Wirtschaftsuniversität Wien
Abteilung für Wirtschaftsinformatik
a.o. Univ.Prof. Dr. Rony G. Flatscher
Dr. Horst Treiblmaier

BAKKELAUREAT E-COMMERCE

Lehrveranstaltungsunterlagen zum Thema Semantic Web

Sommersemester 2005

Manfred Ahorn
✉ h0005522@wu-wien.ac.at

Inhaltsverzeichnis

1 Vorwort

Die vorliegende Bakkalaureatsarbeit wurde aufbauend auf der Seminararbeit „Lehrveranstaltungsunterlagen zum Thema Semantic Web“ von Manfred Ahorn, Thomas Altmutter, Lukas Helm und Dieter Steffen erstellt. Bei der Seminararbeit wurden die Kapiteln „Einleitung“ und „Semantic Web – Grundlagen“ von Lukas Helm, „XML“ und „Ausblick“ von Thomas Altmutter, „RDF“ von Dieter Steffen und „Ontologien“ von Manfred Ahorn mit den jeweils zu den Kapiteln passenden Übungsbeispielen und Lösungen erstellt. Im Rahmen der nun vorliegenden Arbeit wurden sämtliche Teilbereiche überarbeitet, ergänzt und durch zusätzliche Inhalte erweitert.

2Einleitung

2.1 Internet Heute

Das Internet ist heute ein riesiger, dezentraler Wissensspeicher, der Unmengen von Informationen bereitstellt. Die Metapher „Das Internet ist die größte Bibliothek der Welt, nur leider liegen alle Bücher verstreut auf dem Boden“ [Case] beschreibt die aktuelle Situation treffend. Demnach ist es momentan für Nutzer des Internets keine leichte Aufgabe, die benötigten Inhalte zu finden. Die einzige praktikable Möglichkeit, Daten aus dem Internet zu filtern, sind momentan Suchmaschinen. Sie können Websites nach textuellen Mustern durchsuchen, und über deren Metadaten grobe Informationen über die Seite erfahren, jedoch ist es ihnen nicht möglich, die Inhalte einer Seite zu verstehen. Das ist insofern problematisch, als sich diverse Inhalte über mehrere Websites verstreut befinden und diese Zusammenhänge von Suchmaschinen nicht aufgedeckt werden können [vgl. Palm01]. Es ist demnach wahrscheinlich, dass etliche nicht relevante Dokumente gefunden werden, andere relevante Dokumente (z.B. mit Synonymen) aber nicht gefunden werden können. Das Problem liegt darin, dass die Inhalte von Maschinen nicht verstanden oder interpretiert werden können. Diese Tatsache verhindert, dass Computer Verbindungen zwischen Websites herstellen können. Ein großes Potential bleibt ungenutzt.

2.2 Erweiterung des WWW

„The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users.” [BeHL01]

Die Idee des Semantic Web ist es, Webseiten mit Metadaten versehen, deren Semantik in Ontologien repräsentiert wird. Diese Idee stammt von Tim Berners-Lee, dem Vorsitzenden des **World Wide Web Consortium (W3C)**. Inhalte sollen maschinenlesbar gemacht werden. Metadaten sind Daten, die die eigentlichen Daten beschreiben. Beispielsweise könnte man über das Wort *Hund* sagen, dass es sich dabei um ein Lebewesen welches eine Säugetier und Haustier ist handelt. Die Mensch – Maschine Kommunikation soll so verbessert werden. Maschinen sollen befähigt werden, Zusammenhänge zwischen Sites zu verstehen und damit auch eine Suche über mehrere Sites hinweg zu

ermöglichen. Das herkömmliche WWW soll dadurch nicht ersetzt werden. Das Semantic Web stellt vielmehr eine Erweiterung dar. Komplexe Anfragen, die bisher nur von Menschen erledigt werden konnten, sind nun auch von Maschinen durchführbar. Die semantische Suche basiert nun nicht auf bestimmten Suchbegriffen, sondern auf der Bedeutung der Begriffe. Auch die Auswertung der gefundenen Daten kann dann durch den Computer erfolgen.

Als **Beispiel** betrachten wir nun eine Website, die Nachrichten anbietet. Die Inhalte dieser Site sind maschinenlesbar nach den Grundsätzen des Semantic Web verknüpft. Für die Kunden dieser Site, die wir fortan Newsmuster.org nennen wollen, bieten sich dadurch Möglichkeiten, die sie sonst nicht in Anspruch nehmen könnten.

Das Semantic Web soll sich allerdings nicht nur auf eine einzelne Seite beschränken. Gehen wir also davon aus, dass auch die mit „Newsmuster.org“ verknüpften Seiten den Ansprüchen des Semantic Web genügen.

Dadurch ergeben sich viele, mächtige Möglichkeiten, wie das Zusammenspiel zwischen Mensch und Maschine effizient gestaltet werden kann. In unserem Fall kann jemand, der sich für den Autor eines Artikels interessiert, sehr einfach weitere Nachforschungen anstellen. Innerhalb der Site kann nun leicht eruiert werden, welche weiteren Beiträge er verfasst hat, wann der letzte Beitrag eingestellt wurde oder wie viele Beiträge er bereits geschrieben hat. Darüber hinaus ist auch eine Interaktion mit der Site des Autors möglich. Ein Agent könnte automatisch weitere Informationen über den Autor ausfindig machen. Um in Erfahrung zu bringen, auf welcher Universität der Autor studiert hat, kann die Site des Autors den Agent auf die Homepage der Universität weiterleiten. Dort kann der Agent dann nach weiteren Studenten suchen, die im selben Jahr wie der Autor ihr Studium abgeschlossen haben. Das alles wird möglich, ohne dass der Mensch die Maschine dabei unterstützt.

Nehmen wir an, der Autor unseres Artikels heißt Herr Elch. Würde man in heute üblichen Suchmaschinen das Wort Elch als Suchbegriff eingeben, wüsste die Maschine nicht, ob man nach dem Tier oder einer Person sucht. Das Semantic Web macht es möglich, Begriffen eine Bedeutung zuzuordnen und so die Suche zu erleichtern. Man sieht, dass so Suchmechanismen im Internet stark verbessert und unterstützt werden können.

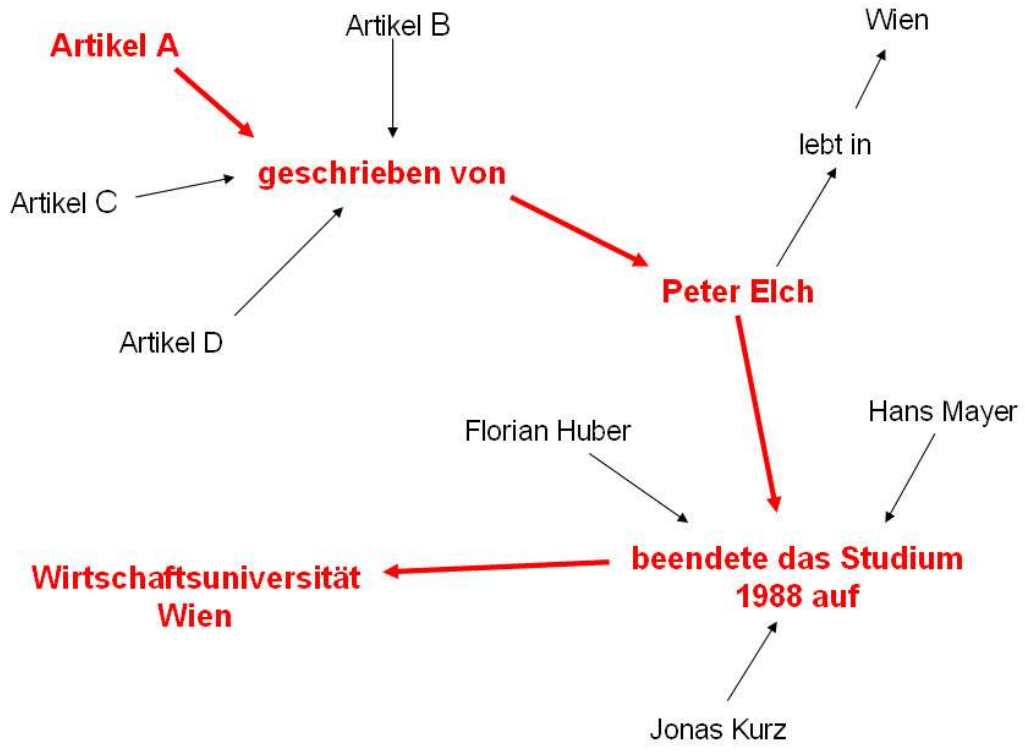


Abbildung 1: Funktionsweise des Semantic Web

Technologien zur Realisation dieser Idee existieren bereits, allerdings sind international anerkannte Standards notwendig. Sollten diese Standards nicht einheitlich umgesetzt werden, ist die Realisierung des Semantic Web nicht möglich.

