta uvedeného ve svých dokumentech, ale že byl ověřen pomocí jiného způsobu. Lucie plán bez váhání schválila a vše bylo domluveno.“

Tento scénář by se mohl na první pohled zdát jako science-fiction, ale není tomu zcela tak. Jednotlivé části tohoto úkolu (automatické vyhledávání, komunikační rozhraní, plánování a usuzování v rámci specifikovaných omezení) jsou do značné míry již vyřešeny [10]. Problém je tedy spíše v nasazení technologií, integraci a standardizaci. Dokud nepokročíme v tomto směru, nelze přesně říci, zda je tato vize[9] reálná ve velkém měřítku celého webu.

2.4 Metadata

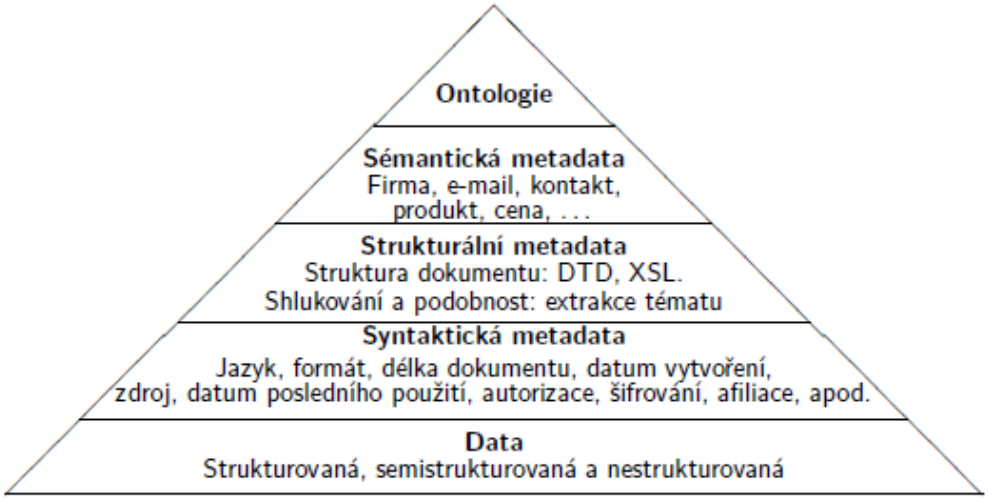
Obvyklý popis metadat jako „data o datech“ je sice pravdivý, ale poměrně zjednodušující. Metadata mohou obsahovat informace různého charakteru. Mohou vypovídat o obsahu dat, o způsobu uložení dat nebo způsobu spravování. Jindy jde zase o informace, které přímo nesouvisí s obsahem dat, například datum vytvoření dokumentu, umístění apod.

[9] Sémantický web chápaný jako prostor pro automatické agenty samostatně vykonavajících úkoly je také nazýván pojmem „executable web“.

Podstatou významu metadat ve vztahu k sémantickému webu je, že metadata jsou také data a jako taková se mohou sama stát zdrojem. Mohou tedy obsahovat informace jak o sobě samých, tak o jiných zdrojích.

Metadata lze rozdělit od několika kategorií [11]:

- Syntaktická metadata – obsahují podrobnosti o zdroji dat (dokumentu). Tato skupina metadat slouží převážně ke katalogizaci nebo kategorizaci.
- Strukturální metadata – se zaměřují na strukturu dokumentu. Tyto informace lze využít při ukládání, zpracování nebo prezentaci dokumentu. Zjednodušují vyhledávání.
- Sémantická metadata – popisují kontextově relevantní informace vzhledem k určité doméně. Užitím sémantických metadat lze získat smysluplnou interpretaci dat při zachování možnosti efektivní spolupráce systému na vyšší úrovni.
- Ontologie – představují nejvyšší formu metadat a současně jsou klíčovým principem sémantického webu. Více viz samostatná kapitola.



Obrázek 4: Typy metadat [11]

2.5 Co je to sémantika

Jak již sám název napovídá, je pro fungování Sémantického webu nejdůležitější právě ona sémantika. Základním a nejjednodušším výkladem slova sémantika je „význam“. Jak bylo představeno v kapitole o vizi Sémantického webu, jeho významnou součástí by měly být programy, softwaroví agenti, kteří budou schopni číst z různých webových zdrojů strojově srozumitelné specifikace o sémantice informací, porozumět jim a na základě toho být schopni vykonávat složité úlohy. Stejně tak bude třeba, aby si agenti byli schopni vyměnovat informace mezi sebou.

Rozeznáváme několik typů sémantiky [11]:

- Implicitní – jedná se o nejjednodušší, intuitivní chápání významu založeném na konsensu lidí. Příkladem může být chápaní značek jako `cena`, `adresa` v XML. Nikde v XML dokumentu, DTD nebo XML schématu nemusí být uveden význam těchto značek a přesto ho lze odvodit. Přesný význam je však z obecného hlediska mnohoznačný (např. jaká měna, s/bez daně).
- Explicitní (neformální) – v tomto případě je význam explicitně vyjádřen např. pomocí slovníku nebo textového dokumentu, avšak z principu možné nejednoznačnosti přirozeného jazyka není tento popis sémantiky strojově zpracovatelný.
- Formální pro zpracování člověkem – explicitní sémantika vyjádřená formálním jazykem, nicméně je určena pouze pro zpracování člověkem. Lze ji uvažovat jako formální dokumentaci, nebo jako formální specifikaci významu.
- Formální pro strojové zpracování – Explicitní, formálně vyjádřená sémantika určená pro stroje umožňuje přímé zpracování včetně automatického porozumění novým pojmům. Jejich sémantika by se automaticky odvodila z existujících znalostí. Jak toho docílit je předmětem zkoumání.

2.6 Ontologie

Pojem ontologie pochází původně z filosofie, kde označuje nauku o jsoucnu a bytí. V oblasti informačních technologií se však používá v trochu jiném smyslu. Definice praví:

„*Ontologie je formální a explicitní specifikace sdílené konceptualizace*“. [12]

Tato definice si jistě zaslouží další vysvětlení. Konceptualizací je myšleno modelování nějaké části reálného světa. Explicitní je použito ve smyslu jednoznačně definované. Formální znamená popsané nějakým konkrétním jazykem s definovanou syntaxí. A sdílená je vyřčení smyslu, že ontologie nevytváříme pro sebe sama, ale pro porozumění s okolím. Ontologie mohou za tímto účelem používat lidé, ale nás bude v této práci zajímat především porozumění mezi počítačovými systémy.

Ontologie můžeme dělit do kategorií podle různých hledisek.

Podle oboru [12]:

- Terminologické ontologie (lexikální) – jde o seznam termínů, jakýsi slovník (tezaurus). Jsou obvykle specifické pro určitou oblast, např. knihovnictví.
- Informační ontologie – jsou nadstavbou nad informačními databázemi. Umožňují vytvářet pokročilá schémata, vyšší úroveň kontroly integrity.
- Znalostní ontologie – koncepty jsou formálně definovány logickými formulemi. Navazují na oblast reprezentace znalostí z oboru umělé inteligence.

Podle zdroje konceptualizace (toho co modelují) [12]:

- Generické ontologie – také označovány jako ontologie vyššího řádu. Modelují zákonitosti a vztahy mezi obecnými pojmy. Např. Dublin Core.
- Doménové ontologie – jsou často používaným typem. Zabývají se popisem pojmů z určité ohraničené oblasti. Např. podnikové, lékařské apod.
- Aplikační ontologie – upraveny pro konkrétní aplikaci.

2.6.1 Struktura ontologií

Ačkoliv jsou různé typy ontologií, základní strukturou se však příliš neliší, tvoří je stejné stavební prvky [12]:

Třídy

Někdy také označovány jako koncepty. Jsou to hierarchicky uspořádané množiny konkrétních objektů. Přirovnat je můžeme k třídám v objektově orientovaném programování, ale na rozdíl od nich nezahrnují žádný procedurální kód. Jejich hierarchie reprezentuje dědičnost, ta může být jednoduchá se stromovou hierarchií, či vícenásobná s hierarchií síťovou.

Instance, individua, objekty

Jsou označením konkrétního objektu z reálného světa, např. osoby, věci apod. Mohou být prvky nějaké třídy, tj. instancemi, ale také nemusí. Potom se jim říká individua. Druhá situace není ve všech ontologických jazycích podporována.

Relace, vlastnosti

Vztahy se v ontologiích vyjadřují pomocí relace. Ta je *n*-ární podle toho, kolik individuí je ve vztahu. Pokud je relace pouze mezi dvěma objekty, tedy binární, pak se také nazývá vlastnost nebo slot. Nemají žádnou vazbu na třídu, pouze na objekty v relaci.

Meta-sloty, omezení

Relacím (slotům) lze také dávat vlastnosti. Ty se někdy označují jako meta-sloty. Tak jsme schopni například vyjádřit podřízenost nebo nadřazenost relací. Např. má předka a má otce. Dále lze slotům specifikovat omezení jako definiční obor, obor hodnot či kardinalitu.

Primitivní hodnoty, datové typy

Do relací může vstupovat jednak objekt, pak se slot nazývá objektový a jednak i primitivní hotnota. Slot je pak tzv. dato-typový. Hodnotou může být např. řetězec nebo číslo.

Axiomy

V ontologiích se mohou vyskytovat také logické, výrokové formule. Slouží např. pro vyjádření disjunkce či konjunkce tříd. Definice axiomů bývá obvykle zahrnuta do definice tříd.

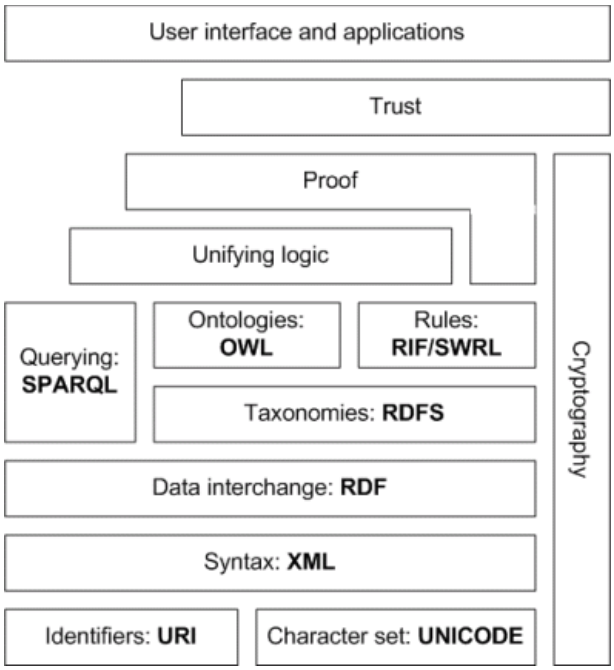
2.7 Problémy Sémantického webu

Je potřeba se na Sémantický web dívat také kriticky. Tato vize byla představena již v roce 2001 a musíme si přiznat, že od té doby jsme se k ní příliš nepřiblížili. Čím je to způsobeno? Existuje zřejmě několik důvodů. Tím prvním bude samotná vize. Jedná se o myšlenku vpravdě revoluční a její představení vyvolalo přehnaně optimistická očekávání. Stav, kdy budou počítače „všemu“ rozumět a vyřizovat za nás i komplikovanou agendu, se zdál být najednou na dosah ruky. Opak se ukázal být pravdou a znovu se potvrdilo, že sémantika je prostě náročná a problematická. Názorně se ukázala souvislost Sémantického webu a oboru umělé inteligence, kde se tento jev „přehnaných očekávání“ objevuje v cyklech již po několik desetiletí. Nepochybnou skutečností je, že svět nikdy nepůjde přesně popsat v celé své šíři, složitosti a dynamice.

Druhým důvodem může být standardizační proces na půdě W3C. To je poměrně zdlouhavá záležitost, musí dojít ke konsenzu mezi členy, kteří mají často rozdílné názory. Celkově dnes existuje v souvislosti se Sémantickým webem mnoho standardů a stále vznikají další. To značně snižuje ochotu vývojářů učit se nové technologie, protože ty se vlastně vůbec nemusí prosadit. S tím jde ruku v ruce nedostatek nástrojů a produktů, které by využily alespoň existující výdobytky Sémantického webu. V podstatě se jedná o tradiční paradox – vývojáři neznačkují data, protože uživatelé nemají programy, které by toho využily a takové programy nevznikají, protože není na webu dostatek kvalitních dat. Tento stav se ale postupně mění k lepšímu, tak se nechejme překvapit, co nám budoucnost přinese.

3 Technologie sémantického webu

Pojmem Sémantický web je souhrnně označováno několik různých technologií. Jak již bylo zmíněno v předcházející kapitole, není to oddělená část webu, ale rozšíření toho současného. Je proto logické, že využívá jako základ některé z již existujících standardů. Nad nimi pak definuje několik vrstev technologií specifických pro Sémantický web. To, že se nejedná o jediný standard má několik výhod. Především je dosaženo potřebné flexibility při přijímání potřebných změn na půdě W3C a celkově se usnadňuje jejich nasazení a použití.



Obrázek 5: Vrstvový model sémantického webu [13]

Na obr. 5 je možné vidět vrstvový model Sémantického webu. V dolní části je jazyk XML umožňující definovat strukturu a syntaxi dokumentu, univerzální kódování UNICODE, dále pak URI poskytující prostor pro jednoznačnou identifikaci zdrojů. Nad nimi je pak formalismus pro grafovou reprezentaci dat RDF. RDFS, neboli RDF Schema, jazyk postavený na RDF, umožňující popisovat hierarchické struktury ontologií a jejich vztahy. Dále OWL, což je mocnější jazyk pro popisování ontologií. Jazyk SPARQL pro dotazování v RDF datech. Zkratky RIF a SWRL označují ještě neschválené návrhy rozšiřující možnosti pravidel. Nad nimi se pak nachází vrstva logiky, vrstva odvozování a dokazování a velmi důležitá vrstva důvěryhodnosti. Ta může k verifikaci zdrojů využívat elektronických podpisů, doporučení od ověřených agentů a různých druhů certifikačních metod. Poslední vrstvou je samotné uživatelské rozhraní, přes které uživatelé k Sémantickému webu přistupují.

Tyto technologie budou podrobněji popsány v následujících podkapitolách.

3.1 XML

XML (eXtensible Markup Language) je značkovací jazyk, vycházející stejně jako HTML z obecného jazyka SGML. Ten se ukázal jako příliš složitý a XML je jeho zjednodušenou, více restriktivní podobou. Lze v něm vytvářet konkrétní jazyky, tzv. aplikace, specifické pro různé domény a typy dat. To jej činí velmi vhodným pro elektronickou výměnu dat nejenom v prostředí Internetu. XML je pro sémantický web zcela zásadní, jelikož je na něm vystavěna většina jeho technologií.

Práce na XML započala v organizaci W3C v roce 1996 a v roce 1998 byla představena verze XML 1.0. Stojí za to si vyjmenovat některé původní cíle návrhu XML [14]:

1. XML bude přímočaře použitelné na Internetu.
2. XML bude podporovat široké spektrum aplikací.
3. XML bude kompatibilní s SGML.
4. Bude jednoduché napsat program zpracovávající XML dokumenty.
5. XML dokumenty by měly být čitelné a relativně srozumitelné pro člověka.
6. Vytvořit XML dokument bude jednoduché.

Dnes již můžeme konstatovat, že tyto cíle byly splněny a XML skutečně proniklo snad do všech oblastí informačních technologií.

```
<univerzita>
  <nazev>VUT</nazev>
  <fakulta jmeno="FIT">
    <student>
      Michal Vrána
     </student>
  </fakulta>
</univerzita>
```

```
<h1>Univerzita VUT<h1>
<h2>Fakulta FIT</h2>
Studenti:
<ul>
  <li>Michal Vrána</li>
</ul>
```

Tabulka 1: Ukázkový kód pro srovnání XML a HTML

V XML není na rozdíl od HTML definována pevná sada značek, ale umožňuje vytvářet vlastní specifické značky, označující určitý druh informace. Navíc dodržuje princip SoC (Separation of Concerns) a zcela odděluje vzhled od struktury. Neříká nic o tom, jak budou jednotlivé elementy prezentovány navenek. K tomuto účelu je možné použít kaskádových stylů nebo lze XML dokument transformovat pomocí XSLT do jiné podoby. Jistou dobu se dokonce věřilo, že XML díky těmto vlastnostem nahradí HTML při tvorbě webových stránek a umožní tak také odstranění jeho sémantických nedostatků. K naplnění této představy však nedošlo, byla zřejmě příliš revoluční jak pro webové vývojáře, tak pro výrobce prohlížečů. Vznikl však jazyk XHTML postavený na XML, ale jinak spíše podobný HTML. Neobsahuje prezentační prvky, ale neumožňuje definovat libovolnou strukturu dokumentu.

V předcházející tabulce č.1 lze dobře spatřit rozdíly mezi XML a HTML. Zatímco v XML jsou specializované značky jako `<univerzita>` či `<fakulta>` mající předem definovaný význam, HTML takové značky nemá a nedokáže je vytvořit. Musíme tedy použít například obecných značek pro nadpisy. Ty sice umožňují předepsat určitý vzhled, ale to není žádná výhoda z hlediska automatického zpracovávání dat. Nevýhodou však je, že stejné značky mohou být použity pro označení naprosto různých věcí.

Definicí vlastních značek, tedy zavedením jakéhosi slovníku, a dodáním gramatiky ve formě DTD nebo XML Schema, lze vytvořit další jazyky. Takových již byla vytvořena i standardizována celá řada. Např. XHTML, SVG, RSS, XMPP a také RDF.

3.1.1 XML dokument a jeho syntaxe

Na začátku XML dokumentu se obvykle vyskytuje část nazývaná prolog. Ta obsahuje XML deklaraci, určující v jaké verzi XML je dokument vytvořen a jaké je jeho kódování. Může vypadat např. takto:

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

Rovněž lze v prologu odkázat na externí zdroj obsahující definici struktury dokumentu.

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

Detailnější informace o DTD si povíme dále.

Po úvodní části již následuje tělo dokumentu. Stejně jako HTML používá XML ke značkování tagů. Ty se do sebe mohou vnořovat a tím vytvářet stromovou strukturu. Nesmí se ale překrývat – pracuje se na „zásobníkovém“ principu, kdy poslední otevřený tag musí být prvním uzavřeným. XML předepisuje oproti HTML povinnost každý tag uzavírat, a to i prázdné elementy. Element se skládá z otevírající a uzavírající značky a obsahu mezi nimi. Např.:

```
<student>Michal Vrána</student>
```

Jména značek lze volit téměř libovolně. První znak musí být písmeno, podtržítko nebo dvojtečka. Žádný název nesmí začínat na `xml`. Obsahem elementu může být text nebo i další elementy. Existuje kořenový element nazývaný „root“, který obsahuje všechny ostatní elementy.

Elementy mohou obsahovat vlastnosti také ve formě atributů. Atribut je složen z názvu a hodnoty ohraničené uvozovkami či apostrofy. Hodnota může obsahovat pouze konečnou informaci, nelze vnořovat další elementy či atributy. V našem příkladu je ukázka atributu u názvu fakulty.

```
<fakulta nazev="FIT">...</fakulta>
```

Pokud je dokument syntakticky správně, označujeme jej jako „well-formed“. Jako validní se označuje well-formed dokument, který splňuje požadavky dané v DTD či XML Schema.

3.1.2 DTD

DTD (Document Type Definition) je jazyk, jehož úkolem je popis struktury XML nebo SGML dokumentu. Vymezuje množinu elementů a atributů použitelných v dokumentu a určuje jejich možné vzájemné uspořádání. Definice může být umístěna přímo v dokumentu, ale častější je použití externího souboru. DTD je již poměrně starý jazyk a existují modernější jazyky s větší vyjadřovací silou, ale i přesto se stále často využívá [5]. Následuje ukázka, jak by mohlo vypadat DTD pro náš vzorový příklad viz. tabulka č.1.

```
<!ELEMENT univerzita (nazev,fakulta+)>
<!ELEMENT nazev (#PCDATA)>
<!ELEMENT fakulta (student*)>
<!ELEMENT student (#PCDATA)>
<!ATTLIST fakulta nazev CDATA #REQUIRED>
```

Tento zápis předepisuje následující:

- V dokumentu je možné použít elementy typu `univerzita`, `nazev`, `fakulta`, `student`.
- Element `univerzita` musí obsahovat jeden element typu `nazev` a jeden nebo více elementů typu `fakulta`.
- Element `nazev` je typu PCDATA, což znamená Parsed character data. Tento typ označuje text, který bude dále zpracován parserem. Entity budou expandovány.
- Element `fakulta` obsahuje libovolné množství elementů student.
- Atribut `nazev` je povinný u elementu `fakulta` a je typu CDATA, což označuje text, který nebude dále zpracováván parserem.

DTD umožňuje definovat tzv. entity, což jsou zkratky, pomocí kterých je možné do dokumentu vkládat text či různé speciální značky. Například úhlové závorky používané pro uvozování tagů, by jinak nešlo do dokumentu vůbec zapsat (při zachování validity). Mezi ty nejznámější entity patří ty definované v HTML viz tabulka č.2.

Entita	**Znak**
`<`	<
`>`	>
`&`	&
` `	nezlomitelná mezera

Tabulka 2: Známé HTML entity

Entitu deklarujeme tímto způsobem:

```
<!ENTITY name "Michal Vrána">
```

Pak se na ni odkazujeme přes znak ampersand.

```
<student>&name;</student>
```

Existují i další možnosti jazyku DTD jak definovat vlastnosti dokumentu. Tento nástin nemá být kompletním popisem, ale pouze ukázkou.

3.1.3 XML Schema

XML Schema, někdy také XSD (XML Schema Definition), slouží ke stejnému účelu jako DTD – umožňuje definovat strukturu XML dokumentu. Oproti němu však nabízí výrazně bohatší možnosti. Syntaxe jazyka je postavená přímo na samotném XML. To zaprvé zlepšuje samotnou čitelnost pro člověka a zadruhé umožňuje využití stejných technologií. Např. zpracovávání jedním parserem, využívání stejných nástrojů pro tvorbu, atd. Důležitým posunem od DTD je možnost upravení již existujícího schématu pomocí přidání nebo omezení některé vlastnosti, což velmi usnadňuje znovupoužití již existujících schémat. Dále umožňuje použití datových typů, a to jak mnoha předdefinovaných, tak speciálně vytvořených. V DTD bylo možné specifikovat pouze řetězce [15].

Následující příklad demonstruje použití XML schématu pro náš „univerzitní“ příklad:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="univerzita">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nazev" type="xs:string" />
        <xs:element name="fakulta" minOccurs="1"
            maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="student" type="xs:string"
                 minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attribute name="nazev" type="xs:string"
                 use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Na začátku definice je zavedení jmenného prostoru `xs` pro elementy jazyka XML schema. Elementy se deklarují pomocí tagu `element`, kde atribut `name` definuje název a atribut `type` jakého je typu. Dále je možné deklarovat kardinalitu značky pomocí dvojice atributů `minOccurs` a `maxOccurs`. Pokud není ani jeden z těchto atributů zadán, musí se element vyskytovat v daném umístění pouze

jednou. Tímto způsobem je možné dosáhnout větší flexibility než v případě operátorů +, *, ? v DTD. Atributy značek se deklarují pomocí elementu `attribute`. U něj se také atributem `name` a `type` deklaruje název a datový typ a atributem `use` lze určit, zda je atribut v dokumentu povinný či volitelný.

Velkou výhodou XML schémat je možnost definice vlastních datových typů. Ale je zde i široký výběr typů zabudovaných:

- číselné typy – `integer`, `short`, `Byte`, `long`, `float`, `decimal`
- řetězcové typy – `string`, `ID`, `IDREF`, `CDATA`, `language`, `token`
- typy pro datum a čas – `time`, `date`, `datetime`, `duration`, `gMonth`

Pro definici složitých datových typů se používá sdružení několika jednoduchých typů a případně atributů do elementu `complexType`. Tento postup můžeme v příkladu spatřit u typu `univerzita` a `fakulta`. Pro určení, zda jsou všechny vnitřní elementy typu povinné popř. jiné druhy omezení, se využívá elementů `sequence`, `choice`, `all`. Dále je možné třeba již existujicí datový typ rozšířit pomocí elementu `extension` nebo naopak omezit pomocí `restriction`. Restrikcí lze například redefinovat povinnost výskytu atributu u elementu nebo změnit hodnoty min/maxOccurence. Aplikováním restrikce na zabudovaný typ lze také vytvořit vlastní jednoduchý datový typ.

```
<xs:simpleType name="shortString">
  <xs:restriction base="xs:string">
    <xs:maxLength value="5"></xs:maxLength>
  </xs:restriction>
</xs:simpleType>
```

Zbývá si říci, jak vlastně na vytvořené schéma odkázat ze samotného XML dokumentu. K tomu nám poslouží další ukázka:

```
<univerzita
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://url UniverzitaSchema.xsd">
...
</univerzita>
```

Stejně jako u DTD byla i tato podkapitola rychlým přehledem a ukázkou základních vlastností.