2.2.1 Begriffserklärung Semantik

Zum besseren Verständnis des Semantic Web werden im Folgenden die Begriffe Syntax, Semantik und Pragmatik erklärt [Knop04]:

- **Syntax:** Im informatischen Sinn legt die Syntax eine formale Grammatik fest, welche die erlaubten Konstrukte einer formalen Sprache (z.B. Programmiersprache) festlegt. Es wird also festgelegt, welche Zeichenfolgen korrekt sind. Syntax ist mit der natürlichsprachlichen Grammatik vergleichbar.
- **Semantik:** Die Semantik bestimmt die Bedeutung von Aussagen. Die Aussage „Der Mensch ist eine Unterart des Hundes“ ist zwar syntaktisch korrekt, da die Zeichenabfolge erlaubt ist, aber semantisch ist sie offensichtlich falsch.

- **Pragmatik:** Die Pragmatik beschreibt die mit dem Auftreten von Zeichen verbundenen Folgen beziehungsweise die Handlungen. Beispielsweise könnte aus „Peter schimpft Paul“ folgen: „Paul mag Peter nicht“.

2.3 Semantic Web - Entwicklung

Die Idee des Semantic Web stammt von **Tim Berners-Lee**. Er gilt als eine der wichtigsten Persönlichkeiten bei der Entwicklung des WWW. Im Jahr 1990 konzipierte er das Hypertext Transfer Protocol (HTTP), das heute die Grundlage der Kommunikation im Internet darstellt. Er entwarf zusätzlich ein Schema zur Adressierung von Dokumenten im Internet, welches er Universal Resource Identifier (URI) nannte. Gegen Ende des Jahres hatte er dann den ersten Browser entwickelt, welchen er WorldWideWeb nannte, und den ersten Webserver installiert. Als das Internet wuchs fürchtete Berners-Lee, dass dessen offener Charakter gefährdet werden könnte. dem entgegen zu wirken gründete er das W3C. Das W3C ist ein Gremium zur Standardisierung und Weiterentwicklung des World Wide Webs und wird im Kapitel 2.3.1 W3C näher beschrieben. Entwickler von Browsern und Servern sollten die Möglichkeit haben mitzuwirken, wie das Web funktionieren soll. [vgl. O.A.a]

Die Idee des Semantic Web wurde entwickelt, um der immer größer werdenden Informationsflut im Web entgegen zu wirken. Im Jahr 1998 veröffentlichte Tim Berners-Lee die *Semantic Web Roadmap,* die erstmals einen Überblick über das Konzept des Semantic Web gab. Wichtig ist auch der 2001 im Scientific American erschienene Artikel *The Semantic Web* von Tim Berners Lee, James Hendler und Ora Lassila.

2.3.1 W3C

Das W3C wurde im Jahr 1994 von Tim Berners-Lee gegründet und hat die Zielsetzung, das World Wide Web zu seinem vollen Potential zu führen, indem Protokolle und Empfehlungen entwickelt werden, die eine langfristiges Wachstum des World Wide Webs sichern. W3C Mitglieder, W3C Mitarbeiter und externe Experten entwickeln gemeinsam Anforderungen und Spezifikationen[1]. Diese bilden den Rahmen für das Web von heute und morgen. Berners-Lee steht bis heute dem W3C vor.

[1] Unter einer Spezifikation versteht man einen formalen Text, der die Syntax und die Semantik (beispielsweise einer Software) beschreibt. Eine Spezifikation beschreibt also, was etwas ist oder was es tut.

400 Organisationen nehmen am W3C teil und bringen fast 700 technische Experten ein. So werden in 50 Working-, Interest- und Coordination-Groups Technologien und Standards entwickelt, geleitet von einem firmenneutralen Team des W3C. Technologien werden auf Implemetierbarkeit und Kompatibilität geprüft und bei positivem Ergebnis als Web-Standard des W3C anerkannt. Die Entwürfe werden außerdem durch die Mitglieder und die Öffentlichkeit geprüft, es bestehen Verbindungen zu 30 anderen Standardisierungsgremien. Zur Zeit gibt es 80 W3C Recommendations, dazu zählen HTML, XML, CSS, DOM, RDF, OWL, XSLT, P3P und viele weitere [W3C04].

2.4 Aufbau des Skriptums

Das Skriptum ist in die drei Teilbereiche: Einführung, Technologiendarstellung und Übungen untergliedert. Der erste Teilbereich sind die ersten zwei Kapitel „Einleitung“ und „ Semantic Web – Grundlagen“. Sie sollen einen Einstieg und einen kurzen Überblick über die behandelten Themen liefern. Im zweiten Teilbereich werden in den Kapiteln „ XML“, „ RDF“ und „ Ontologien“ jeweils am Beginn der Kapitel die Problemstellungen, die durch XML, RDF bzw. Ontologien behandelt werden dargestellt, danach werden XML, RDF und Ontologien detailliert beschrieben. Am Ende der Kapitel zeigen Praxisbeispiel Einsatzgebiete der jeweiligen Technologie. Das Kapitel „Ausblick“ stellt eine kurze Übersicht über zukünftige Entwicklungen des Semantic Webs dar. Durch den letzte Teilbereich „Übungen“ soll das erlernte Wissen an Beispielen angewandt und dadurch gefestigt werden.

3 Semantic Web – Grundlagen

Durch das Semantic Web sollen Daten im Internet maschinenlesbar gemacht werden, wodurch eine Art weltweite Datenbank entstehen soll. Um dieses Ziel zu erreichen, sind Standards unabdingbar. Im folgenden Abschnitt werden die Architektur des Semantic Web und die dazu benötigten Standards näher beschreiben.

3.1 Architektur

Tim Berners-Lee beschreibt den Aufbau des Semantic Web in mehreren Schichten (Abb.2 [KoMi01]). Diese Schichten sind Unicode/URI, XML, RDF, Ontologien, Logic, Proof und Trust.

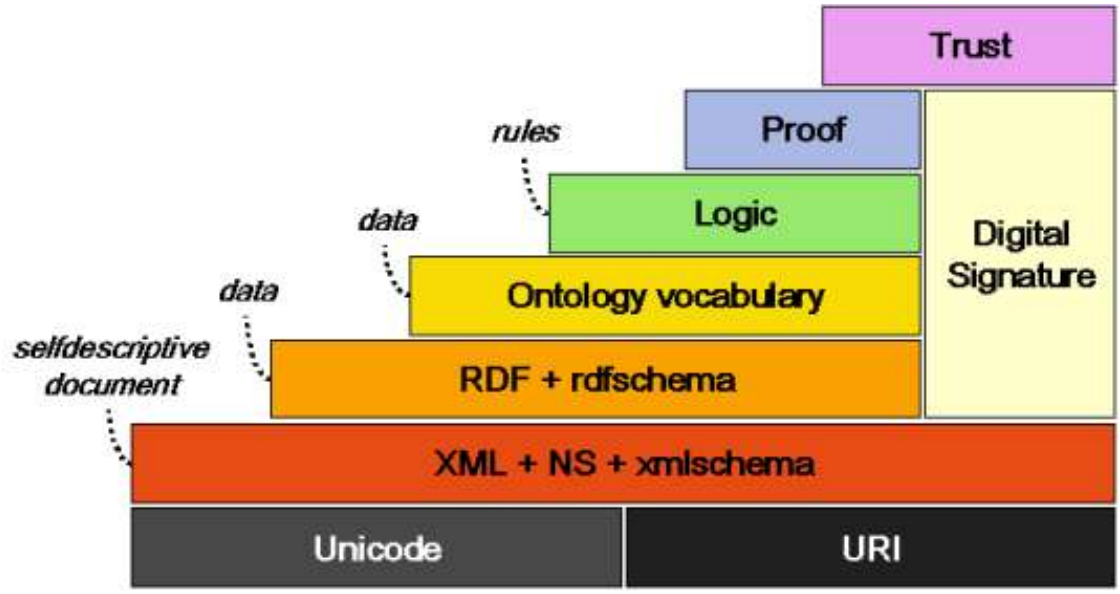


Abbildung 2: Schichtenmodell von Berners-Lee [KoMi01]

3.1.1 Unicode / URI

Unicode

„Der von der ISO genormte Zeichensatz Unicode (ISO 10646) ist ein 16-Bit-Code zur Darstellung von Schrift- und Steuerzeichen. Neben den Zeichen der westlichen und slawischen Sprachen werden durch diese Norm unter anderem Zeichen für Arabisch und Hebräisch, Griechisch, Kyrillisch und Armenisch, Indisch, Einheitszeichen aus dem Chinesischen, Koreanischen und Japanischen, mathematische, technische und grafische Symbole sowie spezielle Zeichen für Anwendungen definiert“ [HaNe01, S.1007].

Heute ist Unicode ein internationaler Standard, der von allen gebräuchlichen Betriebssystemen und Web-Datenformaten unterstützt wird. Eine inkompatible Codierung in unterschiedlichen Ländern soll damit verhindert werden.