3.1.4 Jmenné prostory

Jak již bylo řečeno, XML je meta-jazykem, který umožňuje vytvářet další, z něj odvozené jazyky. Při tvorbě těchto jazyků je autorům ponechána úplná volnost, jaké značky a pro co budou používat. To je na jednu stranu velice flexibilní, ale na druhou stranu to může přinést i problémy. V rámci XML dokumentu je možné pracovat s více různými DTD nebo XSD a jelikož tyto definice mohly vzniknout na sobě nezávisle, mohou obsahovat kolizi – tedy, že obě definice definují tag se stejným názvem. To způsobí, že parser není schopen elementy rozlišit.

Řešením tohoto problému je zavedení jmenných prostorů neboli namespaces jako způsobu dosažení unikátního pojmenování všech elementů a atributů. Jmenný prostor se deklaruje pomocí atributu xmlns, jeho hodnotou je obvykle nějaký URI identifikátor. Pro rozlišení více prostorů je umožněno přidat ještě tzv. prefix. Pokud je prostor deklarován bez prefixu, nazývá se defaultní.

```
xmlns="http://www.w3c.org/1999/xhtml
xmlns:xhtml="http://www.w3c.org/1999/xhtml
```

3.1.5 Dotazování pomocí XPath

XML dokumenty mohou být velmi velké a obsahovat obrovský počet elementů. Abychom mohli s takovými dokumenty snáze pracovat a vyhledávat v nich, vznikly dotazovací jazyky jako je např. XQL, XML-QL, Xquery a zejména standardizovaný jazyk XPath.

XPath umožňuje vytváření výrazů, jejichž výsledkem bývá nejčastěji množina uzlů stromu, vytvořeného podle struktury XML dokumentu. Uzly v tomto stromě mohou být různého typu – kořenový uzel, uzly elementů, atributy, textové uzly, instrukce pro zpracování, komentáře nebo jmenné prostory. Navzájem se liší některými vlastnostmi. Xpath výraz však dokáže vrátit i hodnoty jiných typů, např. logická hodnota, číslo nebo textový řetězec [16].

Uzly stromu mohou být vybrány z různých míst. Toto místo určuje tzv. cesta. Cesta může být:

- absolutní – vycházející z kořenu dokumentu, začínají znakem /
- relativní – vycházející z aktuálního uzlu (ten, který je právě zpracováván)
- z libovolného, přesně určeného místa – pomocí několika funkcí jako je id() se lze v dokumentu přesunout na jiné místo a pak specifikovat požadovanou cestu

Cesta je vlastně sekvence složená z několika částí. Obecně ji lze zapsat jako $/t_1/.../t_n$. Každá část cesty může být navíc doplněna o identifikátor osy a predikát. Identifikátor osy říká, ve kterém směru od aktuálního uzlu se bude dál vyhledávat. Těchto identifikátorů je poměrně mnoho, mezi nimi např. `child::` procházející všechny děti aktuálního uzlu, `attribute::` pro osu s uzly atributů, `ancestor::` pro uzly, které jsou předky aktuálního uzlu a další. Dalším doplňkem cesty je predikát. Ten umožňuje vyfiltrovat uzly podle dalších kritérií. Predikáty se zapisují do hranatých závorek a vyhodnocují se jako logická hodnota. Pokud je predikát splněn, uzel zůstává ve výběru. Číselná hodnota v predikátu označuje pozici uzlu na aktuální ose.

Příklady:

- `/univerzita` – vybere kořenový uzel univerzita
- `/univerzita/fakulta` – vybere všechny uzly fakulta pod uzlem univerzita
- `//student` – vybere všechny uzly typu student kdekoliv v dokumentu. Dvojité lomítko umožňuje prohledávat celý dokument. Není tedy potřeba přesně znát cestu k uzlu

- `//fakulta/@nazev` – vybere atributy nazev ze všech uzlů typu fakulta v dokumentu. Symbol @ se používá pro označení atributu (osa `attributes::`)
- `//student[.="Michal Vrána"]` – vybere všechny uzly student, které obsahují text „Michal Vrána“. V hranatých závorkách je predikát, kterému musí uzly vyhovět
- `/univerzita/*` – vybere všechny uzly pod uzlem univerzita
- `//fakulta[@nazev="FIT"]` – vybere fakulty,které mají název FIT
- `fakulta[last()]/student[1]` – vyhledá uzly fakulta pod aktuálním uzlem a vybere z nich ten poslední, pod ním pak vybere první uzel typu student. `Last()` je jedna z mnoha předdefinovaných funkcí. Vrací pozici posledního uzlu. Odkázat lze také přímo na pozici uzlu – `student[1]`
- `ancestor::*[1]` – vybere prvního předka, tedy rodiče, aktuálního uzlu

V XPath výrazech můžeme používat také různé operátory a funkce. Sjednocení výsledků několika výrazů lze docílit s operátorem |. Následující výraz vybere první a poslední fakultu.

```
fakulta[first()] | fakulta[last()]
```

V predikátech lze používat logických operátorů `and` a `or`, relačních operátorů pro porovnávání nebo operátorů pro matematické operace jako sčítání, odčítání, násobení, dělení a zbytek po dělení. Následující ukázka vrátí každého druhého studenta nebo takového, který má ve jméně Michal.

```
//student[position() mod 2 = 1 or contains(.,"Michal")]
```

V předešlém příkladu je použito funkce `contains()` pro vyhledávání v řetězci. Existuje ale celá řada dalších funkcí, které lze použít pro různé účely – `string-length`, `concat`, `number`, `sum`, `round` a další.

Jazyk XPath je klíčový pro funkce jazyka XSL, který slouží především pro různé transformace XML. Více v následující podkapitole.

3.1.6 XSLT

XML samo o sobě nedefinuje žádné formátovací možnosti. Pokud si například v prohlížeči necháme zobrazit nějaký XML dokument, odpovědí je nám v lepším případě hezky přehledná struktura dokumentu. V situaci, kdy chceme dokument uživateli na webu prezentovat v nějaké přívětivější podobě, je nutné v XML dokumentu vytvořit tzv. instrukce pro zpracování. Ty mohou odkazovat na soubor s kaskádovými styly nebo XSL transformací. CSS styly se hodí zejména tam, kde je požadováno pouze definování vzhledu a XML má již vhodnou strukturu. To se ovšem moc často nestává. Dnešní stránky jsou velmi interaktivní a vizuálně složité. Je proto nutné použít druhý způsob a transformovat dokument do (X)HTML.

Soubor s XSL transformací je složen ze značek (příkazů) jazyka XSL (eXtensible Stylesheet Language) a značek formátovacího jazyka. Oba jazyky jsou aplikace nad XML. Aby šlo jejich elementy kombinovat, používají se jmenné prostory. Formátovací jazyk určuje, v jakém formátu bude výsledný dokument. Může jít o PDF, TeX, XML, nejčastěji používané (X)HTML popř. další. Na začátku souboru je kořenový element `xsl:stylesheet`, kde `xsl` je jmenný prostor http://www.w3.org/1999/XSL/Transform. Podle toho XSLT procesor rozpozná, že jde vůbec o šablonu a začne ji zpracovávat.

Základním prvkem jazyka XSL je element `template` obsahující šablonu. Tělo šablony je složeno z elementů formátovacího jazyka a dalších konstrukcí XSLT.

```
<xsl:template match="xpath výraz">
   tělo šablony
</xsl:template>
```

Mezi tyto konstrukce patří nejčastěji příkazy `value-of` a `apply-templates`. První jmenovaná zpracuje xpath výraz definovaný v jejím atributu `select` a výsledek vloží na výstup jako text. Příkaz `apply-templates` také způsobí zpracování výrazu v atributu `select`. Rozdílem však je, že na nalezené uzly jsou posléze vyhledány a aplikovány odpovídající šablony.

Zpracování celé transformace probíhá tak, že na začátku si XSLT procesor převede vstupní dokument do stromové reprezentace, aby mohl vyhodnocovat xpath výrazy. Strom je posléze procházen v původním pořadí, tj. do hloubky, a hledá se, zda některý z procházených uzlů neodpovídá výrazu z nějaké šablony. Pokud ano, šablona se aplikuje. Potomci nalezeného uzlu již nejsou automaticky dále zpracováváni. Pokud bychom je však zpracovat chtěli, můžeme to udělat pomocí příkazu `apply-templates`. Naopak při použití `value-of` se daná větev dále neprochází. Následující ukázka transformuje náš vzorový XML z tabulky č.1 do jeho „ekvivalentní“ HTML podoby.

```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/univerzita">
    <html>
      <body>
        <h1><xsl:value-of select="nazev" /></h1>
        <xsl:apply-templates select="fakulta"/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="fakulta">
    <h2><xsl:value-of select="@nazev"/></h2>
    Studenti:
    <ul><xsl:apply-templates select="student" /></ul>
  </xsl:template>
  <xsl:template match="student">
    <li><xsl:value-of select="."/></li>
  </xsl:template>
</xsl:stylesheet>
```

3.2 RDF

RDF (Resource Description Format) je formalismem pro grafovou reprezentaci dat. Ačkoliv je často označován jako jazyk, jde spíše o datový model, pomocí kterého lze modelovat libovolnou věc a její vlastnosti. Základním jazykem pro reprezentaci RDF je XML, ale existují i jiné syntaktické reprezentace, jako například N3 nebo N-Triples, které si také později popíšeme. Je důležité si uvědomit, že pomocí RDF pouze popisujeme nějaké věci, nedefinujeme jejich podstatu nebo význam. K tomu slouží ontologické jazyky jako RDF Schema a OWL.

Základními prvky datového modelu jsou:

1. Zdroje (resources) – Zdrojem může být v podstatě cokoliv, co jde identifikovat nějakým unikátním ukazatelem jako je např. URI. Obvykle jsou jimi lidé, místa a věci, o kterých chceme něco sdělit.
2. Vlastnosti (property) – Pomocí vlastností popisujeme nějaký vztah mezi zdroji, např. „je autorem“, „má věk“, „má barvu“, atd. Také vlastnosti mají svůj identifikátor a tedy jde vlastně o speciální typ zdroje.
3. Tvrzení (statement) – Jde o trojici ve tvaru (zdroj, vlastnost, objekt). Objektem může být opět zdroj či atomická hodnota v podobě literálu.

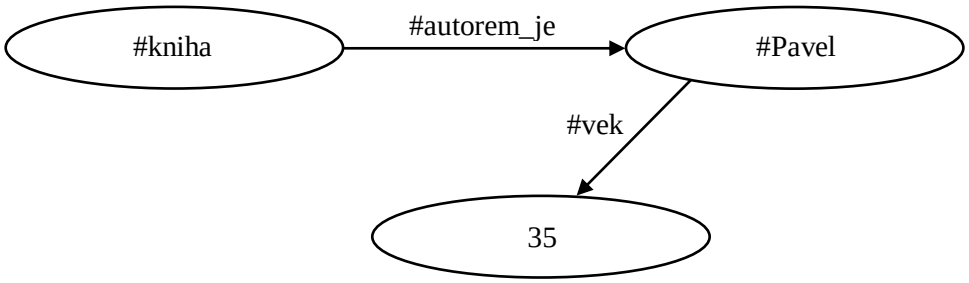
Příkladem takového tvrzení může být věta:

Autrem knihy je Pavel.

Do trojice by se dalo toto tvrzení zapsat následovně:

```
(http://mojeadresa.cz/#kniha,
 http://mojeadresa.cz/rdf#autorem_je,
 http://mojeadresa.cz/#Pavel)
```

Tuto trojici lze také zapsat jako logickou formuli *P(x,y)*, kde *P* je binární predikát mezi objektem *x* a objektem *y*. Tento predikát tvoří vlastnost, *x* je subjekt a *y* objekt. Proto se také někdy pro trojici používá označení subjekt-predikát-objekt.



RDF trojice můžeme reprezentovat také grafem. Jeho výhodou je, že může znázorňovat více různých tvrzení najednou a také je pro uživatele intuitivně pochopitelný. Hrany označují vlastnosti a uzly zdroje.

Reprezentací, která nás však nejvíce zajímá je zápis v XML. RDF dokument je XML dokument obsahující tag `rdf:RDF`. Obsahem tohoto elementu jsou elementy `rdf:Description`. Každý z nich popisuje nějaké tvrzení o zdrojích, které mohou být identifikovány třemi způsoby:

- Atributem `about` – odkazuje na existující zdroj.
- Atributem `ID` – vytváří nový zdroj.
- Bez udání jména – to má za výsledek vytvoření anonymního zdroje.

XML zápis vzorového tvrzení:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:mujns="http://www.mojeadresa.cz/rdf#">
  <rdf:Description rdf:about="http://www.mojeadresa.cz#kniha">
    <mujns:autorem_je rdf:resource="Pavel"/>
  </rdf:Description>
</rdf:RDF>
```

Element `rdf:Description` tvoří tvrzení o zdroji `http://www.mojeadresa.cz#kniha`. Vnořený do něj je tag (`mujns:autorem_je`) popisující vlastnost. Hodnotou vlastnosti je obsah tohoto tagu, popřípadě jeho atribut `rdf:resource`. Předveďme si složitější příklad obsahující více tvrzení.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:mujns="http://www.mojeadresa.cz/rdf#">

  <rdf:Description rdf:about="http://www.mojeadresa.cz#kniha">
    <mujns:autorem_je rdf:resource="Pavel"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.mojeadresa.cz#kniha2">
    <mujns:autorem_je>Jiri</mujns:autorem_je>
    <mujns:ma_nazev>Toto je nazev Knihy2</mujns:ma_nazev>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.mojeadresa.cz#kniha3">
    <mujns:autorem_je>
      <rdf:Description rdf:about="Pepa" mujns:prijmeni="Plch">
        <mujns:jmeno>Pepa</mujns:jmeno>
        <mujns:vek rdf:datatype="&xsd;integer">30</mujns:vek>
      </rdf:Description>
    </mujns:autorem_je>
  </rdf:Description>

</rdf:RDF>
```

Tento příklad obsahuje tři části. První je nám již známé jednoduché tvrzení, že Pavel je autorem knihy. Druhý příklad obsahuje tvrzení dvě – Jiří je autorem knihy2 a Kniha2 má název „Toto je název Knihy2“. Ze zápisu je vidět, jak můžeme snadno definovat dvě tvrzení o stejné věci. Dále také, na rozdíl od části první, nedefinuje autora knihy jako zdroj. O Jiřím, který je autorem knihy2 již tedy nelze dále nic tvrdit. Třetí část opět o knížce. Demonstruje, jak lze pomocí vnořování elementů

vytvořit několik závislých tvrzení a to: Pepa je autorem knihy3, Pepa se opravdu jmenuje „Pepa“, Pepa má příjmení „Plch“ a Pepovi je 30 let. Na tvrzeních není nic moc zajímavého, všimněme si ale rozdílu, jakým byla zapsána. Příjmení je definováno již jako atribut elementu `Description`. Takto lze totiž zkráceně zapsat alementy, které nemají žádné potomky. Jméno je zapsáno klasicky přes element, stejně jako věk. U toho je však navíc použito atributu `rdf:datatype`, který umožňuje specifikovat typ vlastnosti. Hodnotou atributu je opět nějaký obecný identifikátor jako URI.

V předcházejících příkladech je schopen člověk snadno rozpoznat, že Pepa, Jiří a Pavel jsou osoby. Ovšem pro stroj to již rozpoznatelné není, nikde to není formálně deklarované a ze jmen nebo vlastností to nerozezná. Existuje proto možnost, jak je to možné deklarovat. Představme si následující kód zasazený do předcházející ukázky.

```
<rdf:Description rdf:about="Pepa">
  <rdf:type
   rdf:resource="http://www.mojeadresa.cz/rdf#osoba"/>
</rdf:Description>

<mujns:osoba rdf:ID="Pavel">
  <mujns:jmeno>Pavel</mujns:jmeno>
</mujns:osoba>
```

Pro vyjádření vztahu, že určitý zdroj patří do nějaké větší obecnější množiny se v RDF používá element `rdf:type` s atributem `rdf:resource`. V ukázce jsme tedy formálně označili Pepu jako osobu. To stejné bylo provedeno pro Pavla, ovšem pomocí zkráceného zápisu – místo vnořeného elementu `rdf:type` se použije přímo element daného typu, v tomto případě `mujns:osoba`. Namespace samozřejmě musí odkazovat na stejné umístění jako `rdf:resource`.

Dále obsahuje RDF tzv. kontejnerové elementy. Ty se dají použít k označení skupiny zdrojů nebo atributů, o nichž jako o celku chceme něco tvrdit. Takovými elementy jsou `rdf:bag` pro neuspořádanou skupinu, `rdf:seq` pro uspořádanou skupinu a `rdf:alt` pro skupinu alternativ. Následující příklad tvrdí, že knihu4 napsal Pavel nebo Jiří.

```
<rdf:Description rdf:about="http://www.mojeadresa.cz#kniha4">
  <mujns:autorem_je>
    <rdf:Alt>
      <rdf:li rdf:resource="Pepa"/>
      <rdf:li rdf:resource="Jiri"/>
    </rdf:Alt>
  </mujns:autorem_je>
</rdf:Description>
```

3.2.1 Reifikace

RDF umožňuje pro modelování složitějších vztahů vytvářet tvrzení o tvrzení, tzv. reifikace. Je tedy možné vytvářet tvrzení jako:

Jiří věří, že Pavel je autorem knihy.

Reifikace se používají pro popsání domněnky nebo důvěry v jiné tvrzení. Lze toho dosáhnout tak, že se vytvoří nový zdroj reprezentující původní tvrzení a přiřadí se mu unikátní identifikátor. Tím ho lze použít jako zdroj v jiných tvrzeních a říkat o něm něco dalšího. Nově vytvořený zdroj je navázán na původní tvrzení přes vlastnosti `rdf:subject`, `rdf:predicate`, `rdf:object`. Ukázka vytvoření zdroje z původního tvrzení:

```
<rdf:Statement rdf:about="TvrzeniOAutorstvi">
  <rdf:subject rdf:resource="#kniha"/>
  <rdf:predicate rdf:resource="#autorem_je"/>
  <rdf:object rdf:resource="#Pavel"/>
</rdf:Statement>
```

3.2.2 Alternativní syntaxe N3, N-Triples

Existují i jiné syntaktické formáty pro zapisování RDF trojic než jen XML. Krátce si je představíme.

N3 (Notation 3)

Používá velmi stručný zápis a je dobře čitelná pro člověka. Trojice se zaznamenávají ve formátu:

```
<subjekt> <predikát> <objekt> .
```

Ukázka:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix mujns: <http://www.mojeadresa.cz/rdf#> .
@prefix adr: <http://www.mojeadresa.cz/#> .

adr:kniha mujns:autorem_je adr:Pavel ;
  mujns:nazev "Toto je nazev knihy" .
```

Na prvních dvou řádcích lze spatřit definici jmenných prostorů. Následuje definice subjektu následovaná dvojicemi predikát-objekt, ty se oddělují středníkem. Tečka uzavírá popis. Soubory s N3 notací mívají příponu .n3.

N-Triples

Používá stejný formát jako N3, ale nepoužívá jmenné prostory, všechny zdroje musí být plně identifikované. Také subjekt musí být uveden na každém řádku i v případě, že se k němu vztahuje více tvrzení. Nevýhodou tedy je oproti N3 delší zápis, výhodou pak snazší automatické zpracování a generování.

Ukázka:

```
http://www.mojeadresa.cz/#kniha
http://www.mojeadresa.cz/rdf#autorem_je
http://www.mojeadresa.cz/#Pavel .

http://www.mojeadresa.cz/#kniha
http://www.mojeadresa.cz/rdf#nazev
"Toto je nazev knihy" .
```

3.2.3 RDFa

RDFa (RDF in Attributes) je návrh organizace W3C na rozšíření jazyka XHTML (obecně ale i jakéhokoliv jiného XML dokumentu) o atributy pro vkládání metadat. Cílem RDFa je umožnit strojové zpracování a porozumění obsahu webových stránek. Problémem RDF na webu je to, že data se musí udržovat na dvou místech – jednou v HTML kódu a jednou v RDF kódu. Pomocí RDFa je možné mít data v jednom umístění dostupné jak pro prohlížeč, tak označkované pomocí RDF.

Podstatou je sada atributů, které mohou být použity pro RDF metadata. To jsou:

- `about` a `src` – URI specifikující RDF zdroj
- `rel` a `rev` – specifikující vztah a reverzní vztah k jinému zdroji
- `href` a `resource` – specifikace partnerského zdroje
- `property` – specifikuje vlastnost obsahu elementu
- `content` –volitelný atribut, který bude použit místo obsahu elementu v případě použití atributu `property`
- `datatype` – volitelný atribut pro specifikaci typu při použití atributu `property`
- `typeof` – volitelný atribut, který specifikuje RDF typ subjektu

Přidáním RDFa do stránky lze snadno označkovat obrázky, videa, informace o stránce, licenci a v podstatě cokoliv díky obecnosti RDF. Díky tomu už nebudou informace na stránce zpracovatelné jen člověkem, ale i strojem. Ukázka stránky s RDFa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    version="XHTML+RDFa 1.0" xml:lang="en">
  <head>
    <title>RDFa stranka</title>
    <meta property="dc:creator" content="Michal Vrána" />
  </head>
  <body>
    <p xmlns:dc="http://purl.org/dc/elements/1.1/"
       about="http://www.mojeadresa.cz#kniha">
      V knize
      <cite property="dc:title">Toto je nazev knihy</cite>,
      <span property="dc:creator">Pavel</span>
      Neni nic zajimaveho.
      Recenze publikována <span property="dcate"
          content="2010-10-04">v dubnu 2010</span>.
    </p>
  </body>
</html>
```

Problémem RDFa je, že jejich přidání na stránku způsobí, že není validní v rámci XHTML. Proto vznikla varianta s označením eRDF, ale ta se příliš neprosadila. Druhou možností jak zajistit validitu, je použít typu dokument XHTML+RDFa.

3.3 GRDDL

GRDDL (Gleaning Resource Descriptions from Dialects of Languages) je W3C doporučení pro získávání RDF metadat z XHTML popř. XML dokumentů pomocí jazyka XSLT. Cílem je umožnit získání RDF dat i tvůrcům stránek, pro které je tvorba RDF přímou cestou z nějakého důvodu neschůdná. Takový autor může označit zajímavé informace na stránce například pomocí mikroformátů a následně přidáním atributu `profile` do elementu `head` stránky označí dokument jako zpracovatelný pomocí GRDDL. Dále přidá element `link`, který obsahuje odkaz na XSL transformaci. GRDDL kompatibilní klient, který přistoupí na takovou stránku, si pak RDF může vygenerovat sám.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head profile="http://www.w3.org/2003/g/data-view">
<link rel="transformation"
 href="http://www.w3.org/2000/06/dc-extract/dc-extract.xsl" />
</head>
```

3.4 RDF Schema

RDF Schema (RDFS) bývá označováno za ontologický jazyk, ačkoliv jde spíše o sémantické rozšíření jazyka RDF. Ten, jak jsme si již řekli, slouží pouze k modelování vztahů mezi objekty. Neříká, z jaké domény tyto objekty jsou, ani jakou mají sémantiku. To lze vyjádřit pomocí ontologických konstrukcí a právě ty přidává RDFS.

3.4.1 Třídy

V RDF jsme popisovali konkrétní zdroje, jako byl Pavel, Pepa, kniha, atd. Tyto zdroje nazýváme individua. Pro ontologii ovšem musíme tato individua konceptualizovat, vytvořit z nich obecné třídy. Třída může být chápána jako množina elementů. Individua spadající do této třídy jsou jejími instancemi. Vzpomeňme na ukázku použití elementu `rdf:type` v kapitole o RDF. Jednalo se fakticky o vytvoření instance třídy `osoba`. Důležitou schopností RDFS je možnost omezit definiční obor a obor hodnot. Pak nelze zapsat nesmyslná tvrzení jako:

Autorem knihy je kniha2.

Kniha má věk 20.

3.4.2 Dědičnost

V RDFS je možné vytvářet ze tříd hierarchické struktury tím, že kromě tříd definujeme také podtřídy. K tomu se používá vlastnosti `rdfs:subClassOf`. Můžeme si uvést příklad. Opět použijeme třídu `osoba`. Dále si uvedeme třídy `žena`, `muž` a `mladážena`. První dvě jmenované jsou podtřídami třídy `osoba` a mladážena je podtřídou třídy `žena`. Vlastnost `subClassOf` je však transitivní a tak

je `mladážena` zároveň podtřídou třídy `osoba`. Z toho všeho vyplývá, že všechny instance těchto tříd jsou instancemi třídy `osoba`.

Zkrácená ukázka:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdfs:Class rdf:ID="osoba">
</rdfs:Class>

<rdfs:Class rdf:ID="muz">
  <rdfs:subClassOf rdf:resource="#osoba"/>
</rdfs:Class>
```

Základními třídami v RDFS jsou:

- `rdfs:Class` – třída všech tříd
- `rdfs:Resource` – třída pro všechny zdroje
- `rdfs:Literal` – třída pro literály(řetězce nebo čísla)
- `rdf:Datatype` – třída všech datových typů
- `rdf:Property` – třída všech vlastností
- `rdf:Statement` – třída všech reifikací

3.4.3 Vlastnosti

Vlastnosti jsou v RDFS definovány jako relace mezi zdrojem subjektu a zdrojem objektu. Popsat je můžeme pomocí třídy `rdf:Property`, popřípadě jiných vlastností jako `rdfs:range`. Vlastnosti jsou také třídy a můžeme je tedy také hierarchicky členit s pomocí `rdfs:subPropertyOf`. Opět je to relace transitivní a je možná vícenásobná dědičnost.

Základními vlastnostmi v RDFS jsou:

- `rdf:type` – vytváří relaci mezi zdrojem a třídou, tato relace vyznačuje instanci
- `rdfs:subClassOf` – definuje, že jedna třída je podtřídou jiné obecnější třídy
- `rdfs:subPropertyOf` – definuje, že vlastnost je speciálním případem jiné vlastnosti
- `rdfs:domain` – určuje defininiční obor = každý zdroj, kerému je dána taková vlastnost je instancí určené třídy
- `rdfs:range` – určuje obor hodnot = hodnoty budou instancemi určené třídy

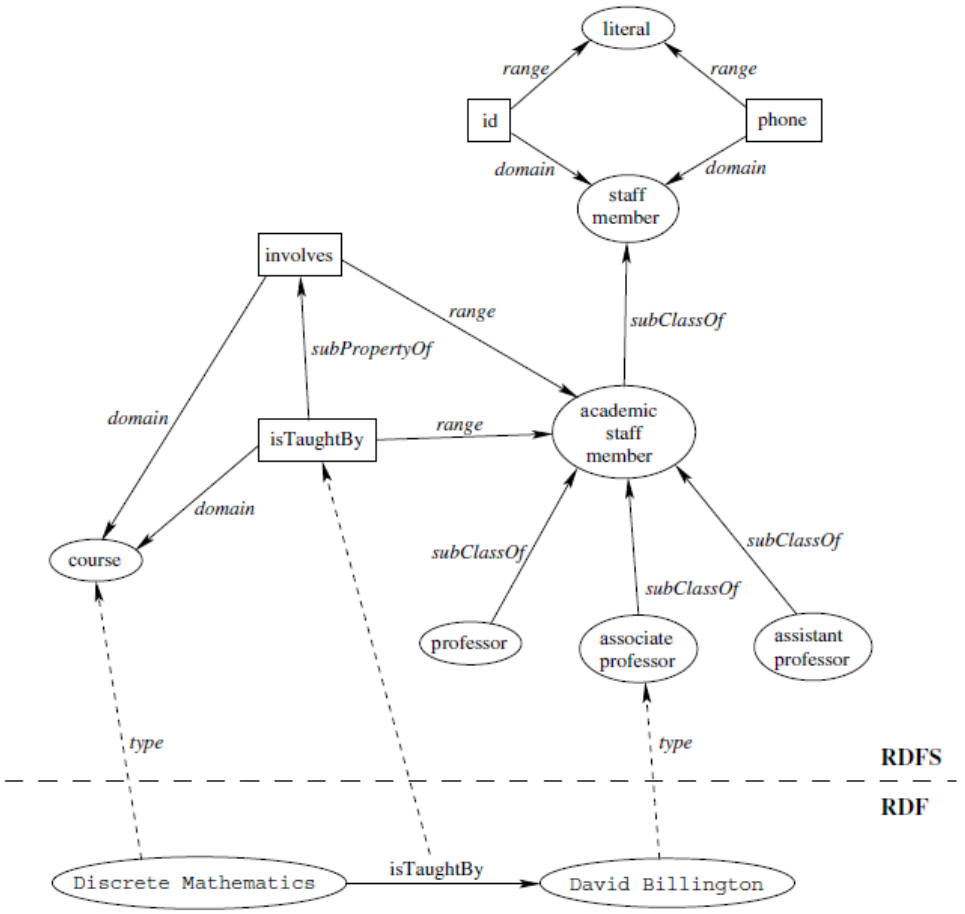
Pomocné vlastnosti:

- rdfs:seeAlso – odkaz na vysvětlující zdroj
- rdfs:isDefinedBy – odkazuje na zdroj s definicí v podobě RDF schématu
- rdfs:comment – vysvětlující komentář
- rdfs:label – snadno čitelný popisek

Ukázka na použití `rdfs:subPropertyOf`, `rdfs:range` a `rdfs:domain`:

```
<rdf:Property rdf:ID="umiRiditVozidlo">
  <rdfs:domain rdf:resource="#osoba"/>
  <rdfs:range rdf:resource="#vozidlo"/>
</rdf:Property>

<rdf:Property rdf:ID="umiRiditAutobus">
  <rdfs:subPropertyOf rdf:resource="#umiRiditVozidlo"/>
</rdf:Property>
```

Na následujícím diagramu lze spatřit „univerzitní“ ontologii a znázornění vztahu RDF a RDFS.



Obrázek 6: Univerzitní ontologie v RDFS [5]

3.5 OWL

Dalším ontologickým jazykem je OWL (Web Ontology Language). Za jeho vznikem stojí opět konsorcium W3C, které si uvědomovalo potřebu mocnějšího ontologického jazyka než je RDFS. Vyšli z jazyka DAML-OIL, což je jazyk vytvořený spojením dvou starších ontologických jazyků DAML a OIL[10]. OWL byl standardizován a je doporučován jako hlavní ontologický jazyk sémantického webu [17].

RDFS dovoluje modelovat pouze některé ontologické znalosti – hierarchie tříd, hierarchie vlastností, omezení relací (domain, range), instance tříd. Neposkytuje však dostatečné vyjadřovací možnosti pro pokročilé modelování ontologií. Neumožňuje:

- přesnější určení možného obsahu domain a range nějaké vlastnosti – např. nemůžeme omezit hodnoty vlastnosti „živí se“ pro masožravce pouze na „maso“, pokud máme u třídy živočichů větší rozsah možností
- označení tříd jako disjunktní – např. třídy muž a žena
- definovat třídu jako sjednocení jiných tříd – např. třídu osoba ze tříd muž a žena
- definovat kardinalitu – např. říci, že osoba má pouze jednu matku a jednoho otce
- deklaraci speciálních charakteristik jako tranzitivita, unikátnost, inverze

Jazyk OWL toto vše umožňuje, ale je potřeba dodat, že s vyšší expresivitou jazyka roste i složitost usuzování a tedy i výpočetní nároky. Zároveň zdaleka ne všichni potřebují a využijí celé spektrum možností jazyka. Vznikly proto tři podoby jazyka.

3.5.1 Verze jazyka OWL

OWL Full

OWL Full podporuje veškeré konstrukce jazyka OWL a tím umožňuje dosáhnout maximální expresivitu. Dovoluje kombinovat prvky OWL s prvky RDF/RDF Schema a je s nimi plně kompatibilní. A to jak syntakticky, tak sémanticky – libovolný RDF dokument je také platným OWL Full dokumentem a jakýkoliv RDFS úsudek je platným úsudkem v OWL Full. Netrvá na striktním oddělování tříd, vlastností, instancí a datových typů a umožňuje měnit význam některých primitiv. Jeho problémem je, že je tak mocný až je téměř nerozhodnutelný – je problémem v něm úplně a efektivně usuzovat [5].

OWL DL

OWL DL je podmnožinou jazyka OWL Full, která omezuje, jak mohou být konstrukce jazyka OWL a RDF použity.Vyžaduje, aby byly třídy, vlastnosti, instance a datové typy navzájem odděleny. Tím jazyk odpovídá deskriptivní logice a jeho výhodou se stává schopnost efektivně usuzovat. Naopak ale

[10] DAML byl návrhem americké agentury DARPA a OIL vznikl z iniciativy EU

ztrácí plnou kompatibilitu – RDF dokument již obecně není platným OWL DL dokumentem, ale OWL DL dokument platným RDF dokumentem je.

OWL Lite

OWL Lite je ještě omezenější verze jakyka OWL. Zakazuje i použití konstrukcí jako výčet, disjunkce tříd, kardinalitu. Je vhodná pro základní použití jako je tvorba hierarchických struktur a je také snazší ji implementovat do různých nástrojů. Platí, že každá OWL Lite ontologie je platnou OWL DL ontologií.

3.5.2 Konstrukce jazyka OWL

OWL ontologie staví nad RDF modelem a pro ten je doporučenou syntaxí XML. Ale lze použít i jiné, viz. kapitola 3.2.2. OWL (RDF) dokument s ontologií je složen z několika částí.

Hlavička

Kořenovým elementem dokumentu je pochopitelně `rdf:RDF`. Včetně deklarace jmenných prostorů vypadá úvod dokumentu obvykle nějak takto:

```
<rdf:RDF
xmlns:owl ="http://www.w3.org/2002/07/owl#"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

Následuje samotná hlavička ontologie – element `owl:Ontology`. Ten obsahuje některé pomocné informace o ontologii jako komentáře, verze ontologie, ale může obsahovat napojení na jiné ontologie pomocí jejich vložení.

```
<owl:Ontology rdf:about="">
  <rdfs:label>Nazev ontologie</rdfs:label>
  <rdfs:comment>Ukazka ontologie</rdfs:comment>
  <owl:priorVersion
    rdf:resource="http://www.mojeadresa.cz/ont_v1"/>
  <owl:versionInfo>v_2</owl:versionInfo>
  <owl:imports
   rdf:resource="http://www.mojeadresa.cz/osoba"/>
</owl:Ontology>
```

Element `owl:Ontology` má prázdný element `rdf:about`, což vyznačuje, že popisovaným subjektem je samotný dokument s ontologií. Účelem `rdfs:label` a `rdfs:comment` je popis snadno čitelný pro člověka, `owl:priorVersion` odkazuje na předchozí verzi ontologie, což může být užitečné pro nějaké automatizované systémy, `owl:versionInfo` pak označuje (v blíže neupřesněném formátu) aktuální verzi ontologie.

Pomocí `owl:import` pak lze importovat další ontologii bez nutnosti zavádět další jmenný prostor. Vlastnost `owl:import` je tranzitivní a tedy případná jiná ontologie, která bude importovat tuto, bude také importovat její importované ontologie.

Třídy

Třídy tvoří hierarchickou strukturu ontologie. V OWL se definuji pomocí elementu `owl:Class` a to jedním z následujících způsobů:

- Identifikátorem třídy – má pouze jméno, nemá žádné prvky ani vlastnosti.

```
<owl:Class rdf:ID="kniha">
```

- Výčtem prvků – třídu tvoří konkrétní instance.

```
<owl:Class rdf:ID="cernobile">
  <owl:oneOf rdf:parseType="Collection">
   <owl:Thing rdf:about="#cerna"/>
   <owl:Thing rdf:about="#bila"/>
  </owl:oneOf>
</owl:Class>
```

- Omezením vlastností – třídu tvoří instance splňující omezení.

```
<owl:Class rdf:ID="#motorka">
  <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource="#madvekola"/>
     <owl:allValuesFrom rdf:resource="#vozidla"/>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- Sjednocením nebo průnikem tříd – je určena množinovou operací nad dalšími třídami.

```
<owl:Class rdf:ID="osoba">
 <owl:unionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#muz"/>
  <owl:Class rdf:about="#zena"/>
 </owl:unionOf>
</owl:Class>
```

- Doplňkem – třídu tvoří všechny individua nespadající do určité třídy.

```
<owl:Class rdf:ID="nelide">
  <owl:complementOf>
    <owl:Class rdf:about="#osoba"/>
  </owl:complementOf>
</owl:Class>
```

Výše zmíněné body ukazují způsoby, jak vytvořit nové ontologické třídy. OWL však obsahuje také několik předdefinovaných tříd. Velmi důležitou z nich je `owl:Thing`, která zastřešuje všechny třídy. Další je `owl:Nothing`, která je prázdnou třídou. Všechny třídy jsou tedy podtřídami `owl:Thing` a nadtřídami `owl:Nothing`.

Podtřídy se definují pomocí vlastnosti `rdfs:subClassOf`, která říká, že všechny prvky dané třídy tvoří podmnožinu prvků třídy nadřazené. Dále je možné pomocí `owl:equivalentClass` označit dvě třídy jako ekvivalentní, tedy definovat že mají stejné prvky. Naopak označit, že třídy nemají žádné společné prvky lze pomocí `owl:disjointWith`.

Dále se blíže podíváme na možnost definovat třídu omezením vlastností. Při takové definici se využívá anonymní třídy obsahující pouze prvky splňující nějakou stanovenou podmínku. OWL umožňuje dva typy omezení – na hodnotu a na kardinalitu. Následující ukázka definuje třídu `dílo` jako vše, co má jako autora minimálně jednu osobu.

```
<owl:Class rdf:about="#dilo">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#autorem_je"/>
      <owl:allValuesFrom rdf:resource="#osoba "/>
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">
        1
      </owl:minCardinality>
     </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

V ukázce je použito konstrukcí s elementy `owl:allValuesFrom` a `owl:minCardinality`. První jmenovaná omezuje všechny hodnoty vlastnosti `autorem_je` na instance třídy osoba. Z hlediska logiky se vlastně jedná o univerzální kvantifikátor. Další možností by bylo použití `owl:someValuesFrom`. To vyžaduje, aby alespoň jedna z hodnot vlastností splňovala dané omezení, tedy jde o existenční kvantifikátor. Dále je zde možnost omezit vlastnost na konkrétní hodnotu pomocí `owl:hasValue`.

Co se kardinality týče jsou zde tři možné způsoby omezení – `owl:minCardinality`, `owl:maxCardinality`, `owl:Cardinality`. První vymezuje třídu všech individuí, kde má určená vlastnost minimálně daný počet hodnot. Druhý způsob se liší od prvního tím, že omezuje maximální počet hodnot. Třetím způsobem je pak určení přesného počtu.

Vlastnosti

V OWL existují dva typy relací:

- Objektové – spojují dva objekty.
  Používá se element `owl:ObjectProperty`.
- Dato-typové – spojují objekt s datovým typem.
  Používá se element `owl:DataTypeProperty`.

Pro základní charakteristiky – hierarchii vlastností, definici definičního oboru a oboru hodnot relace se používá kostrukcí z RDF Schema – `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range`. OWL poskytuje další možnosti pro specifikaci vlastností. Jednou z nich je `owl:inverseOf`. Ta umožňuje definovat relaci inverzní k nějaké jiné. Např. inverzní relací k „vlastnit“ by byla relace „je_vlastněno“. Další možností je konstrukce `owl:equivalentProperty`, která se používá pro definici, že dvě relace mají stejné prvky (relace však i přesto nemusí mít stejný význam).

Dalšími elementy jazyka OWL jsou konstrukce pro globální omezení kardinality jako je `owl:FunctionalProperty`. Tou se definuje vlastnost, která má maximálně jednu hodnotu pro každou instanci. Příkladem je například osoba a její jméno, váha, věk. Ty může mít pouze jednou. Pokud bychom chtěli, aby hodnota byla unikátní pro každé dvě různé instance, použijeme `owl:InverseFunctionalProperty`. Např. rodné číslo osoby je unikátní.

```
<owl:ObjectProperty rdf:ID="maJmeno" />
<owl:FunctionalProperty rdf:about="maJmeno"/>
<owl:ObjectProperty rdf:ID="maRodneCislo" />
<owl:InverseFunctionalProperty rdf:about="maRodneCislo"/>
```

Dále jsou zde elementy `owl:SymmetricProperty` a `owl:TransitiveProperty`. První označuje symetrickou relaci jako je např. „je sourozenec“, druhý pak tranzitivní relaci jako „je větší než“ nebo „je předkem“.

3.6 Dotazovací jazyky

Je logické, že informace zapisované pomocí RDF trojic, chceme také někdy získat zpátky. Proto také pro Sémantický web existují dotazovací jazyky. Existuje jich celá řada – SPARQL, SeRQL, XsRQL, ARQ, Xcerpt a mnohé další. My se však budeme podrobněji zabývat pouze jazykem SPARQL, jakožto tím nejrozšířenějším a oficiálně doporučovaným konsorciem W3C.

Dotazovací jazyky se navzájem značně liší a to jak v syntaxi, tak ve svých možnostech. Z hlediska primárního účelu rozlišujeme několik kategorií [18]:

- Ryze dotazovací jazyky – dotaz specifikuje kritéria vyhledávání a jeho výsledkem je pouze podmnožina vstupních dat.
- Dotazovací jazyky s možností výpočtu nových hodnot – dokáží s použitím aritmetiky a agregačních funkcí popsat vytvoření nové hodnoty a její vložení do výsledku.
- Dotazovací jazyky s možností výpočtu nových trojic – dokáží nejen vytvářet nové hodnoty, ale i nové trojice.

Dále jazyky rozlišuje například různá podpora slovníků, odvozování, přístup k výběru dat z RDF databáze nebo jejich syntaktický původ. Některé se odvozují od jazyka SQL, jiné od dotazovacích jazyků pro XML, funkcionálního či logického paradigmatu nebo se jedná o grafové dotazovací jazyky.

3.6.1 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) je deklarativní jazyk pro dotazování v RDF a také dotazovací protokol, kterým lze přeposílat dotazy na jiné zdroje. Ačkoliv zdaleka nejde o jazyk, který poskytuje nejvíce možností [18], prosadil se jako standard konsorcia W3C. Mnozí

doufají, že jedna norma dopomůže k prosazení Sémantického webu stejně, jako se to kdysi povedlo u normy ANSI SQL a relačních databází.

SPARQL umožňuje psát dotazy, které jsou složené z trojic a jejich konjunkcí nebo disjunkcí. Pracuje na principu prohledávání grafu a hledání vzorů odpovídajících dotazu, které jsou následně vráceny jako výsledek. Základní a nejjednodušší formou vyhledávaného vzoru je právě trojice. V té ale na rozdíl od RDF trojice (subjekt-predikát-objekt) lze používat zastupující proměnné. V dotazu se pak obvykle kombinuje několik takových trojic, které dohromady tvoří hledaný vzor.

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX mujns:<http://www.mojeadresa.cz/rdf#>
SELECT ?kniha ?nazev
WHERE {
  ?kniha rdf:type mujns:Kniha .
  ?kniha
     mujns:autorem_je http://www.mojeadresa.cz/#Pavel ;
     mujns:nazev ?nazev .
}
ORDER BY ?kniha
```

Ukázkový dotaz má za cíl vyhledat všechny knihy, které napsal Pavel a jejich název. V klauzuli where jsou specifikovány tři trojice. První vyhledává všechny instance, které jsou typu `kniha` (tedy mají predikát `rdf:type` s hodnotou `kniha`). Druhý vzor najde všechny knihy, které mají vlastnost `autorem_je` s hodnotou `Pavel`. Třetí pak ve zkráceném zápisu vyhledává všechny knihy, které mají nějaký název a tento „vkládá“ do proměnné `?nazev`. Všechny trojice jsou uzavřené ve složených závorkách, které slouží k seskupování vzorů. Vzory uvnitř jsou implicitně v konjunkci. Otazník označuje proměnnou, za ním následuje její název. Pomocí klíčového slova `PREFIX` je možné deklarovat vlastní jmenné prostory a následně je použít v trojicích, čímž se celý zápis stává kratší a pro člověka přehlednější.

SPARQL sám o sobě nenařizuje podporovat sémantiku danou pomocí RDFS nebo OWL. Záleží na dotazovaném systému, zda to bere v potaz. Rozdíl může být zásadní. Například následující situace – dotaz vyhledává všechny osoby a ty jsou v RDF reprezentovány pomocí podtříd žena a muž. Pokud systém podporuje sémantiku, tak výsledek bude obsahovat také instance podtříd žena a muž. V opačném případě však dotaz nenalezne nic – ve výsledku jsou zahrnuty pouze vzory vyhovující v dotazu explicitně stanovené třídě osoba.

Struktura dotazu je velmi podobná jako u jazyku SQL, obsahuje klauzule select, from, where, order by.

- SELECT – jde o projekci. Určuje co bude ve výsledku a v jakém pořadí.
- FROM – nepovinné určení dotazovaného zdroje. Těchto zdrojů může být i více. Pokud není zdroj dat specifikován, je za něj považována báze znalostí dotazovaného systému.

- WHERE – slouží ke specifikaci grafových vzorů, které jsou následně vyhledávány. Pro zápis trojic se používá syntaxe Turtle podobná N3.
- ORDER BY – je modifikátor uspořádání výsledků. Lze použít známé termíny ASC pro vzestupné řazení a DESC pro sestupné.

Důležitým je klíčové slovo FILTER, které se vkládá do klauzule WHERE a je možné pomocí něj provádět restrikci nalezených vzorů. Lze použít standardní aritmetické a relační operátory a pro porovnávání řetězců také zabudovanou podporu regulárních výrazů (nahrazuje LIKE z SQL).

```
SELECT ?os
WHERE
{
  ?os rdf:type mujns:Osoba ;
  ?os mujns:vek ?vek .
  FILTER (?vek >= 18).
}
```

Dosud uvedené příklady vyžadují, aby instance měly všechny požadované vlastnosti. V tak heterogenním prostředí jako tvoří RDF databáze se však snadno stane, že různé instance mohou mít různé vlastnosti. Proto existuje také možnost, jak nějaký vyhledávaný vzor označit za volitelný. Používá se k tomu klíčové slovo OPTIONAL. Například předcházející příklad by tak, jak je uveden, vrátil pouze osoby, u kterých je znám věk a ten je větší nebo roven 18 let. Kdybychom chtěli mít ve výsledku i osoby, u nichž věk neznáme, museli bychom upravit příklad následovně.

```
SELECT ?os
WHERE
{
  ?os rdf:type mujns:Osoba ;
  OPTIONAL { ?os mujns:vek ?vek . FILTER (?vek >= 18) }
}
```

Posledními příkazy, které si uvedeme, budou UNION, DISTINCT, OFFSET a LIMIT. UNION slouží ke spojení výsledků nalezených podle různých vzorů. Hodí se například tam, kde pro jednu reálnou vlastnost existuje více ontologických vlastností (může se stát při kombinaci více ontologií). Příkladem může být získání jména osoby buď z ontologie FOAF nebo vCard. Zbylé tři příkazy jsou modifikátory známé z SQL. DISTINCT způsobí, že každý řádek výsledku bude mít unikátní kombinaci hodnot, OFFSET udává index prvního vráceného výsledku a LIMIT omezuje celkový počet výsledků.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT DISTINCT ?name
WHERE
{
   { ?person foaf:name ?name } UNION { ?person vCard:FN ?name }
}
OFFSET 10
LIMIT 10
```

3.7 Mikroformáty

Technicky vzato, mikroformáty nepatří pod oficiální technologie Sémantického webu. Nepracují s RDF ani s formálními ontologiemi, avšak jejich myšlenka a cíl je stejný – podporovat sémantiku na webu. Od doby svého vzniku v roce 2006 si získaly poměrně značnou oblibu a jde zřejmě o nejčastěji používanou technologii pro sémantické značkování HTML kódu. Proto se jimi budeme zabývat i v rámci této práce.

Mikroformáty by se daly popsat jako jednoduchá pravidla o tom, jak vkládat sémantické informace do HTML. Nejedná se o žádnou samostatnou a komplexní technologii jako například RDF, CSS nebo (X)HTML. Vznikly jako reakce na stejný neutěšený stav, který dal vzniknout myšlence o Sémantickém webu. Tedy, že zajímavých informací je na webu hodně, ale je problém je najít, identifikovat a automaticky zpracovat. Na rozdíl od technologií jako RDF či OWL řeší mikroformáty problém přidání sémantiky k datům využitím stávajících a běžně používaných technologií. Na tomto přístupu je vidět jisté prozření – s časovým odstupem od vzniku konkurenčních technologií Sémantického webu již bylo jasné, že jejich adopce nejde tak hladce, jak mnozí doufali. Za hlavní důvod byla považována jejich přílišná složitost a nevyzrálost. Byla zde tedy poptávka po něčem jednodušším a tento prostor zaplnily právě mikroformáty.

Mikroformáty jsou otevřeným projektem, na který dohlíží komunita lidí okolo microformats.org. Jedním z autorů původní myšlenky byl Tantek Çelik, současný CTO Technorati, který mimo jiné pracoval u společností Apple a Microsoft. K projektu se později přidaly další významné osobnosti světového Internetu a podporu vyjádřily důležité firmy IT byznysu – mimo oněch dvou zmíněných výše, také Yahoo, Google, Mozilla, aj. Díky tomu se mikroformáty rozšířily a mají solidní podporu v rámci vyhledávačů, prohlížečů a existuje také celá řada různých doplňků.

3.7.1 Definice a principy

Oficiální definice [19]: Navrženy v prvé řadě pro lidi a teprve pak pro stroje. Mikroformáty jsou množinou jednoduchých otevřených formátů dat postavených nad široce používanými standardy. Namísto zahození všeho co dnes funguje, se mikroformáty snaží nejdříve řešit jednodušší problémy tím, že se adaptují současným zvyklostem a způsobům použití. (např. XHTML, blogování).
Základními principy mikroformátů jsou [19]:

- řešit konkrétní problém
- řešit ho co nejjednodušeji
- navrhovat nejprve pro lidi, poté pro stroje
- využít stavebních bloků široce používaných standardů
- modularita/zabudovatelnost
- umožnit a podpořit decentralizovaný vývoj, obsah a služby

3.7.2 Jak vypadají

Jak již bylo zmíněno, mikroformáty se vkládají přímo do HTML kódu. Většinou se k tomu využívá atributu `class`, někdy také `id`, `title`, `rel` nebo `rev`. Do těchto atributů se vloží hodnota, která má předem dohodnutý jednoznačný význam. Příklad:

```
<span class="vcard">
  <span class="fn">Michal Vrána</span>
</span>
```

Tento minimalistický příklad pracuje s mikroformátem `vCard`, který má reprezentovat jakousi elektronickou vizitku. V rámci `vCard` je definováno několik dalších CSS tříd s definovanou sémantikou. Jednou z nich je `fn` (formatted name), která je určena pro jméno. V příkladu je pro označení textu, resp. jména, použito dvou řádkových elementů `span`, nicméně šlo by použít i jiných elementů , jako `address` nebo `p`. Navenek se takové označkování vůbec nemusí poznat, ale pokud CSS třídám nějaké formátování dodáme, lze snadno zvýraznit třeba všechna jména na stránce.

3.7.3 Mikroformáty v odkazech

Jde o mikroformáty založené na elementu odkazu (`<a>`) a použití jeho dvou atributů, `rel` a `rev`. Jednoduchý mikroformát je vytvořen přiřazením určité hodnoty těmto atributům. HTML volně předepisuje množinu těchto hodnot, ale umožňuje si přidat další. Právě toho mikroformáty využívají.
Atribut `rel` popisuje vztah cílové stránky (deklarované v atributu href) ke zdrojové odkazující stránce. Zřejmě nejznámějším a nejčastějším je použití tohoto atributu při linkování souboru s kaskádovými styly [20].