Beispiele (in hexadezimaler Schreibweise) [Unic05]:

Zeichen	A	B	Z	a	=
Codierung	0041	0042	005A	0061	003D

Tabelle 1: Codierung in Unicode

URI (engl.: Uniform Resource Identifier)

“Ein URI […] ist entweder ein Verweis auf einen Ort, an dem ein bestimmtes Dokument gespeichert ist (engl.: uniform resource locator, abgekürzt: URL) oder ein symbolischer Name für eine prinzipiell beliebige Ressource (engl.: uniform resource name, abgekürzt: URN). Die meisten heute verwendeten URIs sind URLs.“ [HaNe01, S.1192f]

Mit Hilfe von URLs werden Ressourcen im Internet adressiert, die Verlinkung von Websites wird ermöglicht. URL Schemata sind beispielsweise http, https oder ftp.

Ein URL setzt sich folgendermaßen zusammen: An erster Stelle Steht das Protokoll, im Internet ist das häufig http, gefolgt von einem Doppelpunkt und zwei Schrägstrichen. Dahinter steht der Dienst, der angibt um welchen Inhalt es sich handelt. So steht www beispielsweise für World Wide Web, dem wichtigsten Internetdienst. Danach folgt der Hostname, der entweder als Domäne oder als IP-Adresse angegeben wird. Hier sind auch Subdomänen möglich. Darauf folgen die Verzeichnisangaben des angefragten Dokuments mit Pfad und Dateiname. Schlussendlich können mit Hilfe eines Fragezeichens als Trennzeichen noch Parameter übergeben werden (siehe Abb.3). Die Übergabe von Parametern dient beispielsweise der Speicherung von Daten, die in einem Formular eingegeben wurden [Wiki a].

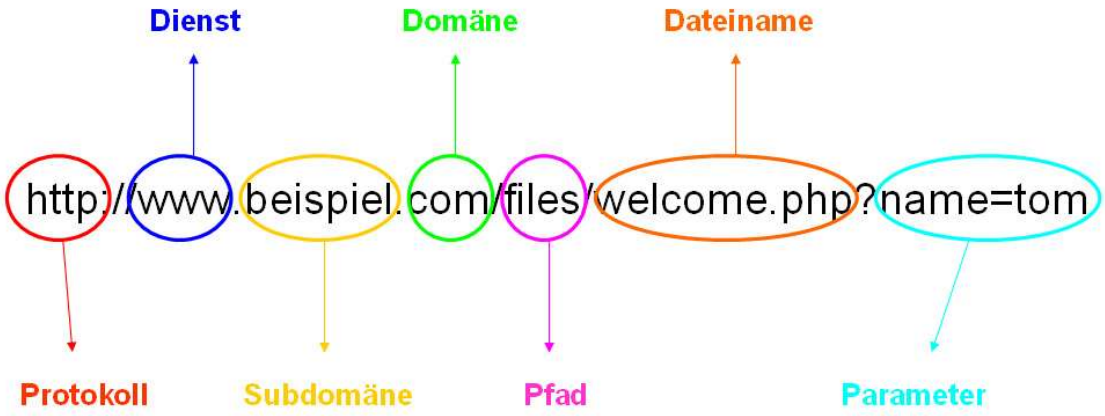


Abbildung 3: Aufbau einer URL

Ein URN ist ein dauerhafter, ortsunabhängiger Bezeichner für eine Ressource. Im Gegensatz zu URLs können sie nicht direkt aufgerufen werden, sondern müssen zuerst in URLs oder andere URIs übersetzt werden um angezeigt zu werden. URNs sind folgendermassen aufgebaut:

`urn:<NID>:<NID-spezifischer Teil>`

„`urn`" drückt aus, das es sich um eine URI nach dem Schema URN handelt. Der schemenspezifische Teil des URIs ist in Namespace Identifiers (NID) unterteilt [Wikiu]. Beispielhaft sei das Strategiekonzeptdokument für URNs der Deutschen Bibliotheken erwähnt. Dieses ist unter der URN „urn:nbn:de:1111-2003121811" gespeichert. Wobei „nbn:de" den spezifische Namesraum angibt und „1111-2003121811" das Dokument in dem Namesraum identifiziert. Um dieses Dokument in einem Browser darzustellen muss die URN zuerst mittels eines URN Resolvers in eine URL umgewandelt werden. Der URN Resolver hierfür lässt sich unter der URL „http://www.nbn-resolving.de/" aufrufen. Vollständigkeitshalber muss erwähnt werden, dass ein URN auch auf mehrere URLs verweisen kann.

Abbildung 4: URN Resolver

Der Vorteil von URN gegenüber URLs liegt in der eindeutigen Bezeichnung einer Ressource. Der Inhalt einer Newsseite, wie es unsere Beispielseite www.-newsmuster.org eine ist, wird sich täglich mehrmals ändern. Eine Verlinkung auf die Startseite http://www.newsmuster.org/index.html um einen spezifischen Inhalt zu erhalten macht daher keinen Sinn. Wird jedoch eine URN angegeben kann man sicher sein, den gewünschten Inhalt zu erhalten.

3.1.2 Extensible Markup Language (XML) / XML Schema

Eine einheitliche Syntax ist eine weitere wichtige Vorbedingung für das Funktionieren eines Systems wie des Semantic Web. Auf dieser Ebene kommt XML zum Einsatz. „XML […] ist eine Metasprache für die Definition von anwendungsspezifischen Auszeichnungssprachen." Eine Auszeichnungssprache ist eine Sprache, mit der Textelementen auf deklarative Weise Eigenschaften zugewiesen werden können. Dadurch kann deren Bedeutung (Semantik) ausgedrückt werden [HaNe01, S. 1043]. Maschinen- und menschenlesbare Dokumente können in Form einer Baumstruktur erstellt werden. Der Grundgedanke hinter XML ist es, den Inhalt vom Layout zu trennen [Wiki b].

Die Festlegung der Elemente und deren Stellung innerhalb des Baumes müssen von Fall zu Fall festgelegt werden. Im Fall des Semantic Web ist es allerdings sinnvoll, bereichsspezifische Konventionen einzuhalten, beziehungsweise sich bestehenden anzuschließen. Sollten gänzlich unterschiedliche Elementnamen benützen werden, wäre die Zusammenarbeit der Maschinen unmöglich.

XML-Schema bietet die Möglichkeit, die Struktur von XML-Dokumenten zu beschreiben. Im Gegensatz zu den Document Type Definitions (DTD) können mittels XML-Schema Datentypen[2] definiert werden. So kann nun festgelegt werden, wie eine XML Datei auszusehen hat und komplexere Strukturen als mit einer DTD können beschrieben werden. Diese genaue Beschreibung von Daten ist auch im Sinne des Semantic Web, da ein hoher Grad an Standardisierung erreicht werden soll.

Namensräume (engl.: namespaces) wurden in XML eingeführt, um der Mehrdeutigkeit von Element- und Attributnamen entgegenzuwirken. Um solche

[2] Ein Datentyp definiert, in welchem Wertebereich sich ein Element befinden darf. Beispiele: Zahlen (Integer), Text (String).

Konflikte zu vermeiden, wurde 1999 vom W3C eine Empfehlung zur Verwendung von Namespaces in XML (http://www.w3.org/TR/1999/REC-xml-names-19990114) gegeben [Hüsl05].

3.1.3 Resource Description Framework (RDF) / RDF Schema

RDF wurde für die Definition von Metadaten entworfen. Metadaten sind Daten über Daten, die Dateninhalte anhand eines bestimmten Vokabulars beschreiben. „Das RDF definiert ein allgemeines Metadatenformat für Information, die im Internet verfügbar ist.“ [HaNe01, S.1050] Jedes im Internet adressierbare Gut wird hier als Ressource bezeichnet. RDF bietet die Möglichkeit, Aussagen über Ressourcen zu treffen und deren Eigenschaften mit Hilfe eines vorgegebenen Vokabulars zu beschreiben. Dieses Vokabular heißt RDF Schema.
Durch Aussagen über Ressourcen werden diesen Eigenschaften zugeordnet. Eigenschaften besitzen wiederum Werte. Diese Werte können strukturierte Daten, Zeichenketten oder auch andere Ressourcen sein. Grafisch sieht das folgendermaßen aus:

Abbildung 5: Aufbau von RDF-Aussagen

Jede Aussage kann somit durch Subjekt, Prädikat und Objekt beschrieben werden, man spricht von RDF-Triples. So könnte man beispielsweise sagen, die Ressource www.google.com besitzt eine Eigenschaft Titel, und diese Eigenschaft hat den Wert Suchmaschine.