```
<link rel="stylesheet" type="text/css" href="style.css" />
```

Atribut `rev` je jednoduše reverzně otočený vztah k `rel`.
Do této kategorie řadíme obvykle mikroformáty pojmenované `rel-license`, `rel-tag` a také `rel-home`. Posledně jmenovaný se příliš nepoužívá. Slouží k vytvoření odkazu, který vede na domovskou stránku webu. Usnadňuje se tím navigace a zlepšuje popis struktury webu.

rel-license

Dle mého názoru velmi užitečný mikroformát sloužící k vyznačení informace o licenci, pod kterou je právě prohlížený obsah distribuován. Cílem odkazu může být rovnou konkrétní licenční ujednání. Díky zanoření do jiných mikroformátů jako např. `hReview`, může být pro různé prvky na stránce specifikovaná jiná licence. Tím mikroformáty podporují decentralizaci služeb. Vyhledávače Google i Yahoo umožňují vyhledat dokumenty podle licence a zohledňují i rel-license.

```
<a href="http://creativecommons.org/licenses/by-nc-nd/3.0/"
rel="license">copyright</a>
```

rel-tag

Jeden z nejčastěji používaných mikroformátů je `rel-tag`. Umožňuje „tagovat“ z libovolného místa, kam ho lze vložit a tím podporuje decentralizaci. Přidáním atributu s hodnotou tag k odkazu říkáme, že odkazovaný dokument je zařazen do tzv. „tagspace“ podle textu odkazu. Specifikace říká, že koncová část URL adresy se musí shodovat s tímto tagspace. Výsledný kód může vypadat třeba takto:

```
<a href="http://technorati.com/tag/tech" rel="tag">tech</a>
```

3.7.4 Mikroformát XFN pro popis vztahů

XFN (XHTML Friends Network) je jednoduchý způsob, jak reprezentovat lidské vztahy pomocí webových odkazů.

Využívá se atributu `rel` u elementu `<a>`. Hodnota nebo hodnoty, které jsou tomuto atributu dány, vyjadřují vzájemné vztahy strany odkazující a odkazované. XFN vzniklo a rozšířilo se díky sociálním sítím a komunitám (především blogerům), kde byla potřeba po lidštějším způsobu vyjádření vztahu k ostatním, např. kdo koho zná, kdo čte jaké blogy atd. [21]

Ukázky XFN mikroformátu:

```
<a href="http://twitter.com/mivra" rel="me">Michal Vrana</a>
<a href="http://www.linkedin.com/in/petrpalas" rel="co-worker">
  Petr Palas</a>
```

XFN definuje malou množinu hodnot, které se přiřazují do atributu `rel`. Ty můžeme rozdělit do několika kategorií:

- Přátelé a známí – hodnoty `friend`, `acquaintance`, `contact`.
- Fyzické – hodnota `met`.
- Profesní – hodnoty `colleague`, `co-worker`.
- Zeměpisné – hodnoty `co-resident`, `neighbor`.
- Rodinné – hodnoty `child`, `parent`, `sibling`, `spouse`, `kin`.
- Citové – hodnoty `muse`, `crush`, `date`, `sweetheart`.
- Osobní identifikace – hodnota `me`.

V praxi se nejčastěji používají hodnoty `friend` a `me`. První slouží k označení stránky přítele a druhá pro označení jiné mé stránky. Při použití `me` je logické, že nelze použít žádnou jinou hodnotu. Ostatní hodnoty mají význam dle svého překladu z angličtiny, není potřeba je více rozebírat. Dovolím si zde poznamenat, že některé z těchto možností mi připadají naprosto nesmyslné. Nedokážu si představit, že by se nějak hromadně označovaly romantické nebo rodinné vztahy.

3.7.5 Mikroformát hCard

Na Internetu existuje obrovské množství webů, které „někde“ na svých stránkách obsahují kontaktní informaci. Často by bylo užitečné pracovat s touto informací automaticky, např. vyhledat si adresu na mapě. A právě mikroformáty se výborně hodí k anotování takových informací, konkrétně mikroformát `hCard`. Ten je určen k reprezentaci lidí i organizací – nejčastěji právě ve smyslu adresy/kontaktních informací. Může obsahovat položky jako např. jméno, poštovní adresa, emailová adresa, telefonní číslo nebo fotografie, ale povinným údajem je pouze jméno.

Dle obecných principů se při jeho návrhu poohlédlo jinam a zjistilo se, že zde existuje standard vCard (RFC 2426), který definuje atributy adresy, jejichž názvy lze použít v HTML jako definici třídy kaskádového stylu.

Zápis adresy ve formátu vCard muže vypadat následovně [22]:

```
ADR;TYPE=dom,home,postal,parcel:;;123
Main Street;Any Town;CA;91921-1234;
```

Stejná adresa, ale jako součást mikroformátu `hCard` pak takto:

```
<div class="adr">
  <abbr class="type" title="dom">U.S.</abbr>
  <span class="type">home</span> address, for
  <abbr class="type" title="postal">mail</abbr> and
  <abbr class="type" title="parcel">shipments</abbr>:
  <div class="street-address">123 Main Street</div>
  <span class="locality">Any Town</span>,
  <span class="region">CA</span>
  <span class="postal-code">91921-1234</span>
</div>
```

Tento kód je ve skutečnosti mikroformátem `adr`, který slouží samostatně pro označení adresy a bývá jednou ze součástí `hCard`. Ten ale musí obsahovat ještě další informace, především již zmíněné jméno. Proto si ukázku upravíme, aby byla kompletní.

```
<div class="vcard">
  <p class="fn n">
    <span class="given-name">Michal</span>
    <span class="family-name">Vrána</span>
  </p>
  <span class="email">mivra@mojeadresa.cz</span>
  <div class="adr">...
  </div>
</div>
```

Přidali jsme tedy element s definovanými třídami `fn` a `n`, které slouží pro označení formátovaného jména a také další element s třídou `email`. Kombinací takových elementů a vnořených mikroformátů tedy vzniká `hCard`. Zajímavostí je, že samotné HTML obsahuje element `<address>`. Bohužel tento je definicí určen jako kontaktní adresa na autora stránek nebo vlastníka dokumentu. Navíc nemá předepsanou žádnou strukturu, která by rozlišila jednotlivé části adresy. Proto se není čemu divit, že je používán v naprosto minimální míře.

3.7.6 Mikroformát hCalendar

Stejně jako `hCard` vychází ze standardu `vCard`, vychází `hCalendar` z `iCalendar`. To je standard pro výměnu kalendářních událostí. Mikroformát hCalendar má stejné vlastnosti. Např. jméno, popis, místo, datum nebo URL. Ve složitějších případech také třeba přesný čas zahájení nebo ukončení. V kódu jsou vyjádřeny hodnotami v atributu `class`. Tou hlavní je `vEvent`, další povinné jsou `summary` (krátký popis) a `dtstart` (začátek události). Dalšími, již volitelnými může být konec události, popis, doba trvání, místo, místo uvedené mikroformátem `adr` nebo celým `hCard`. Velmi jednoduchý příklad mikroformátu `hCalendar`:

```
<div class="vevent">
  <span class="summary">Událost</span>
  <abbr class="dtstart" title="2010-09-13T09:00:00">
    13 září
  </abbr>
  <abbr class="dtend" title="2010-09-14T18:00:00">
    14 září
  </abbr>
  <div class="description">Popis událost</div>
</div>
```

3.7.7 Ostatní mikroformáty

Existuje ještě mnoho dalších mikroformátů na než tu už nezbyl prostor. Veškeré informace o nich lze nalézt na adrese `microformats.org`. Zde jsem se soustředil na ty, které už mají uzavřené specifikace. Mezi ty, u nichž tomu tak není, ale jsou poměrně dost podporované, patří hReview pro recenze, hRecipe na recepty, hProduct pro produkty nebo hResume pro životopisné informace.

Výhodou mikroformátů obecně proti RDFa může být jejich jednoduchost a přeci jen větší rozšíření. Na druhou stranu jsou však velkou nevýhodou nejednoznačné identifikované objekty (nemají URL), jejich omenezý počet a tedy i vyjadřovací možnost a také to může být centralitovaný vývoj.

3.8 Microdata

Microdata jsou dalším způsobem, jak anotovat obsah stránky sémantickou informací. Přímo tedy konkurují mikroformátům a RDFa. Návrh vznikl na půdě W3C ve vývojové skupině pro specifikaci HTML5, jíž by měly být microdata součástí. Microdata se inspirují u konkurentů a snaží se v sobě spojit jejich silné stránky – jednoduchost mikroformátů a expresivitu RDFa.

Mikrodata model tvoříme na stránce pomocí atributů `itemscope` a `itemprop`, které lze definovat libovolnému elementu. `Itemscope` označí prvek, ke kterému se vztahuje množina vlastností. Vlastnost je uvozena pomocí `itemprop` a tvoří ji dvojice jméno-hodnota. Itemscope elementy lze

v kódu zanořovat. Důležitým je dále atribut `itemtype`, který slouží k obecnému určení třídy prvku. Atribut `itemid` zase umí prvku dodat globální identifikátor [23].

Uveďme příklad:

```
<div itemtype="http://microformats.org/profile/hcard"
     itemid="http://www.mojeadresa.cz#vizitka"
     itemscope>
  <div itemscope itemprop="n">
    <div>First Name:
      <span itemprop="given-name">Michal</span>
    </div>
    <div>Last Name:
      <span itemprop="family-name">Vrána</span>
    </div>
  </div>
  <div itemscope itemprop="adr">
    <div>City:
      <span itemprop="locality">Brno</span>
    </div>
  </div>
</div>
```

V ukázce lze vidět, že typ této položky je hCard. Microdata dokáží tímto způsobem použít slovníky definované mikroformáty. Dále je přidána identifikace pomocí `itemid` a po ní již samotný atribut `itemscope`. Uvnitř hlavního elementu jsou další dvě zanořené položky – tím se respektuje stejná struktura jako u mikroformátů. Vlastnosti top-level položky jsou dvě – `n` a `adr`. Až ty posléze obsahují hodnotové vlastnosti pro jméno, příjmení a lokalitu.

4 Využití sémantických technologií

Čas Sémantického webu teprve nastane, ale již nějakou dobu s ním první průkopníci experimentují a snaží se přijít na to, jak nejlépe přetavit jeho potenciál a technologie v nové služby. Vývoj probíhá především ve všech oblastech webu, ale i v desktopových aplikacích a mobilních službách se objevují první reálné produkty. V této práci se však zaměříme na web.

4.1 Vyhledávače

Jak již bylo řečeno v první části této práce, jednou z hlavních motivací pro Sémantický web je změnit nebo alespoň vylepšit způsob, jakým vyhledáváme informace. Google, Bing a ostatní vyhledávače se snaží najít způsob, jakým dostat data bohatá na informace k uživatelům. Doposud nejviditelnějším projevem je obohacování nalezených výsledků o grafické prvky a různé dodatečné informace. Takto upravené výsledky se nazývají „rich snippets“. Jsou uplatněny zatím jen v úzce určených oblastech jako osoby, produkty, recenze, apod. nicméně tento okruh se postupně rozšiřuje a snad jednou nastane doba, kdy budou indexovány a poskytnuty všechny sémanticky anotované informace na webu.

4.1.1 Google

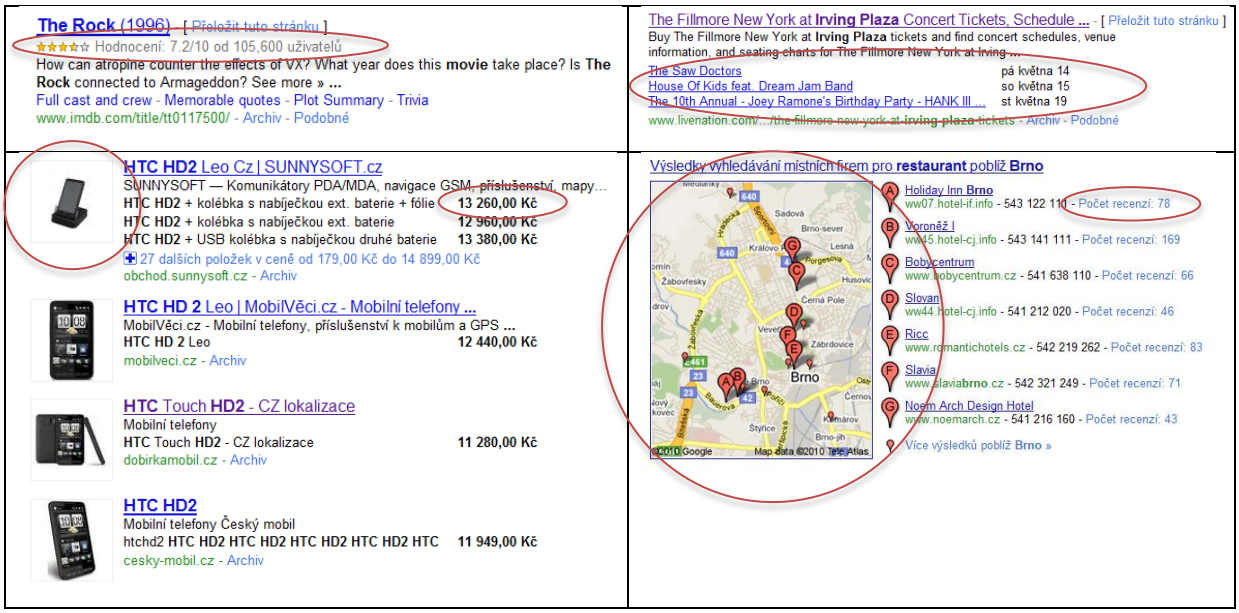
Google přišel s termínem „rich snippet“ v době, kdy začal indexovat mikroformáty, později pak přidal RDFa a velmi rychle adoptoval také microdata[11]. Podporuje tedy všechny tři konkurenční technologie pro sémantické značkování a je možné s nimi anotovat recenze, profily osob, kontaktní informace, produkty nebo různé události. Jak může vypadat jednoduchý snippet pro osobu lze vidět na následujícím obrázku.

Obrázek 7: Ukázka rich-snippet pro hCard mikroformát

Po zadání jména osoby do vyhledávání byl nalezen profil na síti LinkedIn. Ta obsahuje profesní životopisná data, která jsou sémanticky označená a veřejně dostupná. Google tedy může již v rámci výsledků rovnou poskytnout informaci, že „Petr Palas je CEO společnosti Kentico Software“. To může být obzvláště užitečné, pokud bylo nalezeno více osob se stejným jménem. Obecně tedy snippety napomáhají uživatelům v rozhodnutí, zda je pro ně stránka relevantní. Jsou výhodné také pro autory stránek, protože zvyšují pravděpodobnost prokliku o desítky procent.

[11] Hlavním editorem HTML5 je Ian Hickson, aktuálně zaměstnanec Google

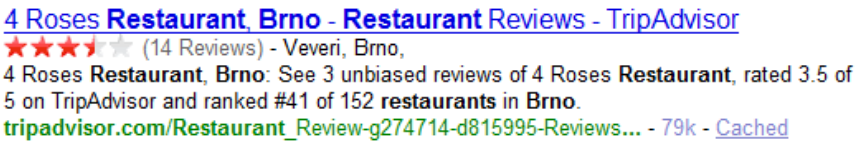
V následující tabulce je ukázka několika dalších snippetů z vyhledávače Google. První zobrazuje informace o hodnocení filmu „The Rock“ nalezené ve filmové databázi na IMDB.com. Už ze stránek vyhledávače jsme schopni odhadnout, zda se má cenu na film vůbec dívat. Na dalším příkladu je ukázáno, jak mohou vypadat výsledky obohacené o data z mikroformátu hCalendar – pro specifické místo je zobrazeno, co je na programu dnes, zítra, pozítří. Dále je zde přehledný souhrn informací o nějakém produktu. Zobrazuje prodejce, obrázek a cenu, což je velice užitečné pro nakupování. A jako poslední je předveden výsledek vyhledávání pro frázi „restaurant brno“.Několik prvních výsledků je zaneseno do mapy, je uvedeno telefonní číslo a také počet recenzí a odkaz na ně. Ne všechny informace z těchto příkladů musejí být nezbytně získané na základě sémantických technologií, ale je jasné, že právě ty poskytují prostředky, jak takové informace získávat obecně a v podstatě pro cokoliv.



Tabulka 3: Ukázky Google rich-snippets

4.1.2 Yahoo SearchMonkey

Yahoo byl zřejmě prvním velkým vyhledávačem, který se chopil myšlenky Sémantického webu. Vytvořil rozšíření nazvané SearchMonkey, které získává metadata umístěná na stránkách ve formě mikroformátů nebo RDF/RDFa/eRDF. Budoucnost tohoto vyhledávače je v současnosti poměrně nejistá vzhledem k obchodním vztahům Yahoo a Microsoftu v souvislosti s jeho vyhledávačem Bing. SearchMonkey dokáže zpracovat informace o produktech, recenzích, osobách, událostech, životopisech, herních titulech a dalších.



Obrázek 8: Ukázka vyhledávacího rozšíření pro Yahoo

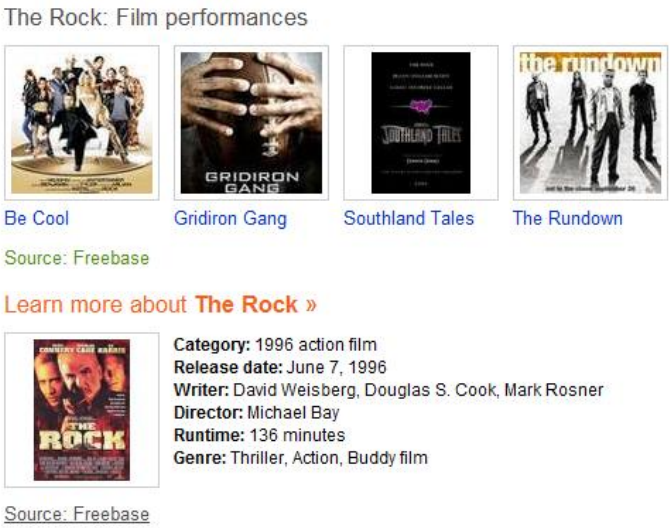
Velmi zajímavá je iniciativa Yahoo nazvaná YQL (Yahooo Query Language) a Open Data Tables. Jedná se o jazyk velmi podobný SQL, kterým je možné dotazovat různé zdroje strukturovaných dat na Internetu. Ty jsou obvykle zpřístupněny pomocí aplikačních rozhraní založených na REST, XML-RPC nebo SOAP. YQL k těmto službám umožňuje vytvořit unifikovanou přístupovou vrstvu. Open Data Table je způsob, který umožňuje přidat vlastní rozhraní mezi zdroje dotazovatelné pomocí YQL.