3.1.4 Ontologien

Im informatischen Kontext bedeutet Ontologie laut T. Gruber die „explizite formale Spezifikation einer gemeinsamen Konzeptualisierung“ [Grub95]. Eine einheitliche Definition hat sich bisher allerdings noch nicht durchgesetzt. Allgemein soll durch eine Ontologie ein Wissensbereich (z.B. das Internet) be-

schrieben werden. Die Beschreibung erfolgt mittels einer standardisierten Terminologie sowie Beziehungen und eventuell Ableitungsregeln zwischen den definierten Begriffen. Mit Hilfe dieser Terminologie können Entitäten beschrieben werden. Damit soll ein einheitliches Verständnis von Begriffen erreicht werden. Es können auch mehrere Ontologien bestehen, die unterschiedliche Terminologien verwenden. Es ist jedem möglich, eine Ontologie zu erstellen. Also kann es auch mehrere Ontologien zu einem einzigen Themenbereich geben, die sich dann auch unterscheiden können.

Newsmuster.org zählt das Impressum zu den Artikeln der Zeitschrift. Bei einer anderen Website umfasst der Begriff des Artikels nicht das Impressum (Abb. 5). Daraus können sich Probleme bei der Kommunikation ergeben, weil jeder etwas anderes unter dem Begriff Artikel versteht. Ontologien sollen helfen, diese Probleme zu lösen.

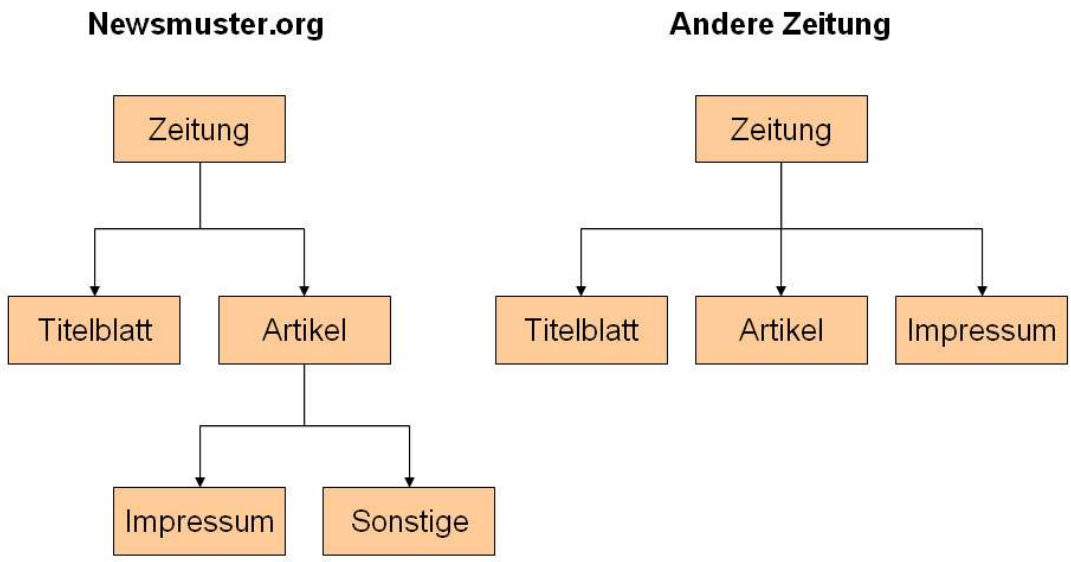


Abbildung 6: Unterschiede in Ontologien

3.1.4.1 Web Ontology Language (OWL)

OWL ist eine Spezifikation des W3C die es ermöglicht, Ontologien in einer formalen Beschreibungssprache zu erstellen und zu verteilen. Terme und deren Beziehungen zueinander sollen so beschrieben werden, dass auch Softwareagenten deren Bedeutung verstehen und sie weiterverarbeiten können.

Die Sprache basiert technisch auf der RDF-Syntax. Es ist auch möglich, Ausdrücke ähnlich einer Prädikatenlogik[3] zu formulieren. [vgl. W3Ca, Kap.5]

3.1.5 Logic

Die Stufe Logic des Semantic Web ist bisher technisch noch nicht realisiert worden. In Zukunft soll dadurch ermöglicht werden, logische Grundsätze aufzustellen aus denen ein Computer Schlüsse ziehen kann. [Choi03]

Unsere Website bietet auch einen Sportteil an. Logic ermöglicht hier einige interessante Schlüsse. So könnte man die Regel aufstellen, dass alle Fußballspieler, die in einer Saison mindestens 10 Tore geschossen haben, in die Liste der „Superscorer“ aufgenommen werden.

3.1.6Proof

Wenn man Systeme aufbaut, die auf Logik basieren, macht es auch Sinn, diese zur Überprüfung von Zusammenhängen oder Aussagen zu nutzen. So soll erklärt werden können, wie und warum man auf eine bestimmte Schlussfolgerung gekommen ist. Es ist einfach, einen veröffentlichten Proof zu überprüfen, schwieriger ist es allerdings, einen Proof zu erstellen. Proof alleine reicht allerdings nicht aus, um das Vertrauen der Benutzer in das Semantic Web zu gerechtfertigen. [Wels03]

Auch dieser Teil steckt noch in den Kinderschuhen und ist technologisch noch nicht verwirklicht worden.

3.1.7 Trust, Digitale Signatur

Im Internet stellt sich die Frage ob man das, was der Computer ausgibt, auch glauben kann, denn theoretisch kann jeder alles, was er will, zu einem Thema behaupten. Das behindert den Fortschritt des Semantic Web, denn ohne Vertrauen in die Korrektheit der Daten wird es kaum jemand benutzen. Es muss eine Möglichkeit gefunden werden, um den RDF-Statements vertrauen zu

[3] Mit Hilfe der Prädikatenlogik lassen sich Schlüsse formulieren. Es werden diverse Bedingungen (Prämissen) aufgestellt, daraus folgt dann ein Schluss (Konklusion). Beispiel: Prämisse 1: Alle Menschen sind sterblich. Prämisse 2: Sokrates ist ein Mensch. Konklusion: Sokrates ist sterblich. [vgl. Pann04]

können. Dabei ist es von besonderer Wichtigkeit, die Identität des Verfassers nachprüfen zu können.

Eine Möglichkeit, das zu prüfen, ist die digitale Signatur. Sie kann in RDF-Dokumente eingebunden werden und identifiziert mittels Verschlüsselungsverfahren eindeutig die Identität des Verfassers. Es kann nun entschieden werden, welchen Signaturen Vertrauen geschenkt wird.

3.1.7.1 Web of Trust

Da es aufgrund der Größe des WWW nicht möglich ist, alle Informationsquellen explizit zu kennen, wurde die Idee des Web of Trust entwickelt.

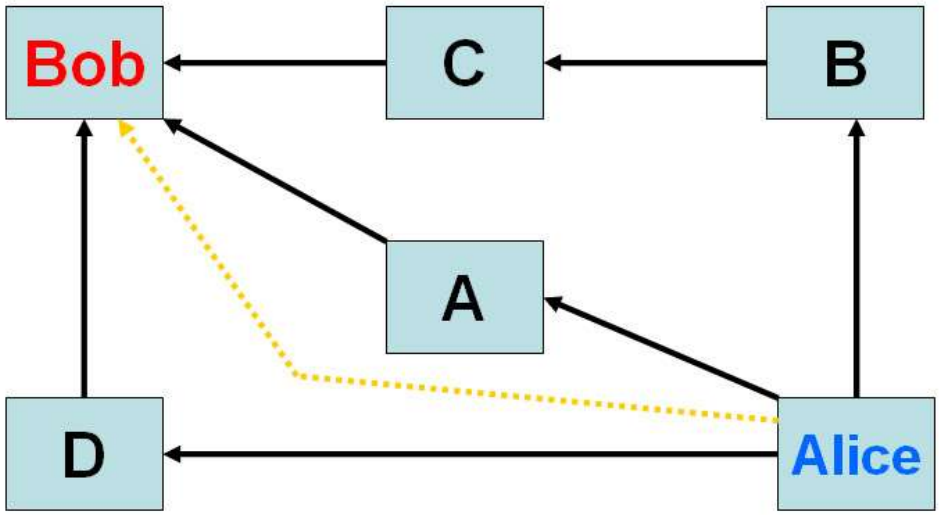


Abbildung 7: Web of Trust Funktionsweise

In Abb. 6 zeigen die durchgehenden Pfeile an, dass das Individuum dem nächsten vertraut. Alice möchte nun wissen, ob sie Bob vertrauen kann obwohl sie noch keine Beziehungen mit ihm hatte. Da einige Individuen ihres Vertrauens auch Bob vertrauen, kann sie davon ausgehen, dass auch er vertrauenswürdig ist.

3.2 Nutzen des Semantic Web

Nun stellt sich die Frage, welchen Nutzen die Technologien des Semantic Web bringen können. Besonders bei der Suche im WWW sind große Potentiale zur Verbesserung vorhanden. Wenn eine Suchmaschine Daten interpretieren und

ihnen eine Bedeutung zuordnen kann, werden weniger irrelevante Ergebnisse geliefert. Gegenüber einer rein textuellen Suche ergibt sich auch eine Beschleunigung, da dem Menschen die Aufgabe der Interpretation und Auswahl der Daten erleichtert oder abgenommen wird.

Voraussetzung hierfür ist allerdings eine weite Verbreitung. Das Semantic Web kann seinen potenziellen Nutzen nur dann entfalten, wenn die verknüpften Webseiten es auch einsetzen. Wird beispielsweise eine semantische Suche durchgeführt, werden jene Seiten, welche die neuen Technologien nicht implementiert haben, außer Acht gelassen werden.

Bei einer regional verbreiteten und konsequenten Umsetzung beschränkt sich der Nutzen aber nicht auf eine bessere Suche. Weitere Möglichkeiten reichen dabei von der automatischen Abstimmung von Zeitplänen bis zu einer regionalen Einschränkung von Geschäftspartnern. Transaktionen könnten viel leichter und schneller vonstatten gehen. Die semantische Verknüpfung von Daten im WWW bringt eine Effizienzsteigerung bei Operationen im Netz.

4 XML

4.1 Einleitung

XML ist eine immer weiter verbreitete Technologie, die in den unterschiedlichsten Bereichen Verwendung findet. Die Abkürzung steht für eXtensible Markup Language, also erweiterbare Auszeichnungssprache. XML ist somit keine *Programmier*sprache im herkömmlichen Sinn.

„Eine Auszeichnungssprache (Markup-Sprache, engl.: markup language) ist eine Sprache, die Regeln zur Auszeichnung von Textelementen bereitstellt. Beliebigen Textelementen können auf deklarative Weise Eigenschaften zugewiesen werden, wodurch deren Bedeutung (Semantik) ausgedrückt werden kann.“ [HaNe01, S.1043]

Markup-Sprachen, wie zum Beispiel HTML, kennzeichnen Teile eines Dokuments mit Metainformationen und ermöglichen es, eine Struktur zu erstellen. Metainformationen sind Informationen *über* etwas. Ein Beispiel für eine Auszeichnung (so wird die „Etikettierung“ bei Markup-Sprachen bezeichnet) wäre:

```
<title>WILLKOMMEN BEI LEARN@WU</title>
```

Das Besondere an XML ist, dass es eine strikte Trennung zwischen Daten und deren Verwendung gibt. Ein XML-Dokument lässt sich dadurch für die verschiedensten Zwecke einsetzen: von einer Überführung in HTML bis hin zur Verwendung in Datenbanken, bietet diese Aufteilung vielfältige Anwendungsmöglichkeiten. Außerdem lassen sich über geeignete Bezeichner Dokumente so unterteilen, dass die Struktur maschinell erkannt werden kann wodurch andere Programme sowohl darauf zugreifen als diese auch weiterverarbeiten können. XML eignet sich somit hervorragend, eine maschinell verwertbare Grundstruktur, die notwendig für die Idee des „Semantic Web“ ist, zu implementieren.

4.1.1 Überblick über hier behandelte Bereiche von XML:

- XML: eine erweiterbare Markup-Sprache; oft als Überbegriff (unter anderem für nachfolgende Technologien) verwendet.
- XML-Schema: in XML verfasste Definition (inklusive Einschränkungen) der verwendeten Tags.

- XSL: Abkürzung für Extended Style Sheet Language; in XML verfasste „Transformationsschablone“ mit der ein XML-Dokument in ein anderes Dokument umgewandelt werden kann.

4.2 Beschreibung XML

Genauso wie bei HTML gibt es auch bei XML so genannte „*Tags*“ (dt.: Etikett, Bezeichner, Aufkleber), welche die Elemente umschließen, auf die sie sich beziehen. Ein Beispiel:

```
<titel>Semantic Web</titel>
```

Diese Umschließung ist ein zwingender Bestandteil von XML. Ein geöffneter Tag *muss*, anders als bei manchen HTML-Befehlen, wieder geschlossen werden, wobei der Endtag dem Anfangstag mit einem vorangesetzten Slash („/“) entspricht:

```
<titel>…</titel>
```

Attribute werden in XML innerhalb des öffnenden Tags wie folgt deklariert:

```
<tagName attributName="attributWert">
```

Beispiel:

```
<titel autor="Mein Name">Titel</titel>
```

Es bleibt dem Verfasser des Dokuments überlassen, welche Informationen in eigenen Elementen und welche in Attributen gespeichert werden. Anstatt wie oben den Namen des Autors in einem Attribut zu speichern könnte man auch folgendes schreiben:

```
<titel>Titel
   <autor>Mein Name</autor>
</titel>
```

Es hängt somit sowohl von der persönlichen Vorliebe als auch von der weiteren Verwendung des Dokuments ab, welche Variante gewählt wird.

4.2.1 Wohlgeformtheit und Gültigkeit in XML

Wichtige Konzepte in Zusammenhang mit XML-Dokumenten sind die *Wohlgeformtheit* (engl.: well-formedness) und die *Gültigkeit* (engl.: validity) eines Dokumentes. Ein wohlgeformtes XML-Dokument erfüllt folgende Kriterien:

- Jeder Tag, der geöffnet wird, muss auch geschlossen werden. Leere Tags, d.h. Tags, die keinen Inhalt umschließen, aber für die Struktur wichtig sind, können anstatt ausgeschrieben zu werden (z.B.: `<titel></titel>`) folgendermaßen abgekürzt werden: `<titel/>`.
- Tags können nur im Zwiebelschalensystem verschachtelt werden
  `<Name><Vorname></Vorname></Name>`
  und NICHT `<Name><Vorname></Name></Vorname>`.
- Es darf nur EIN Wurzelelement (engl.: root-element) geben, das alle anderen Elemente einfasst.
- XML ist case sensitive, d.h. dass zwischen Groß- und Kleinbuchstaben unterschieden wird: <titel> ist ein anderer Tag als <Titel>. Diese Unterscheidung wird sowohl bei Elementen, als auch bei Attributen getroffen.
- Attributen zugeordnete Werte müssen in Anführungszeichen oder Hochkommas stehen.

Lösen Sie sich nun die Fehlersuchbeispiele Beispiel 1 und 2 im Kapitel 79: Übungsbeispiele an. Finden Sie die Fehler, indem Sie das bisher erwähnte anwenden.

Die Wohlgeformtheit eines Dokuments sagt lediglich aus, dass die allgemeinen XML-Regeln eingehalten wurden. Da es sich um eine *erweiterbare* Sprache handelt, sind die Tags bei XML nicht wie bei HTML vorgegeben, sondern frei definierbar. Diese Freiheit in der Definition von Tags führt dazu, dass irgendwie hinterlegt werden muss, welche Tags in welcher Form gültig sind. Desweiteren müssen XML Dokumente richtig strukturiert dargestellt werden. Um Elementen Datentypen, Beschränkungen und Ähnliches zuzuweisen, gibt es zwei Möglichkeiten von unterschiedlicher Mächtigkeit und Komplexität:

- Document Type Definitions (DTD)
- XML-Schema

Gegenüberstellung:

DTD	**XML-Schema**
Nicht in XML verfasst	In XML verfasst

Begrenzte vorgegebene Datentypen	Erstellung von eigenen komplexen Datentypen möglich; größere Auswahl an vorgegebenen Datentypen
Nicht erweiterbar, da Umfang per XML-Spezifikation 1.0 fest vorgeschrieben	Erweiterbar

Tabelle 2: Vergleich DTD und XML-Schema

Da DTDs einen weitaus eingeschränkteren Befehlsumfang als XML-Schema haben, werden sie hier nur der Vollständigkeit halber kurz vorgestellt.

Die XML-Datei	Die dazugehörende DTD
<?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE autorenliste SYSTEM "autoren-liste.dtd"> <autorenliste> <autor> <nr>1</nr> <vorname>Heinz</vorname> <nachname>Kauz</nachname> <alter>123</alter> </autor> </autorenliste>	<?xml version="1.0" encoding="ISO-8859-1"?> <!ELEMENT autorenliste (autor)> <!ELEMENT autor (nr, vorname, nachname, alter)> <!ELEMENT nr (#PCDATA)> <!ELEMENT vorname (#PCDATA)> <!ELEMENT nachname (#PCDATA)> <!ELEMENT alter (#PCDATA)>

Tabelle 3: XML-Datei und zugehörige DTD

Wie sich an obigem Beispiel erkennen lässt, ist eine DTD vergleichsweise kurz (insbesondere im Vergleich zu XML-Schema), aber wie bereits erwähnt, ist der Preis dafür eine eingeschränkte Funktionsvielfalt.

Um die Gültigkeit eines Dokuments zu überprüfen gibt es mehrere Wege: Zum einen gibt es Validatoren im Internet, von denen hier auf zwei verwiesen werden soll (Stand: Mai 2005):

- http://www.validome.org/
- http://www.w3.org/2001/03/webdata/xsv

Die zweite Möglichkeit ist ein Editor mit integriertem Validator. Der gängigste, der auch als für den privaten Gebrauch freie Home-Edition angeboten wird, nennt sich XML-Spy von der Firma Altova.