```
SELECT * FROM flickr.photos.search WHERE text="Brno"
```

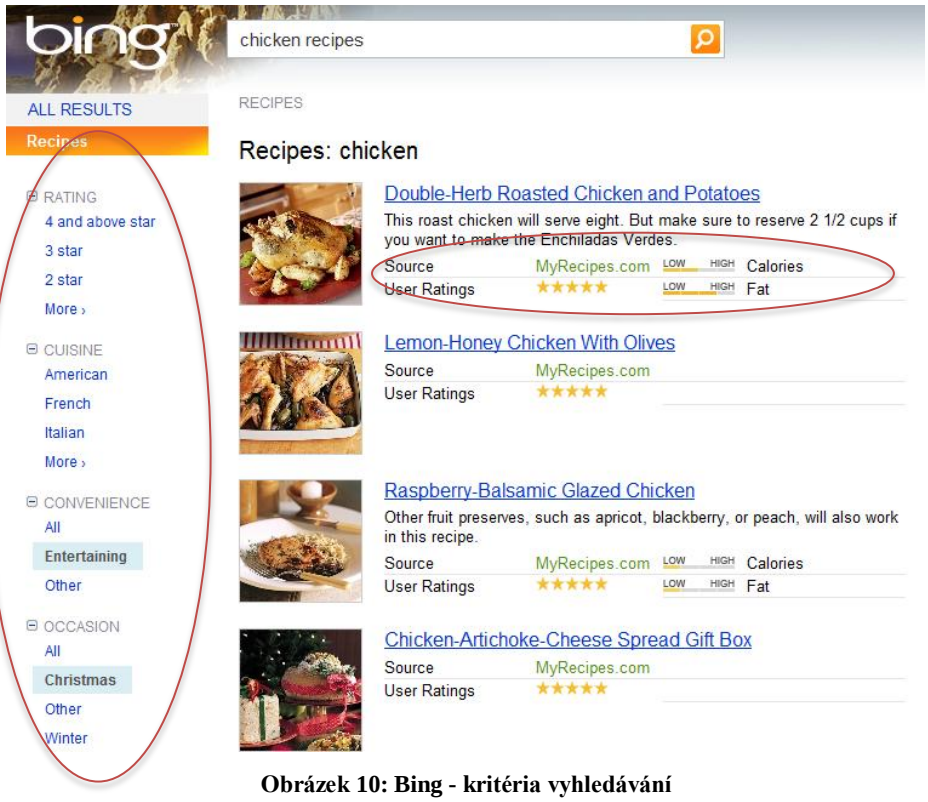
Jak ukazuje předcházející ukázka je YQL opravdu velmi přímočaré. Dokonce umožňuje i výrazy typu INSERT a UPDATE.

4.1.3 Bing

Trochu jinak na to jde vyhledávač Bing od Microsoftu. Snaží se rozpoznat, co je vlastně vyhledáváno a podle toho do výsledku zahrnuje kontextové informace, viz obrázek č. 9. V některých oblastech dokonce umožňuje omezovat vyhledávání podle specializovaných kritérií. Uveďme si následující příklad. Při zadání slova „chicken“ do vyhledávacího pole je kontextově nabídnut odkaz „chicken recipes“. Po kliknutí na něj se v levém sloupci objeví celá řada možností pro filtrování výsledků. Ty umožňují zpřesnit data například podle hodnocení receptu, stylu kuchyně, příležitosti nebo náročnosti přípravy. Filtrování je nabídnuto také při vyhledávání produktů, kde lze vyhledávat podle ceny, značky. Přímo ve vyhledávači lze zobrazit specifikace, obrázky či hodnocení podle hledisek jako výkon nebo snadnost použití. Je jasné, že takto obohacené vyhledávání získává jiný rozměr a jinak spíše marketingové označení „decision engine“ zde nabývá docela reálných základů.



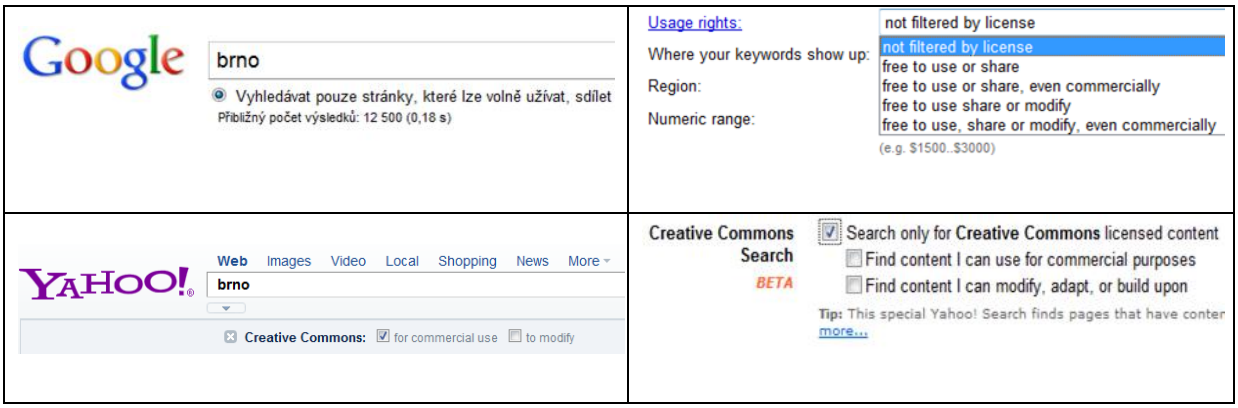
Obrázek 9: Kontextové informace ve výsledcích Bing



Obrázek 10: Bing - kritéria vyhledávání

4.1.4 Vyhledávání podle licence

Google a Yahoo využili značného rozšíření mikroformátu `rel-license` a jiných způsobů, jak označit licenci určitého díla a upravili rozhraní svých vyhledávačů, aby umožňovali specificky hledat dokumenty a obrázky, které se dají volně šířit, upravovat nebo využít pro komerční použití. V tomto směru se zdaleka nejvíce využívá licencí iniciativy Creative Commons. Zatímco u Googlu to podle odzkoušení funguje dobře, Yahoo nacházelo i díla pod jinými licencemi.



Tabulka 4: Google a Yahoo vyhledávání podle licence

4.1.5 Ostatní

Vedle velkých vyhledávačů vzniklo také několik menších, sémanticky orientovaných služeb. Mezi ty známější patří Hakia[12], která organizuje výsledky do záložek jako důvěryhodné zdroje, zprávy nebo obrázky. Dokáže nabídnout také rozšiřující a doplňující dotazy. Mezi dalšími službami jmenujme SenseBot[13], Cognition[14], DeepDyve[15] nebo specializovaný vyhledávač na ontologie Swoogle[16]. Většina těchto vyhledávačů je značně experimentální, ale některým se podařilo dotáhnout svou myšlenku do použitelného výsledku. Jednou z takových služeb je PowerSet[17] odkoupený v roce 2008 Microsoftem. Ten dnes používá jeho služeb k doplnění výsledků ve vyhledávači Bing, ale služba funguje odděleně dál. PowerSet se specializuje na menší, relativně strukturované korpusy jako je WikiPedia nebo FreeBase, kombinuje informace z více zdrojů a hlavně je dokáže přehledně prezentovat [24].

4.2 Sémantické publikování

Hlavním aspektem Webu 2.0 je způsob, jakým vzniká obsah webových stránek. Tvoří ho sami uživatelé na svých blozích, wiki stránkách, v sociálních sítích atd. Obecně tedy publikují nějaké informace. Tyto informace jsou však většinou nestrukturované a provázání na různé další zdroje pomocí odkazů či přidání obrázků je časově náročné. A to nejen pro obyčejného uživatele. Vytvoření bohatého obsahu je náročné i pro tradiční velké informační portály. Proto investují značné prostředky do různých řešení, které mohou pomoci tento problém řešit. A v této oblasti vyniká právě Sémantický web. Na jeho technologiích a formátech jsou postavena rozšíření pro blogovací a publikační systémy jako Zemanta, OpenCalais nebo sémantické wiki systémy. Sémantická data přímo z webu mohou být vynikajícím zdrojem například pro infografiku.

„Sémantické publikování je o publikování dokumentů a informací jako datových objektů za použití formátů Sémantického webu. Je zaměřeno na počítače a jejich schopnost porozumět struktuře a významu informací a tím umožnit jejich efektivnější vyhledávání a integraci.“ [10]

Obecně jsou dva přístupy k sémantickému publikování:

- Publikovat informace a data přímo zapsaná v sémantických formátech a jazycích.
- Použít sémantické technologie k anotaci již existujících dokumentů a databází.

[12] www.hakia.com
[13] www.sensebot.com
[14] www.cognition.com
[15] www.deepdyve.com
[16] www.swoogle.umbc.edu
[17] www.powerset.com

4.2.1 DBpedia

DBpedia je komunitní projekt s cílem extrahovat všechny informace z Wikipedie do strukturované podoby formátu RDF. V současné době je již v databázi Dbpedie více než 100 mil. RDF trojic a je jedním z největších a nejčastěji odkazovaných zdojů Sémantického webu. Svá data poskytuje veřejnosti pod licencí GNU a to přes SPARQL, Linked Data a další rozhraní.

Samotná extrakce informací z Wikipedia probíhá především z částí, které jsou alespoň částečně strukturované, tj. informační boxy, obrázky, kategorie a tabulky. Samotný text příspěvků většinou není moc strukturovaný, proto je extrakce složitější [25].

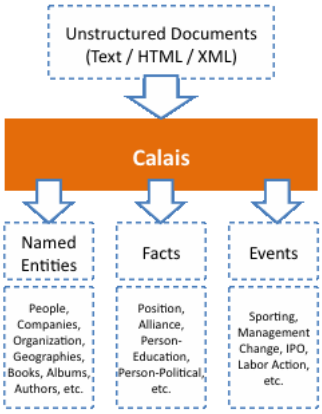
4.2.2 Zemanta

Zemanta je služba zaměřená především na bloggery, která jim pomáhá doplnit k jejich příspěvku relevantní obrázky, tagy nebo odkazy a tím vytvořit bohatší a lákavější obsah. Pro rozeznání relevantních doplňujících informací používá rozeznávání přirozeného jazyka a sémantických algoritmů. Nalezne slova z příspěvku, u kterých má ve své databázi doplňující obsah, a tento zobrazí po boku příspěvku. Blogger pomocí něj doplní svůj článek nebo text může nechat upravit zcela automaticky. Existují jak rozšíření pro prohlížeče (Firefox i IE), tak pro různé CMS/blogovací systémy. Na následujícím obrázku je vidět, jak může vypadat takový obohacený příspěvek. Vpravo je panel, ze kterého lze vybrat obrázek a související články, které mají být k článku přidány. V samotném textu jsou pak vybraná slova prolinkována a je přidán obrázek. Pod textem je pak seznam souvisejících dokumentů a výběr značkovacích tagů.

Obrázek 11: Text rozšířený pomocí Zemanta

4.2.3 OpenCalais

OpenCalais je velmi podobná služba jako Zemanta, za kterou stojí tisková agentura Thomson Reuters. Dokáže z textu identifikovat velké množství různých entit (osoby, místa, společnosti, ...), faktů a událostí. Poté vytvoří propojení na data z Wikipedie, DBPedie a dalších otevřených zdrojů dat, a vytvoří sémantická metadata obohacující text. Poskytuje rozsáhlé aplikační rozhraní, rozšíření pro prohlížeče a CMS systémy a také napojení na LinkedData, o kterých si řekneme více později.



Obrázek 12: Zpracování textu pomocí OpenCalais

4.3 Sociální sítě

Poměrně bouřlivým vývojem si v posledních letech prošly sociální sítě. Patří dnes mezi nejnavštěvovanější stránky na světě. Hlavním problémem je zde pro uživatele uzavřenost některých sítí (bohužel také dominantního Facebooku). Nemohou sociálně interagovat s uživateli, kteří mají profil založený jinde, nevlastní svá data, nemohou se snadno přesunout jinam. Tento stav se nazývá „oddělená sociální sila“. Bude obtížné ho změnit, protože uzavřenost je pro sítě výhodná z obchodního hlediska. V budoucnu by mohl situaci napravit právě Sémantický web. Ten z podstaty podporuje decentralizaci služeb a jejich vzájemnou interoperabilitu. Již dnes existuje ontologie FOAF, která umožňuje popsat osoby a jejich vzájemné vztahy, nebo obecné rozhraní OpenSocial navržené společností Google pro přístup k datům v sociálních sítích.

Další potíž, kterou sociální sítě řeší, je usnadnit uživateli orientaci v nepřeberném množství obsahu, rozpozat co uživatele zajímá a takový obsah mu nabídnout.

4.3.1 FOAF

FOAF (Friend of a friend) je strojově čitelnou ontologií popisující osoby (popř. dokumenty), jejich aktivity a jejich vztahy k ostatním lidem a věcem [26].

Myšlenka stojící za FOAF je velmi podobná jako u XFN – popsat vazby mezi objekty na webu. Jde na to ovšem obecněji než mikroformát XFN – využívá W3C standardů sémantického webu RDF a OWL. Pomocí FOAF jsem schopen sémanticky vyjádřit, že moje jméno je Michal, příjmení Vrána,

že bydlím v Brně, jaké jsou moje záliby, kdo jsou moji přátelé atd. Tyto informace pak lze umístit na web, kde je naleznou roboti vyhledávačů a mohou je nějak dál zpracovat. Autoři FOAF vytyčili několik cílů, které by chtěli s FOAF dosáhnout [27]:

- Získat lepší kontrolu a přehled nad fragmenty dat tvořících současný web.
- Umožnit pokročilé vyhledávání díky vazbám, které FOAF definuje.
- Podpořit decentralizaci služeb.

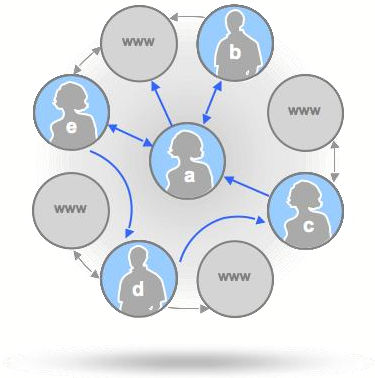
Ukázka jak může vypadat FOAF RDF dokument:

```
<rdf:RDF xmlns:foaf="http://xmlns.com/foaf/0.1/" >
  <foaf:Person rdf:nodeID="me">
    <foaf:name>Michal Vrána</foaf:name>
    <foaf:title>Bc.</foaf:title>
    <foaf:givenname>Michal</foaf:givenname>
    <foaf:family_name>Vrána</foaf:family_name>
    <foaf:mbox>mailto:mivra@atlas.cz</foaf:mbox>
    <foaf:img>
      <foaf:Image rdf:about="http://myimage.url"/>
    </foaf:img>
    <!-- IM účty -->
    <foaf:jabberChatID>mivra@jabber.cz</foaf:msnChatID>-
    <!-- Kontakty -->
    <foaf:knows>
      <foaf:Person>
        <foaf:name>Adam Dostál</foaf:name>
      </foaf:Person>
    </foaf:knows>
    <!-- Zájmy -->
    <foaf:interest>
      <foaf:Document rdf:about=" http://www.w3.org/2001/sw/">
        <dc:title>Semantic Web</dc:title>
      </foaf:Document>
    </foaf:interest>
  </foaf:Person>
</rdf:RDF>
```

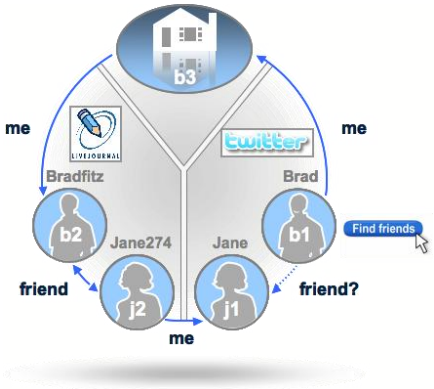
4.3.2 Social graph API

Formáty XFN a FOAF umožňují definovat vazby mezi osobami. Ty jsou reprezentovány unikátními URI adresami. Google se rozhodl využít tyto sémantické informace a upravil své prohledávací roboty tak, aby tato data sbírali. Vazby mezi stránkami(osobami) jsou ukládány do orientovaného grafu a tím je v podstatě vytvářena globální decentralizovaná sociální síť.

Social graph API je rozhraní, které umožňuje jednoduše přistupovat k této síti a dotazovat se jí na užitečné informace o kontaktech uživatelů. Např. kdo všechno jsou přátelé nějakého uživatele nebo kde všude (na jakých portálech) má uživatel vytvořen svůj účet.



Obrázek 13: Socialní graf [28]



Obrázek 14: Nalezení přítele [28]

Jsou vyhledávány dva typy hran – spojení deklarující náležitost k určité osobě (`rel-me`) a spojení s ostatními lidmi. Na obrázku č. 14 lze vidět, jakým způsobem může být rozhraní Social Graph API užitečné k nalezení kontaktů. Modelová situace: v síti (Twitter), ve které se uživatel (Brad) právě zaregistroval, nemá žádné kontaky. Ale proto, že uvedl odkaz na jinou svou stránku (LiveJournal) byla přes API zjištěna vazba na jiného uživatele (Jane). API je dotázáno na další stránky tohoto uživatele a je nalezena stránka na původní síti. Uživatel (Brad) je o tom informován a hned si může přidat známého přítele do svých kontaktů na nové síti.

4.3.3 Twine

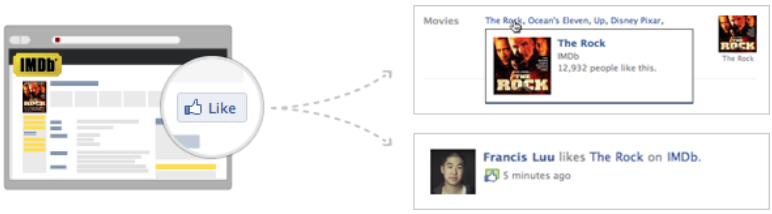
Twine je sociální síť, kde uživatelé sbírají a dávají dohromady různé druhy informací a ty následně sdílí s ostatními. Každý si může navolit, co ho zajímá a systém ho pak na takové věci upozorňuje. To by nebylo nic až tak neobvyklého, ale Twine toho dosahuje s použitím minimálního úsilí na straně uživatele. Tomu totiž stačí pro nějakou stránku vytvořit záložku nebo pouze nahrát dokument a Twine již za pomoci rozeznávání přirozeného jazyka a technologií Sémantického webu zařídí otagování zdroje a anotaci dat v něm. Dále síť poskytuje aplikační rozhraní a může sloužit jako platforma pro další aplikace.

4.3.4 Facebook Open Graph protocol

V dubnu 2010 se přidala k podpoře Sémantického webu i sociální síť Facebook. Nutno však dodat, že tato podpora je značně omezená a také kontroverzní[18]. Facebook uvolnil specifikaci Open Graph protokolu, který umožní, aby se anotované stránky staly součástí sociálního grafu v jeho síti. Taková stránka následně získává stejné vlastnosti a možnosti jako má facebook page – zobrazuje se v profilech vyhledávání, atd. Bohužel slovník určující co může stránka reprezentovat je omezený.

[18] http://www.readwriteweb.com/archives/does_facebook_really_want_a_semantic_web.php

Aktuálně je to zhruba 30 pojmů z kategorií jako jsou věci, lidé, místa, organizace, produkty, filmy, sport nebo webové stránky. Stránka se stává součástí grafu pomocí tlačítka „Like“.



Obrázek 15: Open Graph protocol [29]

Důležitým bodem pro pro Sémantický web je, že pro anotaci stránek bylo zvoleno RDFa. Na následující ukázce je příklad anotace filmového portálu IMDB pro film „The Rock“. Je na ní vidět přidání jmenného prostoru pro slovník Open Graph protokolu a jak je stránka označena vlastnostmi u `meta` tagů.

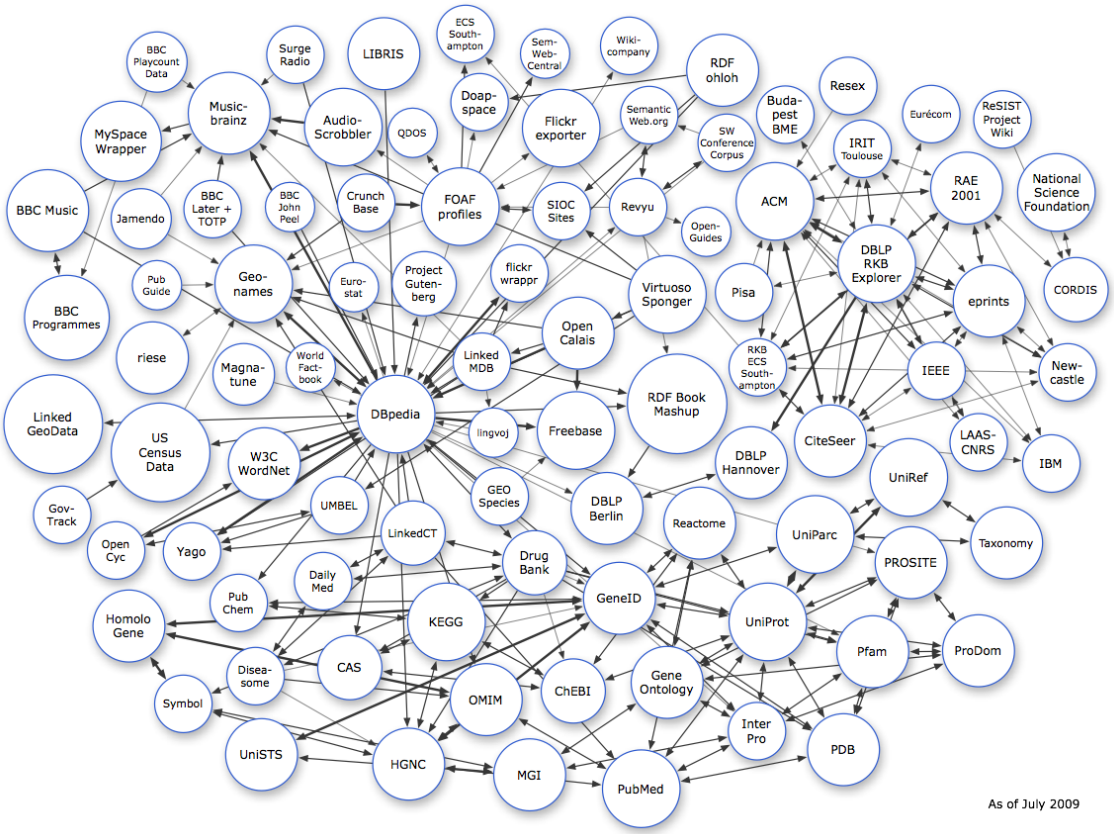
```
<html xmlns:og="http://opengraphprotocol.org/schema/">
<head>
<title>The Rock (1996)</title>
<meta property="og:title" content="The Rock" />
<meta property="og:type" content="movie" />
<meta property="og:url"
  content="http://www.imdb.com/title/tt0117500/" />
<meta property="og:image"
  content="http://ia.media-imdb.com/images/rock.jpg" />
</head>
...
</html>
```

4.4 Linked data

Linked Data je komunitní iniciativa, která má za cíl zveřejnit a zpřístupnit data a informace ukryté v jednotlivých, rozsáhlých, ale navzájem oddělených databázích. Doporučuje způsoby, jak tento obsah pomocí Sémantického webu a jeho prostředků vzájemně propojit a dávat dále k dispozici. Propojením vzniká jedna velká, decentralizovaná databáze[19] přímo v rámci prostoru webu, kterou je možné dále dotazovat např. jazykem SPARQL. Díky propojení databází z navzájem různých oborů, je možné dolovat velmi zajímavé souvislosti. Skupinu již propojených databází je možné vidět na obrázku č. 16. Mezi nimi např. DBpedia, Wordnet, Flickr a řada medicínských databází.

Důležitá data lze nalézt také v národních databázích všeho druhu. Ačkoliv je jejich sběr a uchovávání ve své podstatě placeno daňovými poplatníky a tudíž by měla být veřejná, povětšinou tomu tak není a obtížně se to mění [1]. I tady však dochází k pokroku a americká a britská vláda zveřejnila alespoň část z nich, v rámci linked data na adresách `www.data.gov` a `data.gov.uk`.

[19] V tomto kontextu se někdy SW označuje jako Giant Global Graph (GGG), obdobně k WWW.



Obrázek 16: Zdroje dat v rámci LinkedData [30]

4.5 Dublin Core

„Dublin Core je široce používaným ISO standardem pro popis digitálních materiálů jako je video, audio, obrázky, text a také webové stránky. Definuje skupinu elementů popisujících metadata pomocí RDF.“ [10]

Existuje ve dvou verzích [31]:

- Simple Dublin Core – obsahuje základní skupinu 15 elementů. Všechny jsou volitelné a mohou se opakovat. Mezi ty nejčastěji používané patří `Title`, `Creator`, `Subject`, `Description`, `Publisher`, `Date`, `Language`.
- Qualified Dublin Core – obsahuje upřesňující elementy a schémata pro detailnější popis.

Ukázka RDF dokumentu s Dublin Core:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://mojeadresa.cz/#page">
  <dc:title>Page</dc:title>
  <dc:creator rdf:resource="http://mojeadresa.cz/#me" />
</rdf:Description>
</rdf:RDF>
```

4.6 RSS

RSS je syndikační formát založený na XML. Panují kolem něho určité nejasnosti, protože existuje ve dvou verzích, které jsou zcela odlišné. První verze RSS 1.0 (RDF Site Summary) je založená na RDF, zatímco pozdější verze RSS 2.0 (Really Simple Syndication) je pouze jednoduché XML. Zřejmě díky své jednoduchosti se více prosadila druhá verze, ale pro Sémantický web je použití RDF verze nasnadě. Je velmi snadno rozšiřitelná díky použití slovníků z jiných jmenných prostorů. Ty se označují jako moduly. Často přidávaným modulem je právě Dublin Core.

Ukázka fragmentu z RDF:

```
<item rdf:about="http://mojeadresa.cz/clanek1">
   <title>Clanek 1</title>
   <link> http://mojeadresa.cz/clanek1</link>
   <description>Clanek o SW</description>
   <dc:date>2010-04-15</dc:date>
</item>
```

5 Systémy pro správu obsahu

Systém pro správu obsahu, anglicky CMS (Content Management System), je označení pro softwarové aplikace, jejichž primárním úkolem CMS je zavést jednotnou správu informací, centralizované úložiště digitálního obsahu (v tomto smyslu jím rozumíme dokumenty, články, obrázky a další multiméda … zkrátka vše, co lze převést do digitální podoby), umožnit jeho snadné zobrazení a vyhledávání. Správou se rozumí spojení pravidel a procesů tak, aby jeho elektronické uchovávání bylo považováno za řízené, spíše než za neřízené.

Jedna z mnoha definic zní: „CMS je nástroj, který umožňuje skupině odborných i neodborných pracovníků vytvářet, editovat, spravovat a publikovat (v mnoha formátech) různorodý obsah (jako text, grafiku, video, dokumenty) při dodržení centralizovaných pravidel a procesů, které zajistí ucelený a validní obsah.“ [32]

„Pomocí CMS systému se přes různé osoby ve firmě může pohybovat určitý materiál, který mohou připomínkovat, měnit anebo jinak společně modifikovat. CMS systém je častokrát používaný i jako nástroj na archivaci dokumentů. Tuto možnost běžně využívají mediální firmy (televize, vydavatelé atd.), jejichž předmět podnikání je přímo spojený s poskytováním obsahu. CMS systémy však používají i všechny velké mezinárodní společnosti na správu své firemní komunikace. Tato funkce je však čím dále důležitější i pro malé a střední firmy.“ [33]

5.1 Dělení CMS

Konkrétní požadavky na CMS systém se mohou pro každé nasazení značně lišit. Někomu stačí mít možnost snadno upravovat webovou prezentaci, jiný potřebuje mít pokrytou celofiremní správu dokumentů v průběhu celého jejich životního cyklu. Podle účelu se proto tyto systémy v průběhu vývoje rozdělily na několik dalších skupin. Krátce si tyto skupiny charakterizujeme, ale dále v práci se budeme zaměřovat na systémy pro tvorbu webu.

5.1.1 Document Management System (DMS)

Systémy pro správu dokumentů se orientují na správu dokumentů pořízených primárně v jiných programech (Office, PDF, multimediální soubory, …) a následně nahraných do uložiště v DMS. Vznikly na základě potřeby firem vypořádat se se velkým množstvím dokumentů, které jsou nedílnou součástí jejich business procesů. Hlavní funkčnosti jsou tedy zaměřeny na uchování, sdílení, verzování těchto dokumentů v průběhu jejich životní cyklu. Dokument je v těchto systémech samostatná položka, mohou mu být definována přístupová práva nebo workflow. Více dokumentů je možné sdružovat do skupin na základě definované taxonomie.

5.1.2 Enterprise Content Management (ECM)

Enterprise Content Management představuje kompletní řešení pro veškeré firemní procesy. Pokrývá úplné řízení a zpracování obsahu, který firma vytváří a využívá. Spojuje řešení groupware (podpora spolupráce pracovníků firmy), WCMS pro web a DMS systému pro dokumenty. Časté je také propojení s ERP a CRM systémy. Cílem je celkové usnadnění a zrychlení přístupu k firemním informacím, snížení celkových nákladů.

5.1.3 Web Content Management System (WCMS)

Jak už název napovídá, primárním zaměřením WCMS je obsah související s webovou prezentací, ať už se jedná o podnikový intranet nebo rozsáhlé řešení pro elektronický obchod. Systém musí především umožňovat vytvářet a editovat HTML stránky a celou strukturu webu. Těmto systémům se bude pod obecným názvem CMS věnovat následující část práce.

5.2 Historie a vývoj CMS systémů

S růstem Internetu se stal web jedním z hlavních komunikačních médií, a to nejen pro firmy a státní správu, ale i pro jednotlivce. Je jasné, že dnes již nemůže každý, kdo si chce stránky vytvořit nebo jen upravit, rozumět jazyku HTML a dalším webovým technologiím tak, jak tomu bylo před 15 roky. Tehdy ale nebyla jiná možnost. Časem vznikly WYSIWIG editory jako Microsoft Frontpage nebo Adobe Dreamweaver, které tvorbu stránek usnadnily, ale bariéru pro laiky zcela neodstranily. I pro jednodušší zásahy, jako je upravení textu nebo vložení obrázku, byl často pro jistotu povoláván drahý profesionál.

Dalším problémem byly rostoucí nároky na kvalitu a rozsah webových prezentací. Vytvářet manuálně jednotlivé stránky a individuální řešení pro každého klienta se přestávalo vyplácet. Proto si programátoři začali vytvářet pomocné nástroje, které jim pomohly při rutinní práci. Sami už jen stránky naplnili obsahem a upravili vzhled dle přání zákazníka.

Tyto nástroje se časem vyvinuly v samostatné produkty, které označujeme souhrně jako CMS systémy. Trh s CMS programy je dnes obrovský, existuje nepřeberné množství programů nabízených jako svobodný software (Open Source) i komerčních řešení (Kentico CMS). Existují i firmy, které vydělávají na Open Source CMS jiných firem, ne jejich prodejem, ale prodejem podpory (instalace, konfigurace, školení) pro daný produkt.

CMS se člení dle řady kritérií, například rozsahu řešení, použitého vývojového prostředí nebo cílové skupiny. Pro CMS se někdy používají podobné termíny redakční či publikační systém. Tyto názvy nejsou zcela přesné. Redakční/publikační systémy jsou obecně podmnožina WCMS. Většinou se jedná o jednodušší systémy s důrazem na jednoduchost použití pro neodborné uživatele.

5.3 Základní funkce a vlastnosti CMS

Základní funkcí webového CMS je správa HTML obsahu a ostatního materiálu, který se zveřejňuje na webu, a také údržba webu jako takového. Vyjmenujme funkce v následujících bodech:

- Tvorba, editace a publikace dokumentů (článků) zpravidla prostřednictvím webového rozhraní, často s využitím jednoduchého online WYSIWYG editoru nebo jednoduchého systému formátování textu (není nutná znalost HTML).
- Tvorba struktury a vzhledu webu.
- Správa uživatelů a jejich zařazení do rolí/skupin.
- Řízení přístupu k dokumentům, zpravidla se správou uživatelů a přístupových práv.
- Vyhledávání dokumentů.
- Správa diskusí či komentářů.
- Správa souborů.
- Správa obrázků, případně galerií.
- Kalendářní funkce.
- Sledování statistiky přístupů.

Profesionální systémy dále nabízejí propracované workflow pro spolupráci více uživatelů a editorů obsahu, podporu stagingu (přenos mezi vývojovým a produkčním serverem), personalizaci stránek pro každého uživatele, komunitní prvky a mnoho dalších funkcí. Jednotlivé části funkcionality jsou obvykle u CMS systémů členěny do oddělitelných modulů.

Elementární vlastností dobrého WCMS je oddělení vlastního obsahu od prezentační části. Toho je obvykle docíleno použitím šablon. Ty umožňují zobrazení jednoho obsahu různými způsoby, např. seznam produktů vs. detail produktu nebo generování RSS. Jednotkou obsahu je webová stránka, která má definované umístění v rámci struktury webu. Stránky jsou mezi sebou intenzivně prolinkovány. Moderní CMS systém by měl mít podporu „URL rewritingu“, tedy umožňovat mít vytvořený obsah dostupný pod adresou, která je kvalitní z hlediska SEO optimalizace.

5.4 Sémantické technologie v CMS systémech

Tato práce je o Sémantickém webu. Proto se nyní zaměříme na to, jak vybrané CMS systémy umí pracovat s jeho technologiemi.

Obecně v každém CMS systému lze manuálně vytvářet HTML kód a tedy i manuálně publikovat sémanticky anotovaná data. Tento přístup však nelze považovat za zabudovanou podporu. CMS systémy samy vznikly s cílem usnadnění práce autorů, ne manuálního psaní kódu, nehledě na to, že anotovat správně data v RDF/OWL je poměrně složité i pro odborníky.

Za CMS systém, který alespoň částečně podporuje Sémantický web lze považovat pouze ten, který splňuje některý z následujících bodů:

- napomáhá uživateli v tvorbě sémantického kódu nebo tento kód sám generuje
- přes speciální rozhraní dává k dispozici data v sémantických formátech
- umí zpracovat sémantický obsah z jiného zdroje

CMS systémy obsahují uvnitř svých databází strukturovaná data. Definují různé typy dokumentů a jejich vlastnosti. Např. takový typ „článek“ má vlastnosti jako text, autor, datum publikace. To vše jsou informace, které se po převedení do HTML podoby stránky navzájem smíchají. Cílem pro sémantický CMS systém tedy je, aby tuto podobu doprovodil dodatečnými informacemi tak, aby šlo tyto vlastnosti stále rozlišit.Tento model dokumentového typu a jeho vlastností, lze snadno převést do podoby RDF.

Jak na tom tedy existující CMS systémy s podporou jsou? Rovnou si můžeme prozradit, že všeobecná podpora zde rozhodně není, ale některé systémy adaptují alespoň některé technologie. Světlou výjimku tvoří systém Drupal, který si dal podporu Sémantického webu jako jednu ze svých priorit. Jeho autor Dries Buytaert na koferenci Drupalcon (Boston, 2008) [34] zhodnotil jednotlivé éry webu z pohledu CMS systémů. Web 1.0 označil za klasický systém pro správu obsahu, Web 2.0 přidává nutnost mít podporu uživatelů a „neomezenou rozšiřitelnost“ a Web 3.0 je o „neomezené interoperabilitě“ ve smyslu přenositelnosti dat a aplikačních rozhraních.

5.4.1 Výběr systémů

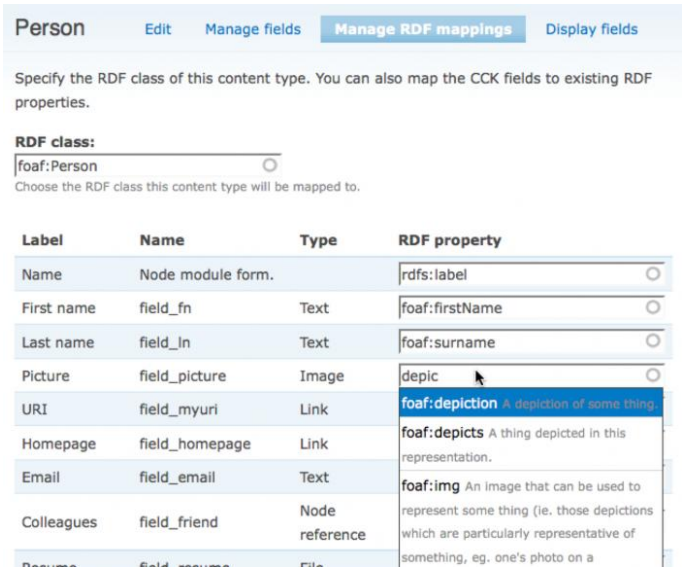
Jedním z cílů této práce je srovnat použití technologií Sémantického webu v různých CMS systémech. Těch je ale neuvěřitelné množství, a proto se zaměříme pouze na ty nejpoužívanější. Samotný úkol zjistit nejrozšířenější systémy však není zcela triviální. Využil jsem proto existujícího reportu [35], který společně vytvořili agentura Water&Stone a magazín CMSWire. Ten obsahuje analýzu provedenou na základě povědomí o značce, počtu vyhledávání, návštěvnosti stránek a přítomnosti na sociálních sítích. Jejím výsledkem bylo zjištění, že nejpoužívanějšími CMS systémy jsou Joomla, Drupal a WordPress.

5.4.2 Drupal

Drupal je volně dostupný Open Source CMS, podporovaný silnou komunitou uživatelů. Umožňuje publikovat a spravovat širokou škálu webových prezentací – od rozsáhlých projektů, po osobní stránky, blogy nebo diskusní fóra. Jeho historie sahá do roku 2001, kdy jej vytvořil holandský student Dries Buytaert. Je naprogramován v jazyce PHP a aktuálně se blíží vydání jeho sedmé verze, která je pro nás obzvláště zajímavá, protože přímo obsahuje a používá sémantické technologie.

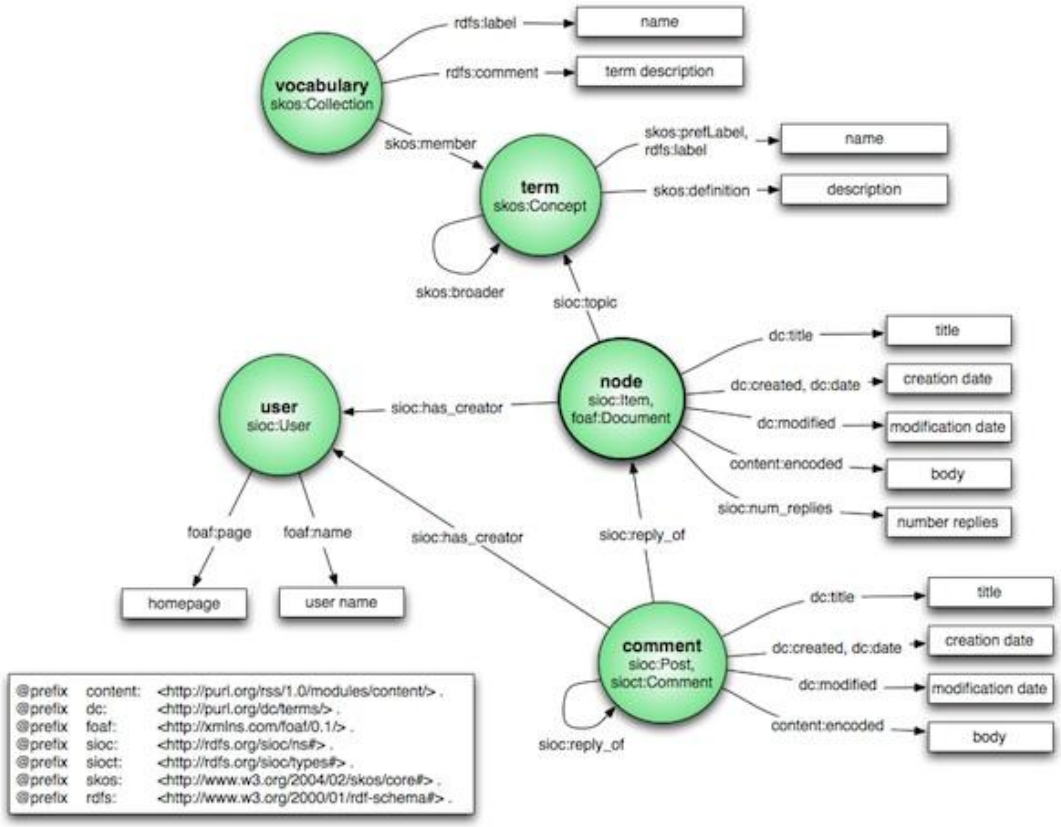
Drupal staví na rychlém a stabilním jádře, jehož funkcionalitu lze rozšiřovat pomocí obrovského množství doplňujících modulů. Podporu RDF a další sémantická rozšíření bylo donedávna možné přidat pouze tímto způsobem. Drupal 7 však přinese podporu zabudovanou přímo do jádra systému a to zapnutou ve výchozím nastavení. Vzhledem k rozšířenosti Drupalu si od toho komunita zabývající se Sémantickým webem hodně slibuje.

Ačkoliv první rozšíření vznikla již dříve, významným byl teprve modul „RDF“ vytvořený v roce 2007, který přinesl obecnou podporu pro tvorbu RDF. V roce 2008 pak vznikl modul „RDF CCK“, který umožnil namapovat RDF/RDFa přímo na obsah. Každá položka obsahu se v Drupalu (ale i některých jiných CMS) nazývá „node“ – uzel. Uzly obvykle reprezentují jednotlivé stránky webu. Mohou být vytvořeny, editovány, smazány nebo mohou být např. přiřazeny do různých kategorií. Pro vytváření typů těchto uzlů, tj. typů obsahu (content types), z administračního rozhraní existuje modul CCK (Content Construction Kit). Pomocí něj mohou administrátoři definovat jednotlivá pole typů uzlu a jejich charakter. Tato pole vlastně určují vlastnosti typů, např. `title` pro jméno článku, `summary` pro jeho souhrn. Modul RDF CCK přidává do administračního rozhraní možnost namapovat tyto vlastnosti na libovolné vlastnosti z existujících ontologií, např. titulek na dc:title.



Obrázek 17: Mapování typu obsahu na RDF

Na obr. 18 je vidět výchozí mapování Drupalu 7. Obsahuje pět typů entit – `node`, `comment`, `user`, `vocabulary`, `term`. U těchto entit se sledují atributy jako `title`, `creator`, `created`, `modified`, `comment count`, `reply_of` a všechny další pole namapovaná pomocí RDF CCK. Aby bylo anotování smysluplné a nevytvářely se pouze izolované grafy, je dobré používat veřejné ontologie. Drupal mapuje vlastnosti na RDF třídy z již představených ontologií Dublin Core, FOAF. Dalšími jsou SIOC a SKOS. SIOC je ontologie pro popisování informací z online komunit (diskuzní fóra, wiki, blogy, ...) a SKOS slouží na klasifikaci a taxonomii.



Obrázek 18: Drupal 7 mapování na ontologie [36]

Nesmíme zapomenout na správnou identifikaci samotných subjektů a objektů RDF. Drupal 7 pro tento úkol používá speciálních URL nazývaných „cool URIs“ nebo taky „curries“. Pro každou entitu v systému existuje unikátní URI, ve které je obvykle použit typ entity a jeji id. Např.

```
node/4
user/225
comment/68#comment-68
```

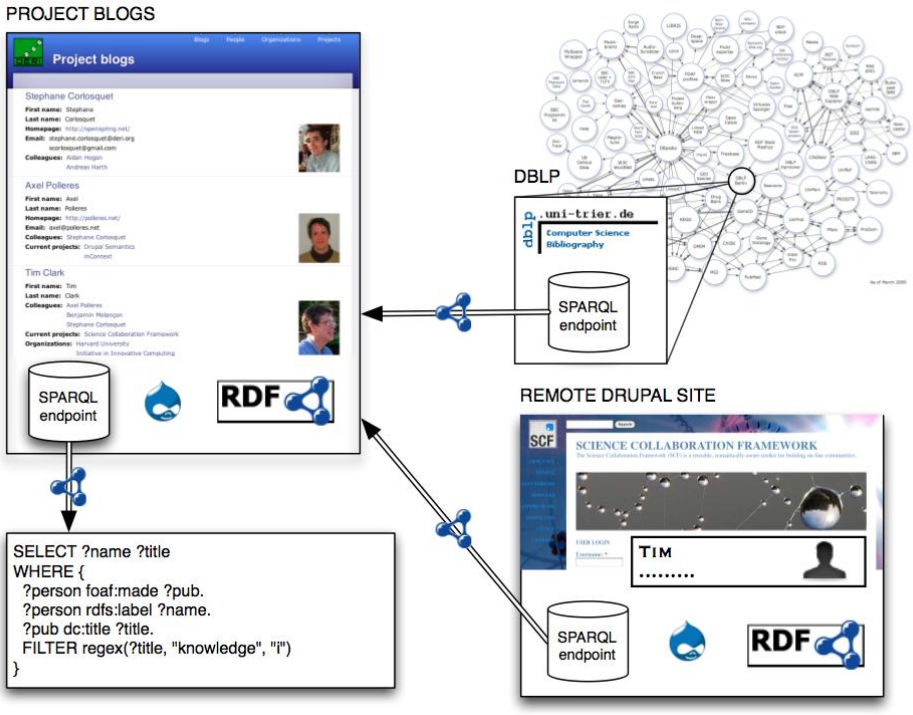
Upravený fragment HTML anotované stránky z Drupalu může vypadat např. takto:

```
<div typeof="sioc:Item foaf:Document"
     about="/drupal7/?q=node/1">

<span datatype="xsd:dateTime"
      content="2010-05-12T03:19:12+02:00"
      property="dc:date dc:created">Wed, 05/12/2010 - 03:19
</span>
<span rel="sioc:has_creator">
  <a property="foaf:name"
     typeof="sioc:UserAccount"
     about="/drupal7/?q=user/1"
     href="/drupal7/?q=user/1">mivra</a>
</span>

<div property="content:encoded">
  <p>Obsah stranky</p>
</div>
</div>
```

RDFa data na stránce se generují za běhu. Mimo RDF CCK je možné RDF generovat také programově – RDF API je k dispozici.

Důležitou funkcionalitou Drupalu je, že poskytuje také dotazovací rozhraní pro jazyk SPARQL (endpoint). Toto rozhraní je postavené na knihovně PHP ARC2. Po její instalaci se vytvoří RDF úložiště, které obsahuje všechna data vygenerovaná pomocí RDF CCK. Naplnění úložiště se provede naindexováním webu přes administrační rozhraní.

Drupal 7 bude mít také podporu tzv. RDF proxy, což je způsob, jak se připojit k datům z různých webů a automaticky aktualizovat informace na vlastní stránce. Na obr. 19 je ukázka znazorňující stránku s blogy, data na této stránce se získávají z různých zdrojů – LinkedData, cizí i vlastní webová stránka. Kdykoliv se změní data ve svých původních zdrojích, stránka s blogy bude aktuální [37].



Obrázek 19: RDFProxy a Linked Data [37]

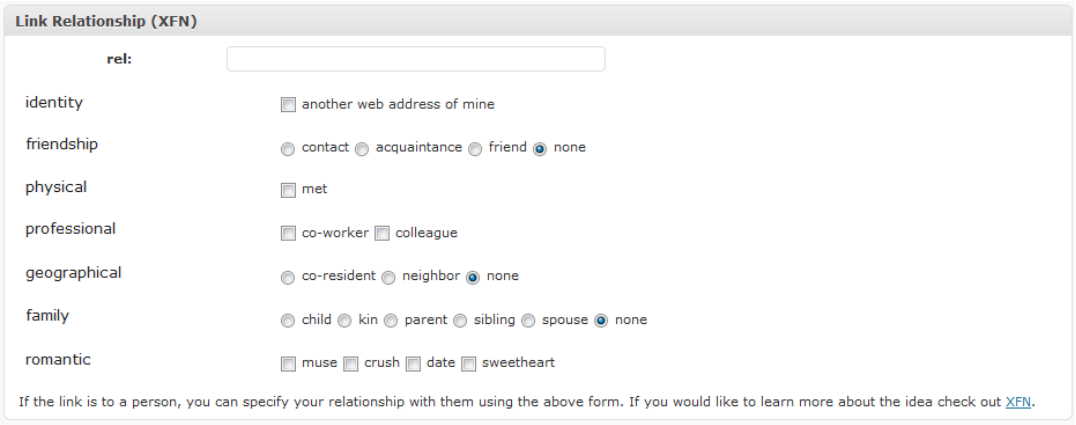
Další technologií, kterou Drupal využívá, je knihovna Triplify. Ta slouží k exportu RDF dat přímo z relační databáze. V rámci Triplify jsou definovány konfigurační soubory, které obsahují SQL dotazy, pomocí kterých se získávají data a převedou se do podoby RDF (syntaxe XML i N3) nebo i JSON.

Co se týká mikroformátů, tak v základu Drupal žádnou funkcionalitu neobsahuje. Trochu překvapivě není k dispozici ani žádný modul, který by ucelenou podporu dodával. Existuje však několik oddělených malých modulů, které přidávají možnost vytvářet mikroformáty `hCard/vCard` a omezeně XFN nebo hCalendar.

5.4.3 WordPress

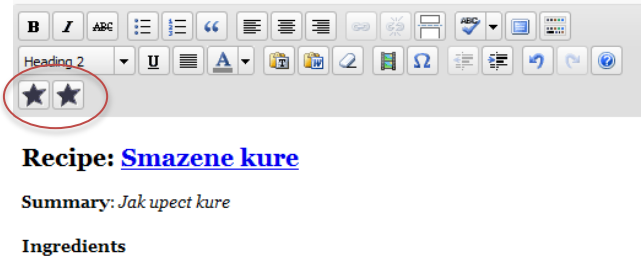
WordPress je jeden z nejpoužívanějších redakčních a publikačních systémů. Zaměřuje se především na blogy. Jedna instalace podporuje obvykle jen jeden web. Systém vznikl v roce 2003 a je šířen pod licencí GNU GPL 2.

Co se sémantiky týče, nemá WordPress příliš mnoho zabudované podpory. Ve výchozí podobě umí přidávat mikroformát XFN k odkazům (obr. 20). Ostatně XFN z oblasti blogů vzešel. Dále umí systém vytvářet formáty hCard resp. vCard. Ty přidává například k autorům komentářů u příspěvků. Anotuje také značky (tagy) příspěvků pomocí rel-tag. Bohužel ve standardním nastavení nemají adresy na konci své URL uvedený onen tag (předává se až pomocí parametrů) a tedy mikroformát není použit správně. Toto lze však změnit globální úpravou formátu adres.



Obrázek 20: Mikroformát XFN pro odkazy

Funkcionalitu WordPressu lze rozšířit pomocí pluginů. Těch jsou až tisíce a několik z nich také přidává další mikroformáty nebo podporu RDF. Existují rozšíření pro mikroformát hRecipe (recepty) a hReview (recenze). Ty se integrují přímo do ovládacího menu WYSIWYG editoru. Po kliknutí na přidaná tlačítka se zobrazí dialog, kde lze vložit patřičná data.



Obrázek 21: hRecipe, hReview plugin

Dalšími sémantickými doplňky jsou hned dvě rozšíření pro vkládání/vyznačení Creative Commons licence. Lze nastavit jak licenci globální, tak pro jednotlivé články. Výborně integrované je rozšíření pro službu Zemanta. K automatickému otagování poslouží dobře také OpenCalais a YQL auto tagger.

Co se týká podpory RDF, tak je pro WordPress k dispozici plug-in wp-RDFa. Ten aktuálně podporuje pouze slovníky FOAF a Dublin Core. Pro FOAF umožňuje nastavit mimo osobních

informací také vztah `knows`. Z Dublin Core používá elementy `title`, `creator` a `date`. Bohužel způsobem vložení způsobuje nevaliditu stránky.

WordPress také již v základu umožňuje přidat ke každému článku libovolný počet tzv. custom fieldů. To jsou obecně pole pro uchování libovolných typů metadat, která lze při zpracovávání stránky programově zpracovat a různě zobrazit na stránce.

Nalezl jsem také neoficiální rozšíření RDF Tools a SparqlPress, které by měly údajně vytvořit SPARQL endpoint a umožnit přístup k LinkedData, ale je to již poměrně dlouho, co byl aktualizován a nepovedlo se mi ho zprovoznit.

5.4.4 Joomla

Joomla je oblíbený víceúčelový Open Source redakční systém, postavený také na jazyku PHP a databázi MySQL. Jeho vznik se datuje na počátek roku 2005, ale ve své podstatě šlo jen o přetransformování CMS systému Mambo, od kterého se vývoj oddělil. Na rozdíl od Drupalu Joomla obsahuje hodně ze své funkcionality přímo v instalaci a není nutné si stahovat tolik rozšíření a modulů.

Co se týká sémantiky, tak na tom není Joomla příliš dobře. Když pomineme základní schopnost editovat kód přímo a možnost nastavení těch nejzákladnějších metadat stránky, tak vlastně neobsahuje nic dalšího. V rámci doplňujících modulů je možné dodat podporu pro Dublin Core. Pro Joomlu je také k dispozici Zemanta rozšíření, Open Calais zřejmě ne. Překvapivé je, že ani v rámci její velké komunity nevzniklo téměř žádné rozšíření pro mikroformáty, nebo se mi ho nepodařilo nalézt.

Pro export dat v RDF lze využít opět Triplify, který má i pro Joomlu předvytvořené konfigurační soubory. Existuje také rozšíření pro e-shop modul „VirtuaeMart“, které k němu přidává anotaci produktů pomocí ontologie GoodRelation.