(http://www.altova.com/download_spy_home.html ; letzter Abruf am 2005-06-23)

4.3 XML – Der Anfang

Am besten lernt man XML jedoch, wenn man selbst Dokumente verfasst. Für den Anfang empfiehlt sich ein einfacher Text-editor. Der Autor empfiehlt, die ersten Beispiele handschriftlich oder mit einem Texteditor zu absolvieren bevor man zu einem Programm wie XML-Spy greift, da dadurch die Übung der Grundlagen verbessert wird.

Am folgenden einfachen Beispiel sollen die Grundelemente eines XML-Dokuments vorgestellt werden:

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <!DOCTYPE autorenliste SYSTEM "test01.dtd">
3 <autorenliste>
4     <autor>
5           <nr>1</nr>
6           <vorname>Heinz</vorname>
7           <nachname>Kauz</nachname>
8           <alter>123</alter>
9     </autor>
10    <autor>
11          <nr>2</nr>
12          <vorname>Martin</vorname>
13          <nachname>Muster</nachname>
14          <alter>26</alter>
15    </autor>
16    <autor>
17          <nr>3</nr>
18          <vorname>Gwendula</vorname>
19          <nachname>Bernfriedstein</nachname>
20          <alter>76</alter>
21    </autor>
22 </autorenliste>
```

Autorenliste.xml

Die Zahlen am Beginn der Zeile bezeichnen die Zeilennummer und dienen nur der besseren Beschreibung und sind *nicht* Bestandteil des Dokuments. Der Text, der von den frei wählbaren Ausdrücken in spitzen Klammern, um-

schlossen wird, stellt die eigentliche Information dar. Spitze Klammern und Schrägstriche kennzeichnen die Tags.

Die erste Zeile gibt an, dass es sich um ein XML-Dokument handelt und legt im „encoding="-Teil fest, welcher Zeichensatz benutzt wird. Zwingend erforderlich, um eine XML-Dokument als solches zu kennzeichnen ist:

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

Bitte erstellen Sie eine XML-Datei mit einem einfachen Texteditor → Beispiel 3 auf Seite 79.

In Zeile 2 folgt die Referenz auf eine DTD oder Schema-Datei, die für die Gültigkeitsüberprüfung des Dokuments notwendig ist und in der die selbst definierten Regeln für den Dokumentaufbau enthalten sind. In diesem Beispiel sieht man die Referenz auf eine DTD. Wird jedoch einen Schema-Datei verwendet *entfällt* Zeile zwei und Zeile drei wird folgendermaßen modifiziert:

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

Um nun nachvollziehen zu können, was in dieser Zeile passiert, muss man das Konzept der Namespaces kennen lernen.

4.3.1 Namespaces

Um auch XML-Vokabular (Tags) benutzen zu können, das von anderen bereits definiert wurden, werden „Namespaces“ (dt.: Namensraum) verwendet. Durch Namensräume ist es möglich gleichlautende Auszeichnungen zu differenzieren. Diese Unterscheidung wird folgendermaßen erreicht:

```
<root xmlns:test="http://www.meintest.at"
xmlns:versuch=" http://www.meinversuch.at">
     <unterelement>
          <test:name>Namespaces</test:name>
          <versuch:name>Namemsräume</versuch:name>
     </unterelement>
</root>
```

Beispiel für Namespaces

Bei der Angabe der Namensräume wird *keine* Verbindung mit dem Internet hergestellt. Es gibt lediglich, zum Beispiel bei Browsern, bereits integrierte Na-

mespace-Vokabulare, die Funktionen für den Browser zur Verfügung stellen. Das ist bei selbst definierten Namensräumen nicht der Fall. Dort kann man die angegebenen URLs vielmehr als ein eindeutiges Kennwort für die Namensraumabkürzung verstehen, die dadurch Namenskonflikte verhindern sollen.

„`xmlns:`“ Befehl zur Namensraumdeklaration

Mit Namespaces wurde auch im vorhergehenden Abschnitt zuerst der XML-Schema-Namensraum eingebunden, um auf die vordefinierten Elemente zugreifen zu können:

`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`

Danach wird durch `xsi:noNamespaceSchemaLocation="…"` festgelegt, dass sich alle in der betreffenden Schema-Datei vorkommenden Elemente der „Autorenliste“ im Vorgabennamensraum “`xmlns:xsi= "http://www.w3.org/2001/XML-Schema-instance"` befinden. Durch die Schema-Datei wird also kein eigener Namensraum definiert [Vgl. Jeck04, Kap. 2.3].

Die erweiterte Zeile drei noch einmal im Überblick:

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

4.3.2 XML-Schema

Der Nachfolger der Document Type Definition ist, wie bereits weiter oben erwähnt, ungleich mächtiger. Mit XML-Schema lassen sich eigene Elementtypen und Attribute kreieren. So ist es zum Beispiel möglich, festzulegen, ob ein Element nur ganzzahlige Werte oder ausschließlich Zeichenketten enthalten darf. Allein die Auswahl an vorgegebenen Datentypen, die sich wiederum kombinieren lassen, bietet eine Komplexität, die DTDs nicht aufweisen.

Das folgende Beispiel beschreibt eine Liste der Artikel für unsere Nachrichtenwebsite:

XML-Datei (artikel.xml):

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <artikelliste xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.newsmuster.org/artikellis
  te.xsd">
3   <artikel>
4     <titel>Die Fußballmeisterschaft ist vorbei.</titel>
5     <autor>2</autor>
```

```
6      <datum>2005-05-28</datum>
7      <rubrik>sport</rubrik>
8    </artikel>
9    <artikel>
10       <titel>Heißester Tag im Mai steht bevor!</titel>
11       <autor>3</autor>
12       <datum>2005-05-26</datum>
13       <rubrik>wetter</rubrik>
14     </artikel>
15   </artikelliste>
```

artikel.xml

Dazugehörige Schema-Datei (artikelliste.xsd):