5.4.5 Ostatní CMS systémy

Mimo tři výše popsané systémy jsem prozkoumal i několik dalších CMS systému. Jmenovitě např. Typo3, Alfresco, oCportal, SiteCore, Ektron, Umbraco, BlogEngine. To jsou systémy nejenom „ze světa“ PHP a Open Source, ale i komerční systémy postavené na platformě .NET nebo Java. Výsledek je bohužel téměř všude stejný (s výjimkou ocPortal a BlogEngine). Přímá podpora pro technologie Sémantického webu zabudována není, v lepším případě jsou k dispozici alespoň nějaká rozšíření. Domnívám se, že je to způsobeno tím, že na trhu prostě není dostatečná poptávka. Pro koncového uživatele webu není zatím příliš možností, jak využít sémantických anotací. Také zde doposud nebyl nikdo, kdo by udal jasný směr tak, jako to nyní činí Drupal 7.

6 Návrh a implementace

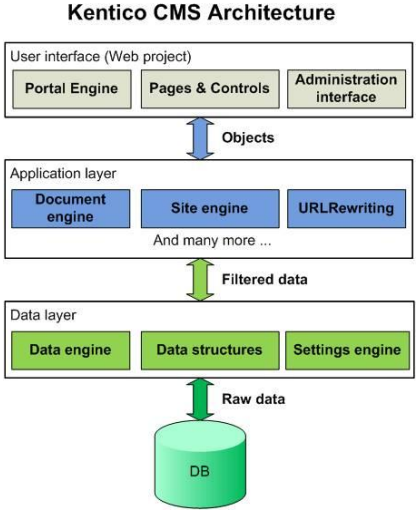
Již předchozí kapitola poměrně jasně určila, jakým směrem by se měl návrh rozšíření pro systém Kentico CMS vydat. Mělo by jít zejména o obecnou podporu pro tvorbu sémanticky označeného obsahu a jeho automatické generování. Před popisem kenkrétních návrhů si však představíme samotné Kentico.

6.1 Kentico CMS

Kentico CMS je komplexní, třívrstvý, vysoce flexibilní systém s extrémní škálovatelností, skládající se z mnoha modulů. Tento CMS systém je stavěn až na výjimky výhradně na technologiích firmy Microsoft, využívá .NET Framework (2.0+), databázi Microsoft SQL Server (MS SQL 2005+) a celé je napsáno ve vysokoúrovňovém, objektově orientovaném programovacím jazyce C#, který je od vzniku s platformou .NET úzce spjat.

6.1.1 Architektura

Architekturu Kentico CMS lze rozdělit do tří hlavních celků (obr. 22). Datová vrstva získává požadovaná data z databáze dělené do jednotlivých tabulek zpravidla v závislosti na jednotlivých modulech nebo funkčních celcích. Dále je filtruje a předává aplikační vrstvě, která je zpracuje a jako objekty předá uživatelskému rozhraní.

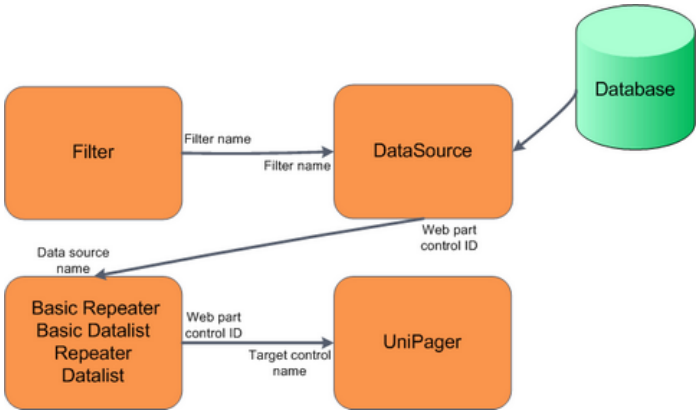


Obrázek 22: Architektura Kentico CMS

Kentico CMS nabízí uživatelům dva modely pro vývoj webu. První z nich je označován jako Portal Engine a je určen pro tvorbu webu uživateli nedisponujicími pokročilými znalostmi programování. Webová stránka je v tomto případě jednoduše vytvořena přes webové rozhraní za použití webových částí (z angl. webpart). Pro uživatele, kteří dávají přednost užití standardní ASP.NET architektury

a vývoji ve Visual Studiu, je určen ASPX templates model. Tento přístup vyžaduje již zmíněné Visual Studio, znalost jazyka C# nebo VB.NET a přehled v ASP.NET. Oba přístupy mají své výhody a nevýhody vyplývající z toho, komu jsou určeny.

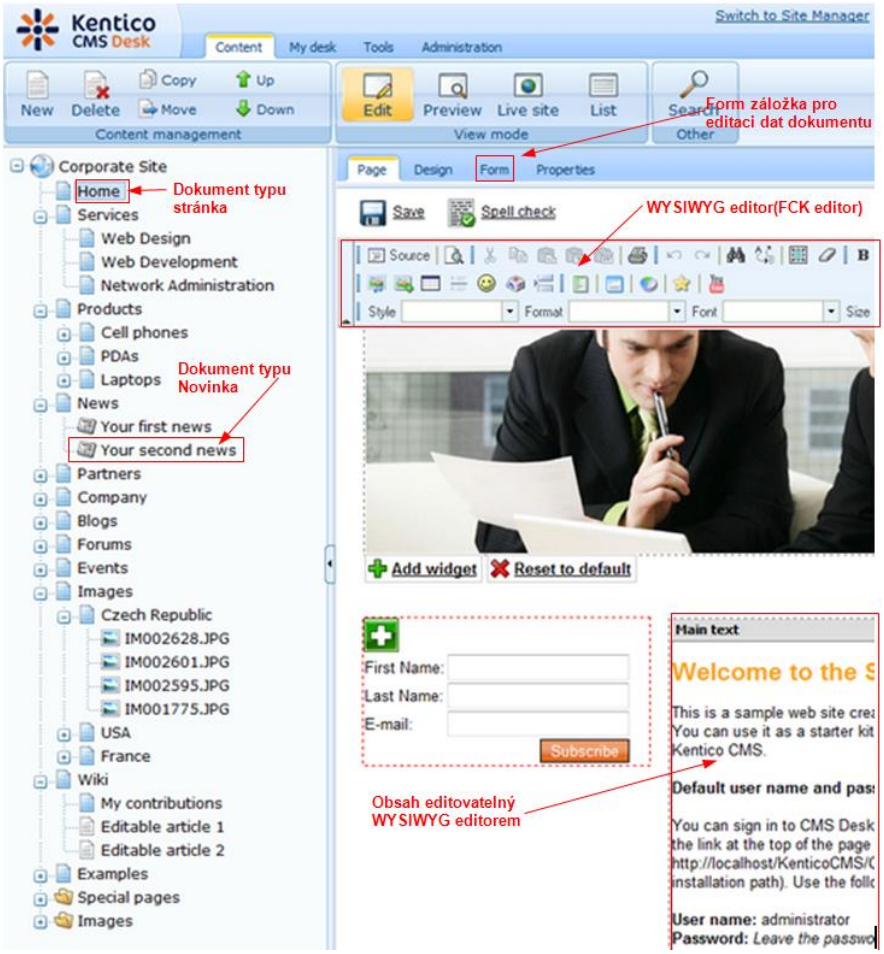
Webové části jsou uživatelskými ASCX kontroly, které tvoří bloky obsahu nebo kombinaci obsahu a funkcionality. Jsou základními stavebními kameny stránky tvořené v Portal enginu. Mají mnoho vlastností jako jsou kódové jméno, název zobrazovaný v uživatelském rozhraní nebo typ – existují části zobrazující obsah, části sloužící jako datový zdroj (DataSource) a části filtrující data získaná datovým zdrojem. Webové části typu datového zdroje jsou navrženy k získávání dat z databáze a předávání jiným částem k zobrazení. Obr. 23 ukazuje, jaké mohou být interakce mezi jednotlivými typy webových částí.



Obrázek 23: Interakce webových částí

Existuje mnoho dalších vlastností webových částí týkajících se zpracování, zobrazování, užití AJAX, časových zón, kešování a dalších, které mohou být závislé na typu části.

Obsah webu není tvořen pouze pomocí HTML stránky. Nachází se zde také odkazy, obrázky, videa, textové dokumenty různých formátů a další. Tento různorodý obsah je v Kentico CMS uložen ve stromové struktuře stránek a dokumentů. Dokument obsahuje strukturovaná data, která mohou být zobrazována na stránkách. Stránka je v podstatě jen speciální typ dokumentu zobrazující určitý obsah. Každý dokument je určitého typu reprezentující jeho účel – tedy například článek, novinka, prodejní artikl internetového obchodu, stránka a mnoho dalších. Všechny dokumentové typy mají svoji datovou strukturu (políčka v databázi), vzhled editačního formuláře, transformace (design) a jiná nastavení. Dokumenty typu stránka obsahují převážně nestrukturovaný obsah umístěný v editovatelných oblastech dostupný ze záložky Page v administračním rozhraní. Oproti tomu záložka Form je určena pro strukturovaná data dokumentu konkrétního typu, která jsou spjata s konkrétními databázovými tabulkami určitého typu dokumentu.



Obrázek 24: Uživatelské rozhraní Kentico CMS

Obsah je ukládán dvěma různými způsoby. Buď jako text uložený v XML formátu ve vlastnostech dokumentu. Editace probíhá na záložce Page, kde lze jednoduše editovat WYSIWYG editorem (v Kentico CMS FCK editor) integrovaným do uživatelského rozhraní. Druhým způsobem je uložení v oddělených databázových tabulkách, kde má každé políčko formuláře svázané s typem dokumentu vlastní sloupec v odpovídající tabulce. Tato data mohou být editována pomocí SQL příkazů a jejich obsah je zobrazován díky XSLT nebo ASCX transformacím speciálních kontrolů nebo webových částí. Transformace je kombinace HTML kódu s ASP.NET příkazů a tzv. Data Binding výrazů (Obrázek 25).

```
<div style="text-align:center;border: 1px solid #CCCCCC">
<h2><a href="<%# GetDocumentUrl() %>"><%# Eval("ComputerName") %></a><h2>
<img src="<%# GetFileUrl("ComputerImage") %>?maxsidesize=120" />
</div>
```

Obrázek 25: ASCX Transformace

Dokumentový typ je plně nastavitelný, není problémem přidávat, odebírat nebo upravovat jeho políčka. Uživatel má také možnost definovat své vlastní dokumentové typy. Nový dokumentový typ mohou v Kentico CMS vytvářet uživatelé-administrátoři, kteří mají přístup do speciální části administračního rozhraní nazývané Site Manager. Nový dokumentový typ je vytvořen pomocí průvodce v němž je prvním krokem definování kódového označení a názvu, který se bude zobrazovat

v uživatelském rozhraní. V následujících sedmi krocích se definují název databázové tabulky pro ukládání detailů daného dokumentového typu, její priparní klíč, její sloupce a datové typy, povolené rodičovské dokumentové typy, webové stránky, na kterých může být daný typ použit a způsob indexování pro vyhledávací modul. Výsledkem úspěšného průchodu tímto průvodcem je nový dokumentový typ, který má vlastní databázovou tabulku a je zaznamenán ve třech systémových tabulkách. Dále již se o operace nad těmito tabulkami stará samotný systém.

6.2 Implementační platforma

Jedním z cílů této práce bylo navrhnout sémantická rozšíření pro publikační systém Kentico CMS. Toto řešení je postaveno zcela na platformě ASP.NET 2.0, což je jedna z technologií .NET frameworku společnosti Microsoft.

6.2.1 .NET Framework

.NET Framework je rozsáhlá vývojová platforma, která umožňuje tvorbu rozličných softwarových aplikací. Můžeme na ní vyvíjet nejen klasické aplikace pro Windows, ale také webové aplikace a služby, aplikace pro mobilní zařízení a mnoho dalších. Verze .NET Framework 1.0 byla představena v roce 2002, v dubnu 2010 byla uvolněna aktuální verze 4.0 společně s vydáním Visual Studio 2010 [38].

Celý framework je souborem mnoha technologií. Je v něm obsažena celá řada knihoven pro běžné úkoly jako je čtení a zápis do souborů, renderování grafiky, práci s databázemi, podpora XML, atd. Ty se souhrnně nazývají BCL (Base Class Library). Dále obsahuje běhové prostředí CLR (Common Language Runtime), které zajišťuje běh a kompilaci aplikací[20]. Jedná se o jakýsi virtuální stroj, podobně jako má jazyk Java. Programy napsané v .NET Frameworku tedy nejsou nativní a jejich kód je označován jako řízený (managed code). To má výhody z hlediska bezpečnosti – nepracuje se s ukazateli, nehrozí přetečení či podtečení, je zde striktní typová kontrola na úrovni CLR, ale je to také vykoupeno obecně menší rychlostí oproti nativnímu kódu. Existují však techniky[21], které výkon zlepšují natolik, že v běžných programech není výkonový rozdíl příliš poznat.

Díky běhu nad virtálním strojem jsou také aplikace principiálně přenositelné mezi různými systémy. Pro běh na systémech unixového typu (Linux, FreeBSD, MacOS X, Solaris) je možné využít platformu Mono, jejíž vývoj zaštiťuje firma Novell. Bohužel vývoj této platformy je trochu zpožděn a často nelze využívat všech možností hlavního .NET Frameworku.

Obrovskou výhodou .NETu je možnost používat různé programovací jazyky. Je to díky tomu, že CLR je implementací specifikace CLI (Common Language Infrastructure). Ta definuje jazykově neutrální platformu pro vývoj a běh programů. Všechny jazyky vyhovující CLI mohou být

[20] Kompiluje se do byte-kódu zvaného CIL (Common Intermediate Language) nebo někdy MSIL
[21] Předkompilace, Just-In-Time kompilace

zkompilovaný do stejného mezi-kódu, který je následně prováděn. Těchto jazyků jsou dnes již desítky. Hlavními jazyky jsou C#, Visual Basic .NET a Managed C++, ale dále jsou zde F#, IronPython, J# (Java), L# (Lisp), P# (Prolog), #Smalltalk a mnohé další. Všimněme si, že v tomto seznamu jsou jazyky zcela různé povahy – funkcionální, imperativní, staticky typované, dynamicky typované a přesto všechny dokáží běžet nad CLR.

6.2.2 ASP.NET

ASP.NET je technologie určená pro vývoj webových aplikací nad .NET Frameworkem. Nahradila původní technologii ASP (Active Server Pages). Ačkoliv mají podobné jméno, jsou obě zcela odlišné. Původní ASP bylo spíše podobné PHP. Kód stránky obsahoval mix HTML a skriptovacího jazyka (VBScript, JScript). ASP.NET přišlo se zcela odlišným, revolučním přístupem nazvaným WebForms. WebForms se inspirovali ve vývoji tlustých desktopových aplikací pro Windows. Ideou bylo přenést model tvorby Visual Basic aplikací na web a tím i umožnit vývojářům tohoto jazyka snadno vytvářet webové stránky téměř bez znalosti HTML.

ASP.NET klade důraz na produktivitu vývojářů. Místo přímého psaní HTML kódu se pro tvorbu stránek využívá ovládacích prvků, tzv. kontrolů. Takovými kontroly jsou např. tlačítko (Button), textové pole (Textbox), nápis (Label) a další. Tyto komponenty jsou důvěrně známé ze světa desktopových aplikací, avšak ASP.NET obsahuje množství dalších (GridView, Login, ...), které jsou specifické pouze pro něj. Dokonce lze přidávat komponenty vlastní. Všem těmto ovládacím prvkům lze deklarativně přiřazovat určité vlastnosti, zpracovávat jejich události, atd. O vykreslení, či spíše vygenerování odpovídajícího HTML kódu, se starají komponenty samy.

Každá stránka v ASP.NET aplikaci je složena jednak z HTML kódu s komponentami a jednak i z kódu v pozadí, tzv. code-behind. Ten zařizuje zpracování události komponent a vůbec se stará o funkcionalitu celé stránky. Tento kód je buď možné umístit odděleně přímo do HTML souboru (koncovka .aspx) nebo se doporučuje mít code-behind v souboru vlastním (koncovka .cs). Tím se oddělí vzhled a obsah aplikace od aplikační logiky, díky čemuž je aplikace přehlednější a nevzniká tzv. spaghetti code. V rámci kódu stránky lze využívat veškerých funkcí a tříd, které .NET Framework nabízí.

Je důležité zmínit, jak se v ASP.NET událostně řízené programování vyrovnává s bezstavovostí protokolu HTTP. Aby totiž mohlo dojít k nějaké události a následně jejímu zpracování, je nutné rozpoznat, že se stav vůbec změnil. K tomu se používá důmyslného mechanizmu zvaného ViewState. Jedná se v podstatě o jakousi nadstavbu nad protokolem HTTP, která umožňuje udržovat stavové prostředí při komunikaci s prohlížečem klienta. Před odesláním stránky klientovi se ASP.NET postará o serializaci vlastností všech komponent, které nemají defaultní hodnotu. Tato serializovaná data se v podobě řetězce přidají do stránky v hodnotě skrytého formulářového pole. V případě, že uživatel

provede na stránce nějakou akci, která vyžaduje obnovení stránky (tzv. postback), jsou tato data odeslána zpátky na server, kde se z nich zrekonstruuje původní stav.
Celé ASP.NET je značně modulární. Díky objektově orientovanému modelu lze chování poměrně snadno přizpůsobit. Obvykle stačí napsat nějakou třídu (s využitím dědičnosti) a použít ji místo vestavěného řešení nebo overridovat nějakou metodu. Nastavení aplikace – přístup do databáze, registrace modulů a handlerů, přidávání referencí, atd., lze provádět v souboru web.config.

6.3 Návrh rozšíření

Je jasné, že sémantické anotování je orientované na jednotlivá pole dokumentového typu, spíše než na na dokument jako celek. Ten může klidně obsahovat pole, která se nijak neoznačují nebo ani nevypisují. Kentico ani nemá přímo zabudovanou podporu na zobrazování pouze konkrétního typu či objektu. Obvykle je stránka vytvořena hierarchicky – od nejvyšší urovně šablony po ty nižší. Každá vrstva obsahuje různé webové části nebo datové zdroje, ze kterých se díky transformacím tvoří kód. Obsah vkládaný na záložce Page se zobrazí pomocí speciálních tzv. editovatelných webových částí. Z toho plyne, že jsou tři místa, kde se tvoří samotný kód stránky – programově uvnitř webové části, na záložce Page (tam tvoří obsah uživatel) a pomocí transformace. Všechny tyto části by proto měly být pokryté nějakou podporou pro tvorbu sémantického obsahu.

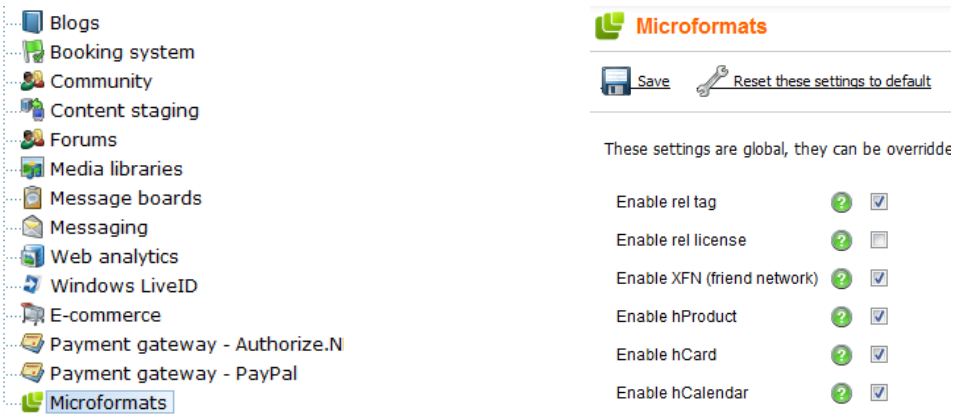
Konkrétně tedy budou vytvořeny nové webové části, budou upraveny výchozí transformace (s možností volat pomocné funkce pro vložení nějaké sémantické části obsahu) a nakonec úpravu FCK editoru, což bude sloužit především koncovému uživateli/editorovi.

6.4 Podpora pro mikroformáty

Podporu mikroformátů je nutné rozdělit na dvě části. Na to co systém automaticky vygeneruje sám a na to co může vložit sám uživatel. Obě varianty jsou však ovlivňovány nově přidanou položkou „Microformats“ v globálním nastavení webu (obr. 26). V té je možné globálně nebo pro každý z webů zvlášť zapínat/vypínat podporu jednotlivých mikroformátů. Pokud je mikroformát vypnutý nenabízí se v nabídkách a ani se automaticky negeneruje. Avšak zůstává tam, kde již byl vložen jako statický text. Do systému přibyla podpora pro mikroformáty `rel-tag`, `rel-license`, `rel-nofollow`, `XFN`, `hCard`, `hCalendar` a `hProduct`.

`Rel-tag` je pochopitelně použit především u blogů. Automaticky se generuje ve webové části „tag-cloud“ a byl přidán do standardní transformace pro zobrazení příspěvku na blogu. Tedy tam, kde se tagy zobrazují. Při implementaci nastal problém s adresou odkazu. Jak již bylo zmíněno `rel-tag` používá k určení jména tagu nepříliš vhodně poslední část URL. Ta definuje tzv. tag space. Kentico však standardně odkazuje na tyto tagové prostory pomocí parametrů v adrese. Proto bylo nutné toto chování změnit. U tag-cloudu přibylo nastavení pro používání „friendly-url“. Nyní lze specifikovat

formát, resp. adresu tagového prostoru. Tedy například `/tagspace/{tagid}/{tagname}`. Pro úpravu transformace bylo nutné upravit funkci `BlogFunctions.GetDocumentTags`, která vrací řetězec s odkazy do tagových prostorů. Tyto odkazy jsou však podle uvedeného formátu bez přípony, a proto je nutné nastavit systém tak, aby toto uměl. Nejjednoduším způsobem je zřejmě přímo konfigurace na IIS pomocí modulu URL rewrite.

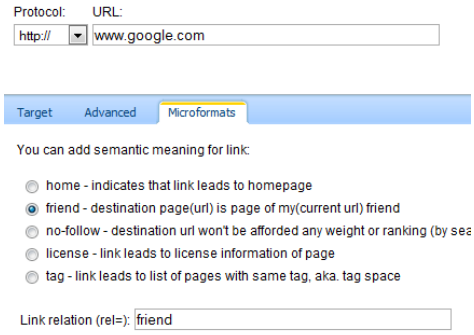


Obrázek 26: Globální nastavení mikroformátů

Pro `rel-license` mikroformát, jsem vytvořil webovou část, která vkládá odkaz na licenci. V této oblasti je běžné používat licence Creative Commons. Pro ty existuje několik online služeb na generování vkládaného kódu. Já jsem se ovšem nechtěl omezit pouze na tento druh licencí. Proto jsem vytvořil generátor vlastní, který umožňuje odkázat na libovolnou licenci. Nastavení je uloženo ve formátu JSON a umožňuje zpětnou editaci. Kentico nemá standardně možnost, aby si uživatelé mohli sami určit pod jakou licencí spadají jejich příspěvky. Vzhledem k tomu jsem se rozhodl vytvořit alespoň widget, který uživatelé budou schopni vložit do předpřipravených zón.

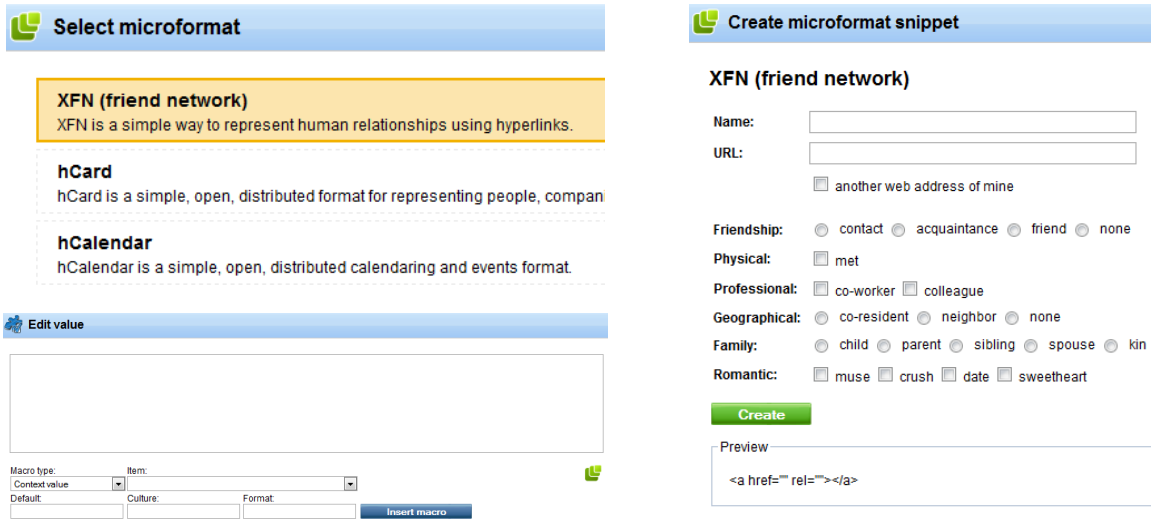
Obrázek 27: Ukázka z webové části pro generování licence

Podpora pro odkazové mikroformáty přibyla i do standardního dialogu pro editaci elementu odkazu. Je možné zde vložit obvyklé `rel` hodnoty. Jmenujme například `rel-nofollow`, kterým se dá označit odkaz, kterému nemají dávat vyhledávače žádnou váhu.



Obrázek 28: Rozšíření odkazovacího dialogu

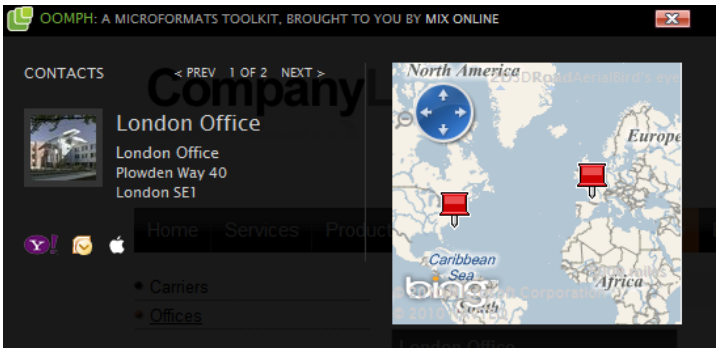
Kentico CMS podporuje deklaraci přátelství mezi uživateli. To je vše z mikroformátu `XFN`, co lze automaticky generovat. Upravil jsem transformaci, která je použitá na profilu uživatele. Tam se zobrazují odkazy na přátele a ty jsou nyní dekorovány atributerm `rel` s hodnotou `friend`. Aby však byla zachována možnost vypnutí mikroformátu `XFN` (a jiných) v transformacích, musel jsem upravit třídu `CMSAbstractTransformation`. Tato třída obsahuje veškeré pomocné metody, které je možné používat přímo uvnitř ASCX transformací. Přidal jsem obecné metody `MFEval`, `MFClassEval` a `MFWrapper`, které umožní generovat kód mikroformátu pouze pokud je zapnutý. V tranformaci pro XFN jsem použil `MFEval("rel=firend", "XFN")`. Uplné generování mikroformátu XFN lze dosáhnout vložením přes dialog (obr. 29). K tomuto dialogu se lze dostat přes FCK Editor (WYSIWYG editor pro „rich“ text) nebo přes kontrol `LargeTextArea` používaný například v nastavení webvých částí právě pro generování částí HTML kódu. Přes tento dialog je možné vložit mikroformáty `XFN`, `hCard` a `hCalendar`. Přidat podporu dalšího mikroformátu lze poměrně snadno. Mikroformát `hCard` byl zabudován do transformací pro dokumentový typ `Office` a také do profilu uživatele.



Obrázek 29: Ukázka dialogů pro vkládání mikroformátů

Jedním z posledních podporovaných je mikroformát `hProduct`. Ten slouží k anotaci produktů a je podporován například ve vyhledávání Google. Byly pro něj upraveny transformace zobrazující produkty na ukázkovém webu „Ecommerce site“.

Každou webovou část nebo celou zónu je možné obalit tzv. webovým kontejnerem. To umožňuje části/zóny vizuálně oddělovat. Velmi snadno se pomocí této funkcinality implementovala podpora pro neoficiální, ale užitečný mikroformát hSlice. Tento mikroformát vymyslel Microsoft v rámci vývoje prohlížeče Internet Explorer 8. Umožňuje označit blok stránky, který si uživatel „připne“ do svého prohlížeče a může k němu snadno a rychle přistupovat.



Obrázek 30: Webová část Oomph

Dále je pro mikroformáty k dispozici webová část „Oomph“. Název je odvozen od stejnojmenného balíku, který si klade za cíl usnadnit práci s mikroformáty. Jedna z jeho částí je javascriptová knihovna, která popřidaní na stránku hledá výskyt mikroformátů a pokud nějaké nalezne, dá o tom vědět zelenou ikonkou v levém horním rohu prohlížeče. Po kliknutí na ni se rozbalí pomocné okénko, ve kterém lze například rovnou spatřit adresu na mapě. Hezky to funguje právě na mikroformátech `hCard` použitých pro dokumenty typu „Office“.

Druhou částí balíku Oomph, je sada ASP server kontrolů pro tvorbu mikroformátů deklarativním způsobem přímo z Visual Studia. To ideálně vyhovuje pro použití v Kentico CMS při ASPX modelu vývoje. Z tohoto důvodu jsem tuto sadu zakomponoval do navržených rozšíření. Jak vypadá kód při použití těchto kontrolů je vidět zde:

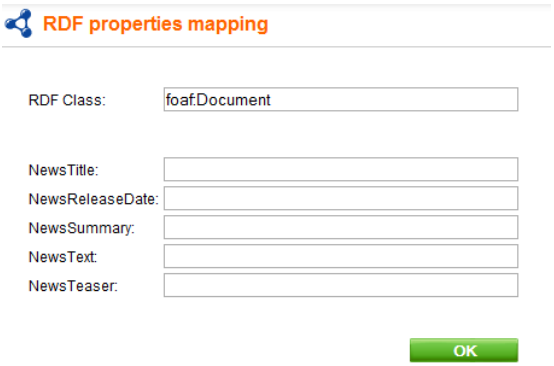
```
<mf:hCalendar runat="server" ID="hCalendar">
    <Properties>
        <mf:Summary Value="Event" />
        <mf:DateCollection>
            <Dates>
                <mf:StartDate DateTime="5/7/2010" />
            </Dates>
        </mf:DateCollection>
        <mf:Location Value="Brno" />
        <mf:Url Value="http://www.brno.cz" />
        <mf:Description Value="Nejaky popis." />
    </Properties>
</mf:hCalendar>
```

6.5 Podpora pro RDFa

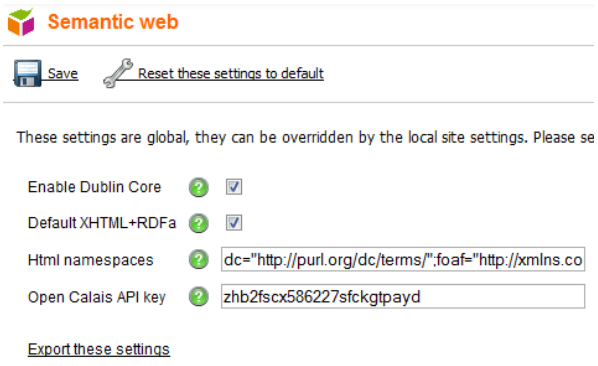
Výhodou RDFa (RDF zapisované přímo v HTML kódu stránky) je, že nepracuje s omezeným množstvím ontologií jako mikroformáty. Každý si tedy může vytvořit libovolnou vlastní ontologii nebo si může nějakou již existují upravit/rozšířit podle své potřeby. Zároveň pokud by tak činil ve velké míře, tak vytrácí aspekt „sdílení“ ontologie. Tyto možnosti sebou však přinášejí vyšší nároky na implementaci podpůrných rozšíření.

Pro RDFa se výborně hodí koncept vytváření vlastních dokumentových typů a jejich vlastností. Tato schopnost je již dlouhou dobu integrální součástí systému Kentico a do výchozího balíku se dostane i v nové verzí Drupalu 7. Cílem je tedy vytvořit podobné rozrhraní pro mapování vlastností dokumentového typu na vlastnosti RDF tříd i pro Kentico.

To sebou nese potřebu umět definovat jmenné prostory používané v systému. Toto lze nastavit v globálním nastavení. Definuje se zde řetězec, který obsahuje středníkem oddělené definice jmennýchprostorů, které se při zpracovávání do stránky vloží. Další položkou v nastavení je také určení DTD vygenerované stránky. Kentico standardně používá XHTML Transitional, pro použití RDFa je však nutné určit typ XHTML+RDFa. Již vytovořeným šabonám je nutné to nastavit zvášť.



Obrázek 31: Mapování RDF v Kentico CMS



Obrázek 32: Část globálního nastavení

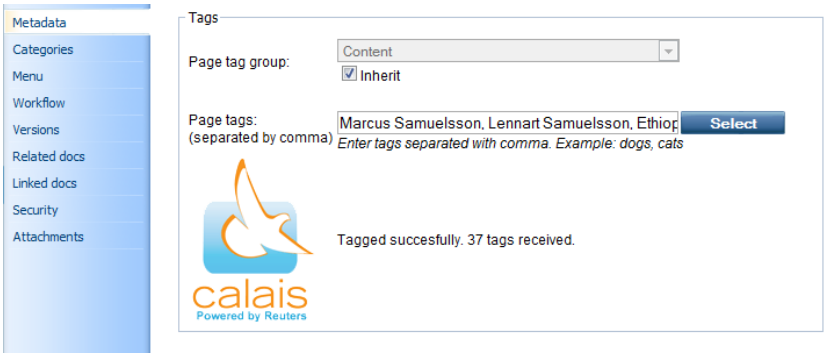
Použít vytvořené mapování je možné v transformacích přes speciální funkce v již zmíněné třídě `CMSAbstractTranformation`. Jako další součást byla plánovaná možnost anotování textu přímo z rozhraní FCK Editoru. Ta však zůstala značně omezena.

Celkově se využívá pro práci s RDF knihovny SemWeb. Ta je k dizpozici pro platformu .NET a umožňuje čtení i zápis RDF v notaci XML i N3 nebo také vytváří RDF úložiště s podporou pro dotazování pomocí jazyka SPARQL.

6.6 Dublin Core a Open Calais

Implementoval jsem do Kentico CMS ještě některé další sémantické prvky. Jedním z nich je podpora Dublin Core. Díky tomu je možné generovat základní informací o dokumentu jako je jméno, autor nebo datum vytvoření. Na stránku jsou tyto informace přidány v podobě meta elementů. Tuto vlastnost je také možné kontrolovat z globálního nastavení.

Dále jsem využil existence knihovny Calais.NET a vytvořil pomocí ní napojení na službu Open Calais. Je možné ji použí k automatickému otagování dokumentu na základě obsahu editovatelných webových částí.



Obrázek 33: Automatické tagování pomocí Open Calais

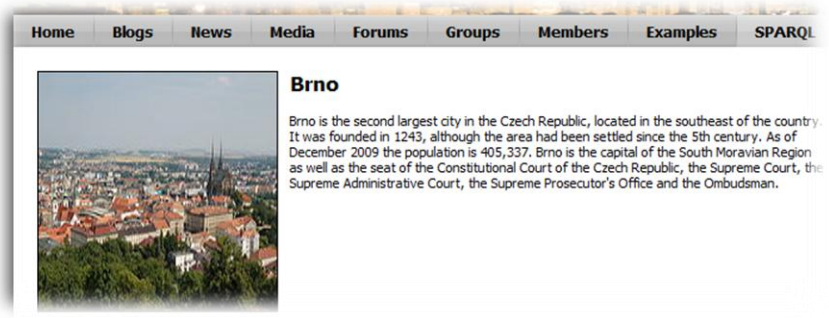
6.7 Získávání dat z externích zdrojů

Za externí zdroj můžeme považovat buď sémanticky anotovanou stránku nebo SPARQL endpoint. Stránky mohou být anotovány pomocí mikroformátů nebo RDFa. Oba způsoby jsou jiné, a proto je potřeba je rozlišit. Pro všechny tyto typy zdrojů jsem připravil webové části, pomocí kterých z nich lze získávat a zpracovávat data. Celkem jsou tedy k dispozici tři části – Optimus microformats transformer, GRDDL transformer a SPARQL data source.

První z nich slouží k transformaci mikroformátů na stránce do lépe zpracovatelné podoby – formátu XML, JSON nebo RSS. Jako samotné transformační jádro se používá předpřipravených XSL transformací ze stejnojmenného Open Source projektu. Pomocí parametrů transformace lze zvolit, které mikroformáty mají být ze stránky extrahovány. Podporované jsou `hcalendar`, `hcard`, `hentry`, `hresume`, `hreview`, `hlisting`, `xfolkentry`, `adr`, `geo`, `xfn`, `votelinks`, `rel-nofollow`, `rel-tag` a `rel-license`. Problémem z hlediska XSL transformací je, že mikroformáty mohou být zakomponovány i do HTML stránek, které nejsou validními XML dokumenty. Takovou stránku je tedy před zpracováním potřeba převést na validní XHTML, aby byla použitelná v transformaci. Samotný převod není triviální a byla proto použita knihovna HTML Tidy. Při přístupu na stránku, kde je umístěna tato webová část, se na výstup dostanou extrahovaná data ve zvoleném formátu.

Na podobném principu je postavena extrakce RDFa. Zde se ale využívá standardu GRDDL. Pokud webový vývojář nedefinuje v nastavení části „GRDDL Transformer“ explicitní použití konkrétní transformace, postupuje se tradičním způsobem. Na zdrojové stránce je vyhledán element `link`, který má atribut `rel` s hodnotou `transformation`. Ten odkazuje na XSL tranformaci. Jinými slovy, stránka sama si určí jak má být transformována. Standard GRDDL ovšem není tolik rozšířen, a proto jsem přidal možnost zvolit transformaci manuálně. Taková bude načtena z lokálního disku. Několik jsem jich rovnou přiložil k webové části, ale je možné si snadno přidat další.

Zřejmě nejzajímavějším způsobem získávání dat je pomocí jazyku SPARQL. Pro tento případ byla vytvořena část „SPARQL datasource“. Jedná se o standardní Kentico datasource část a tedy data z ní lze zobrazovat pomocí jiných webových částí jako „Repeater“ nebo „DataGrid“. Vzhledem k tomu, že jsou ovšem data ze SPARQL endpointu získávána v podobě XML, může být velmi vhodné data místo převodu do ASP DataSetu ponechat v XML dokumentu a následně zpracovávat pomocí XSLT. Proto jsem části přidat nastavení `Mode`, kterým se určí typ zdroje. Dalšími parametry je adresa SPARQL rozhraní a samotný dotaz. Jak může vypadat výsledek lze vidět na následujícím obrázku.



Obrázek 34: Ukázka zobrazení dat získaných přes SPARQL datasource

Využil jsem rozhraní DBpedia a napsal ukázkový dotaz pro nalezení všech míst, které mají určité jméno. Získá se jméno, obrázek a popis tohoto místa a zobrazí na stránku v podobě HTML. Uvnitř SPARQL dotazů lze krásně využít i makro výrazy systému Kentico. V tomto případě jsem využil výraz `{?place|Washington?}`. Ten zapříčiní, že část SPARQL dotazu bude nahrazena za parametr `place` získaný z URL stránky. Pokud takový parametr nebude existovat, použije se výchozí hodnota „Washington“. Zde bych chtěl zmínit problém, se kterým jsem se v rámci tvorby tohoto příkladu setkal. Samotné vymyšlení nějakého smysluplného dotazu není úplně jednoduché. Musí totiž splňovat několik vlastností. Musí pracovat nad relativně rozsáhlou doménou dat, jinak nejsou dotazy relevatní. Myšleno tak, že sice mohu položit dotaz např. na „luxusní auta“, ovšem v DBPedii je jich jen několik. Zdaleka ne všechny informace na WikiPedii jsou totiž dobře a úplně označeny. A na druhou stranu pokud máme instancí hodně, tak se zase snadno dostaneme do situace, kdy DBpedia nezvládne dotaz zpracovat.

6.8 Porovnání s ostatními CMS

Hlavním rozdílem vytvořených rozšíření oproti ostatním představeným CMS systémům je, že jde o kompletní řešení, nikoli o oddělené rozšiřující moduly. Tedy s výjimkou Drupalu verze 7. Ten bude mít podporu RDF a sémantiky zabudovanou přímo do svého jádra.V jeho případě se většinou jedná o integraci a vylepšení již existujících modulů. Velmi intenzivě se tomu věnuje široká skupina odborníků. Je proto jasné, že se jedná o systém se suverénně nejlepší podporou sémantiky.

Komunity okolo systému WordPress a Joomla mají o poznání menší snahu. Funkcionalita proto není kompletní a každý ze systémů umí trochu něco jiného. Rozšířené Kentico CMS oproti nim nemá například SPARQL rozhraní, ale má širokou podporu mikroformátů a umí mapovat vlastnosti dokumentových typů na RDF.

7 Závěr

V této práci jsem se zabýval Sémantickým webem. Představil jsem jeho vizi a problémy, které by mohl pomoci řešit. Podrobně jsem popsal různé jazyky a technologie, které jsou nezbytné pro jeho fungování. Shrnul jsem jeho současné možnosti využití a věnoval se také problémům, které ho provázejí. Jaká bude budoucnost je těžké předvídat, ale mým osobním názorem je, že na podobu, ve které byl představen v původní vizi, si budeme muset počkat ještě dlouho. Celá oblast je značně roztříštěná, což je mimo jiné způsobeno více pohledy na Sémantický web jako takový. Lze ho chápat jako „web čistých strukturovaných dat“ nebo jako na sémanticky obohacený obsah webu stávajícího. Stále je tedy na čem pracovat. Dosažené dílčí úspěchy se projeví postupným prorůstáním do různých oblastí informačních technologií.

Tento pokrok bude probíhat také na poli různých podpůrných systémů jako jsou i systémy pro správu webového obsahu. Ty by měly napomoci jak ke snadnější tvorbě sémanticky anotovaného obsahu, tak naopak k jeho zpětnému využití a zpracování. Velké odhodlání prokazuje především komunita kolem systému Drupal, který je jasnou jedničkou v této oblasti.

Přínosem této práce je implementace podobných sémantických rozšíření do systému Kentico CMS. Tohoto cíle bylo dosaženo a vytvořená rozšíření jsou popsána v šesté kapitole. Věřím, že alespoň některé z nich se podaří prosadit do produkční verze tohoto systému. Pro budoucí vývoj lze uvažovat integraci do jádra systému, poskytnutí více možností nastavení a užší provázání se všemi moduly.

8 Literatura

[1] **Berners-Lee, Tim.** Tim Berners-Lee on the next Web. *TED.* [Online] únor 2009. [Citace: 8. ledna 2010.] http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html.

[2] **Berers-Lee, Time.** The Semantic Web. *Scientific American.* [Online] květen 2001. [Citace: 18. ledna 2010.] http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

[3] **Wikipedia contributors.** History of the World Wide Web. *Wikipedia.* [Online] 30. března 2010. [Citace: 5. dubna 2010.] http://en.wikipedia.org/wiki/History_of_the_World_Wide_Web.

[4] **Berners-Lee, Tim.** Information Management: A Proposal. *W3C.* [Online] CERN, 1989. [Citace: 8. ledna 2010.] http://www.w3.org/History/1989/proposal.html.

[5] **Antoniou, Grigoris a van Harmelen, Frank.** *Semantic Web Primer.* místo neznámé : The MIT Press, 2008. ISBN 978-0-262-01242-3.

[6] **Universal McCann.** Power to the people - Social media tracker. [Online] červenec 2009. [Citace: 15. března 2010.] http://universalmccann.bitecp.com/wave4/.

[7] **O'Reilly, Tim.** What Is Web 2.0. *O'Reilly.* [Online] 30. září 2005. [Citace: 15. prosince 2009.] http://oreilly.com/web2/archive/what-is-web-20.html.

[8] **Spivack, Nova.** How the WebOS Evolves? [Online] Radar Networks, 09. únor 2007. [Citace: 18. března 2010.] http://novaspivack.typepad.com/nova_spivacks_weblog/2007/02/steps_towards_a.html.

[9] **Spivack, Nova**. Beyond Keyword (and Natural Language) Search. [Online] Radar Networks, 01. březen 2007. [Citace: 18. března 2010.] http://novaspivack.typepad.com/nova_spivacks_weblog/2007/03/beyond_keyword_.html.

[10] **Pollock, Jeff rey T.** *Semantic Web For Dummies.* Hoboken : Wiley Publishing, Inc., 2009. ISBN: 978-0-470-39679-7.

[11] **Ing. Vilém Sklenák, CSc.** *Metadata, sémantika a sémantický web.* [dokument] Praha VŠE, 2004.

[12] **Svátek, Vojtěch.** Ontologie a sémantický web. [Online] 2002. [Citace: 18. března 2010.] http://nb.vse.cz/~svatek/onto-www.pdf.

[13] **Wikipedia contributors.** Semantic Web Stack. *Wikipedie.* [Online] 20. ledna 2010. [Citace: 20. února 2010.] http://en.wikipedia.org/wiki/Semantic_Web_Stack.

[14] Extensible Markup Language (XML) 1.1. *W3C.* [Online] 29. září 2006. [Citace: 4. dubna 2010.] http://www.w3.org/TR/xml11/.

[15] **Kosek, Jiří.** Xml v příkladech – XML schémata. [Online] 2004. [Citace: 18. dubna 2010.] http://www.kosek.cz/xml/schema/index.html.

[16] **Kosek, Jiří.** Xml v příkladech – XPath. [Online] 2004. [Citace: 23. dubna 2010.] http://www.kosek.cz/xml/xslt/xpath.html.

[17] **Svátek, Vojtěch a Labský, Martin.** Objektové modely a znalostní ontologie. [Online] [Citace: 10. dubna 2010.] http://nb.vse.cz/~svatek/obj03fi.pdf.

[18] **Galgonek, Jakub.** *Dotazovací jazyky pro Sémantický web.* Praha : diplomová práce, MFF UK, 2008.

[19] **Allsopp, John.** About Microformats. *Microformats.* [Online] [Citace: 15. března 2010.] http://microformats.org/about.

[20] **Emily, Lewis.** Getting Semantic With Microformats, Part 1. *A Blog Not Limited.* [Online] 9. září 2008. [Citace: 25. dubna 2010.] http://www.ablognotlimited.com/articles/getting-semantic-with-microformats-part-1-rel/.

[21] **Çelik, Tantek, Mullenweg, Matthew a Meyer, Eric.** XFN 1.1 relationships meta data profile. *XFN 1.1.* [Online] [Citace: 25. dubna 2010.] XFN 1.1 relationships meta data profile.

[22] About Microformats. *Microformats.* [Online] [Citace: 15. března 2010.] http://microformats.org/about.

[23] HTML5 (including next generation additions still in development). *WhatWG.* [Online] 12. května 2010. [Citace: 13. května 2010.] http://www.whatwg.org/specs/web-apps/current-work/multipage/links.html#microdata.

[24] Top 5 Semantic Search Engines. *Pandia Search Engine News.* [Online] 16. února 2009. [Citace: 6. května 2010.] http://www.pandia.com/sew/1262-top-5-semantic-search-engines.html.

[25] The DBpedia Data Set. *DBpedia.* [Online] 5. května 2010. [Citace: 10. května 2010.] http://wiki.dbpedia.org/Datasets.

[26] **Wikipedia contributors.** FOAF (software). *Wikipedia.* [Online] 23. dubna 2010. [Citace: 27. dubna 2010.] http://en.wikipedia.org/wiki/FOAF_(software).

[27] Introducing FOAF. *FOAF Project.* [Online] [Citace: 27. dubna 2010.] http://www.foaf-project.org/original-intro#lGoals.

[28] About the Social Graph. *Google code.* [Online] Google. [Citace: 5. května 2010.] http://code.google.com/intl/cs/apis/socialgraph/docs/.

[29] Open Graph protocol. *Facebook.* [Online] duben 2010. [Citace: 22. dubna 2010.] http://developers.facebook.com/docs/opengraph.

[30] Linked Data - Connect Distributed Data across the Web. *LinkedData.* [Online] [Citace: 15. dubna 2010.] http://linkeddata.org/.

[31] **Wikipedia contributors.** Dublin Core. *Wikipedia.* [Online] 22. dubna 2010. [Citace: 25. dubna 2010.] http://en.wikipedia.org/wiki/Dublin_Core.

[32] What is a Content Management System or CMS? *Enterprise Content Management.* [Online] [Citace: 10. května 2010.] http://www.contentmanager.eu.com/cms.htm.

[33] Co je CMS / Redakční systém? *CMS Systémy.* [Online] [Citace: 10. května 2010.] http://www.cms-systemy.cz/o--portali/co-je-cms-redakcni-system--/.

[34] **Buytaert, Dries.** State of Drupal presentation (March 2008). [Online] 12. března 2008. [Citace: 15. května 2010.] http://buytaert.net/state-of-drupal-presentation-march-2008.

[35] Open Source CMS Market Share. [Online] 2009. http://www.cmswire.com/downloads/cms-market-share/.

[36] **Corlosquet, Stéphane.** RDF presentation at DrupalCon San Francisco 2010. [Online] 2010. [Citace: 18. května 2010.] http://www.slideshare.net/scorlosquet/rdf-presentation-at-drupalcon-san-francisco-2010.

[37] **Corlosquet, Stéphane.**. Produce and Consume Linked Data with Drupal! *OpenSpring*. [Online] 2009. [Citace: 12. května 2010.] http://openspring.net/.

[38] **Wikipedia contributors.** .NET Framework. *Wikipedia*. [Online] 11. května 2010. [Citace: 13. května 2010.] http://en.wikipedia.org/wiki/.NET_Framework.

Seznam příloh