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3    <xsd:element name="artikelliste">
4      <xsd:annotation>
5        <xsd:documentation>Kommentarmöglichkeit</xsd:documentation>
6      </xsd:annotation>
7      <xsd:complexType>
8        <xsd:sequence>
9          <xsd:element name="artikel" maxOccurs="unbounded">
10              <xsd:complexType>
11                <xsd:sequence>
12                     <xsd:element name="titel" type="xsd:string" />
13                     <xsd:element name="autor" type="xsd:int" />
14                     <xsd:element name="datum" type="xsd:date" />
15                     <xsd:element name="rubrik" type="xsd:string" /
>
16                </xsd:sequence>
17              </xsd:complexType>
18            </xsd:element>
19         </xsd:sequence>
20       </xsd:complexType>
21     </xsd:element>
22   </xsd:schema>
```

artikel.xsd

Die Funktion der Zeilen eins und zwei ist bereits beschrieben worden (Deklaration als XML-Dokument und Verwendung eines vorgegebenen Namensraumes). Auch die Struktur der nachfolgenden Zeilen entspricht, im Gegensatz zur DTD, der bekannten XML-Schachtelung. Neu ist, dass hier

vorgegebene Befehle des XML-Schema-Namensraumes benutzt werden. Die folgende Auflistung gibt einen Überblick über einige wichtige Schema-Befehle:

Befehl	**Beschreibung**
“element” +Attribut “name”	Definitionsbeginn eines Elements Bsp.: <xsd:element name=”katalog”>
“type”	Typeneinschränkung Bsp: <xsd:element name=”zahl” type=“integer“ />
“simpleType”	Definiert, dass das übergeordnete Element nur Daten, aber keine anderen Elemente aufnehmen kann
“complexType”	Notwendig, wenn ein Element andere Elemente enthalten soll oder der gewünschte Datentyp nicht bei den vordefinierten enthalten ist
“sequence”	Gibt an, dass folgende Elemente genau in dieser Reihenfolge auftreten müssen
„choice“	Gibt an, dass von den folgenden Unterelementen der Schema-datei nur eines in der XML-Datei vorkommen darf

“restriction”	Leitet eine Beschränkung ein:
	Bsp.: `<xsd:simpleType>`
	`<xsd:restriction base=”xsd:integer”>`
	`<xsd:minInclusive value=“1“/>`
	`</xsd:restriction>`
	`</xsd:simpleType`

“minOccurs” bzw. “maxOccurs”	Attribute, die angeben, wie oft ein Element mindestens vorkommen muss oder maximal vorkommen darf Bsp.: `<xsd:sequence maxOccurs=“unbounded“>`

Tabelle 4: XML-Schema: einige Elemente und Attribute

Bitte bearbeiten Sie im Übungsteil Beispiel 4 auf Seite 79 und Beispiel 5 auf Seite 80.

4.3.3 XSL

Ein XML-Dokument kann mittels XSL in andere Dokument(typen) transformiert werden, die unterschiedlichen Anforderungen gerecht werden (z.B. XHTML, HTML, Text,…). Abhängig von der Verwendung werden sie dementsprechend angewendet. Um alle Möglichkeiten, die XSL-Transformationen bieten, zu beschreiben, reicht der Platz in dieser Arbeit nicht aus. Es soll nur eine „erste Bekanntschaft“ geschlossen werden, um einen Eindruck davon zu bekommen, was man mit XSL umsetzen kann.

Bevor jedoch XSL vorgestellt werden kann, benötigt man eine Sprache, die auf einzelne Elemente eines Dokuments zugreifen kann. XPath bietet diese Möglichkeit.

Mit XPath navigiert man sozusagen durch das XML-Dokument. Dadurch ist es auch möglich, einzelne Elemente herauszugreifen und nur diese mit XSL (oder anderweitig) weiterzuverarbeiten.

/	Wurzelelement; Trennzeichen Bsp.: /autor/name
//	Suche im gesamten Dokument; Beispiel: „//titel“ ergibt als „Suchergebnis „jedes „titel“-Element, unabhängig vom Platz im Dokument
.	Aktuelles Element
@	Markiert ein Attribut, dass dem Suchkriterium entspricht

Tabelle 5: Wichtige Pfadangaben in XPath [vgl. JLI05, S. 216]

Eine Visualisierungshilfe mit dem Namen XPath-Demo, die sich gut zum experimentieren eignet, findet sich im WWW unter http://www.futurelab.ch/xmlkurs/xpath.de.html (letzter Aufruf: 2005-06-21).

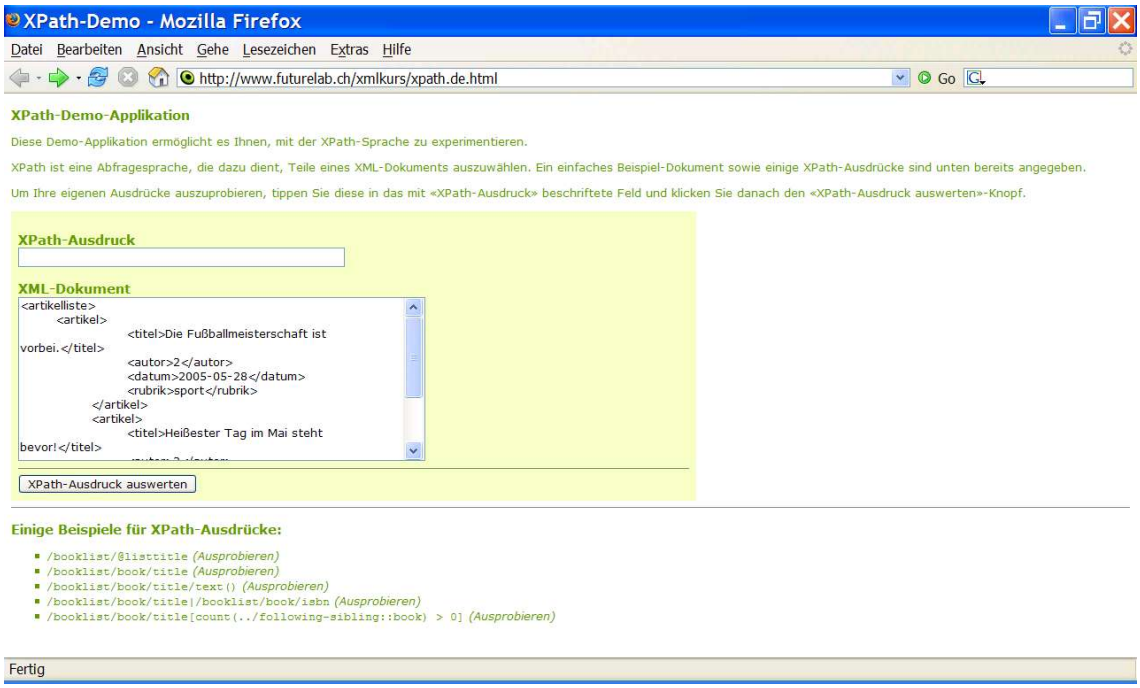


Abbildung 8: Screenshot von XPath-Demo – Startseite

Wenn wir nun unser XML - Beispiel „artikel.xml“ aus dem Kapitel 3.3.2 in XPath - Demo eingeben, können daran XPath Ausdrücke einfach „ausprobiert“ werden. Die Teile des XML-Dokuments, welche vom Ausdruck ausgewählt werden, stellt XPath - Demo rot markiert dar.

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <artikelliste>
3   <artikel>
4     <titel>Die Fußballmeisterschaft ist vorbei.</titel>
5     <autor>2</autor>
6     <datum>2005-05-28</datum>
7     <rubrik>sport</rubrik>
8   </artikel>
9   <artikel>
10      <titel>Heißester Tag im Mai steht bevor!</titel>
11      <autor>3</autor>
12      <datum>2005-05-26</datum>
13      <rubrik>wetter</rubrik>
14    </artikel>
15  </artikelliste>
```

artikel.xml

Beispielhaft kann nun der XPath Ausdruck „/artikelliste/artikel/titel“ eingeben werden. Als Ergebnis erhält man rot markiert „<titel> Die Fußballmeisterschaft ist vorbei. </titel>“ und „<titel> Heißester Tag im Mai steht bevor! </titel>“.

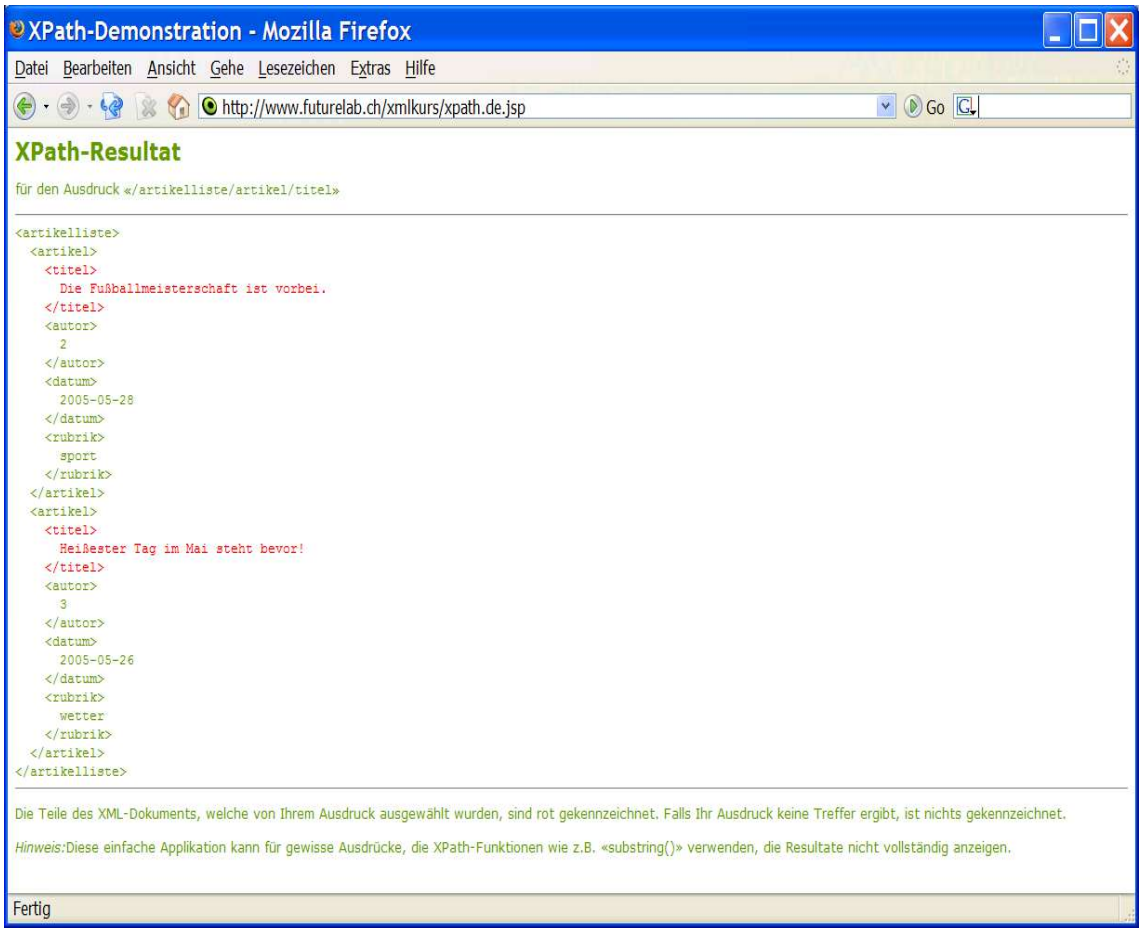


Abbildung 9: Screenshot von Xpath-Demo - Auswertungsseite

Mit dem Wissen um XPath kann man nun eine Transformation von XML in bspw. HTML vornehmen. Als Grundlage dient wiederum die Datei artikel.xml aus dem Kapitel 3.3.2. Hier ist die zweite Zeile hervorzuheben, welche die Referenz auf den XSL-Stylesheet darstellt:

```
   <?xml-stylesheet type=”text/xsl”
href=”http://www.newsmuster.org/artikel.xsl”?>
```

Bevor die XSL-Datei erstellt wird, sollte man sich überlegen, wie die HTML-Ausgabe aussehen soll bzw. welche Elemente der XML-Datei vorkommen sollen. Nehmen wir an, wir wollen eine Übersicht aller Artikel, die aus deren Titel, Datum und Rubrik besteht, in einer Tabelle ausgeben. Der Kopf der XSL-Datei ist folgendermaßen aufgebaut:

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <xsl:stylesheet xmlns: xsl=http://www.w3.org/1999/XSL/Transform
  version="1.0">
3 <xsl:output method="html"/>
4   <xsl:template match="/">
```

Nun folgen die HTML-Anweisungen:

```
5     <html>
6       <head><title>Transformation</title></head>
7       <body>
8         <table border="1">
9           <tr>
10               <td>Titel</td>
11               <td>Datum</td>
12               <td>Rubrik</td>
13             </tr>
```

Bis hier findet sich reines HTML. Ab der nächsten Zeile finden sich auch XSL-Befehle, die die gewünschte Information aus dem XML-Dokument „markieren“ und deren Wert zwischen die HTML-Tags setzen.

```
14             <xsl:for-each select="artikelliste/artikel">
15               <tr>
16                 <td><xsl:value-of select="titel"/></td>
17                 <td><xsl:value-of select="datum"/></td>
18                 <td><xsl:value-of select="rubrik"/></td>
19               </tr>
20             </xsl:for-each>
21           </table>
22         </body>
23       </html>
24     </xsl:template>
25 </xsl:stylesheet>
```

Mittels xsl:for-each wird eine for-Schleife eingeleitet, die die Anweisungen innerhalb der Schleife für jeden Eintrag „`artikel`“ anwendet. xsl:value-of nimmt

den Inhalt des durch select gekennzeichneten Elements („`titel`“, „`datum`“, „`rubrik`“) und setzt ihn in die Tabellenfelder ein.

Nach erfolgter Transformation sieht das Ergebnis so aus:

Titel	Datum	Rubrik
Die Fußballmeisterschaft ist vorbei.	2005-05-28	sport
Heißester Tag im Mai steht bevor!	2005-05-26	wetter

Abbildung 10: Ergebnis der XSL-Transformation

Lösen Sie nun die Übungen 6 und 7 im Übungsteil auf Seite 81 und 81.

4.4 XML in der Praxis

XML ist DER Standard um Daten strukturiert zu speichern. Auch viele „neue“ Standards wie ATOM[4], RSS oder eben auch das Konzept des Semantic Webs bauen auf XML auf. Weitere Indinzen für die enorme Verbreitung und Verwendung von XML sind die Vielzahl der erschienenen XML Publikation und XML Hilfstools. Selbst in der Automobilbranche wird XML verwendet. Die Entwicklungsabteilungen der Automobilkonzerne nützen XML als Standard , da die immer komplexer werdende Entwicklungsprozesse der Bordnetzelektronik nicht proprietäre Beschreibungssprachen und Applikationen benötigen [Weim05].

[4] Atom ist ein auf XML basierender Standard der den plattformunabhänigen Ausstausch von Informationen im Internet ermöglicht.

5 RDF

5.1 Einleitung

Das Resource Description Framework (RDF) ist ein Modell zur Repräsentation von Metadaten. Metadaten sind Daten, die Informationen über andere Daten enthalten. Beispielsweise sind Titel, ISBN-Nummer und Autorenangaben Metadaten eines Buches.

RDF besteht aus einem grafischen Modell zur Repräsentation von Metadaten und aus einer XML-Syntax, die dem gleichen Zweck dient. Das Grundmodell hierfür baut auf der Graphentheorie auf, die in 5.11 Exkurs: Graphentheorie näher beschrieben wird. Verglichen zu einfachen XML-Instanzen haben die RDF-Instanzen zusätzliche Vorgaben zu erfüllen. Diese vorgegebene Struktur dient als Rahmen für verschiedenste Metadatenformate, die als DTD, mit XML-Schema oder auch mit RDF-Schema spezifiziert werden können. Die Verwendung des RDF-Schemas erlaubt die Bildung von Begriffshierarchien (Ontologien) für die semantische Einordnung von Begriffen. Diese können sowohl auf die einzelnen Elemente der Metadatenformate, als auch auf deren Inhalte angewendet werden [vgl. Ecks04, S. 235].

Man kann die Suche nach Informationen im Internet wesentlich verbessern, wenn man nicht nur auf syntaktischer Ebene (Namen, Bezeichnungen, …) mit Suchmaschinen arbeitet, sondern auch mit Hilfe von Beschreibungen die Webquellen semantisch einordnet, sowie die Begriffe mit Hilfe von Ontologien zueinander in Beziehung setzt. Die technische Umsetzung soll mittels RDF gestaltet werden. [vgl. Ecks04, S. 236]

Es ist zu erwarten, dass in naher Zukunft die Suchmaschinen auch semantische Informationen verarbeiten können. Da in den Begriffshierarchien z.B. auch Verwandtschaftsbeziehungen beschrieben werden, können dann auch Seiten gefunden werden, die nicht genau den gesuchten, sondern nur einen verwandten Inhalt haben.

5.2 RDF-Modell

Das Basisdatenmodell von RDF ist einfach aufgebaut und dient der Beschreibung beliebiger Ressourcen, die durch RDF-Ausdrücke beschrieben werden und eine eindeutige URI haben. Das sind z.B. einzelne Webseiten, Sammlungen von Webseiten, aber auch Objekte wie beispielsweise Bücher, Personen, etc., auf die nicht direkt über das Web zugegriffen werden können.

Man unterscheidet folgende drei Grundbausteine bzw. Objekttypen:

- Ressourcen (engl. „Resources“)
- Eigenschaften (engl. „Properties“)
- Aussagen (engl. „Statements“)

Eine Ressource wird durch ihre Eigenschaften wie z.B. Datum, Autor, Bezeichnung, usw. beschrieben. Die Eigenschaften werden als Kanten dargestellt, welche die beiden Knoten für die Ressource und den Wert der Eigenschaft verbindet.

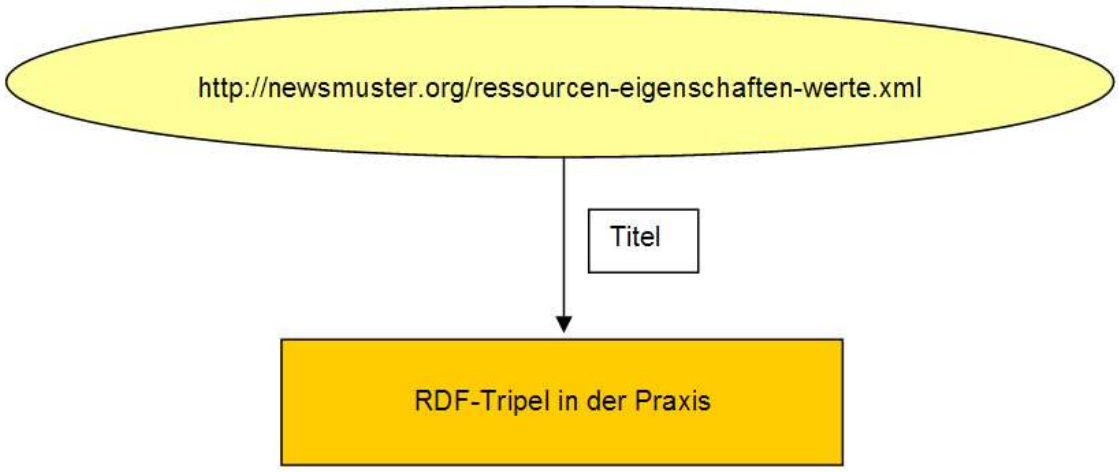


Abbildung 11: Ressource, Eigenschaft und Wert der Eigenschaft

Die URI „*http://newsmuster.org/ressourcen-eigenschaften-werte.xml*“ ist der Verweis auf die zu beschreibende Ressource, „*Titel*“ ist die Eigenschaft und „*RDF-Tripel in der Praxis*“ ist der Wert der Eigenschaft. Ein Wert einer Eigenschaft kann noch weiter unterteilt werden, indem man anstelle des Rechtecks ein leeres Oval zur Darstellung verwendet. Dieses symbolisiert eine unbenannte Ressource der die einzelnen (unterteilten) Werte als Rechtecke über benannte Kanten zugeordnet werden.

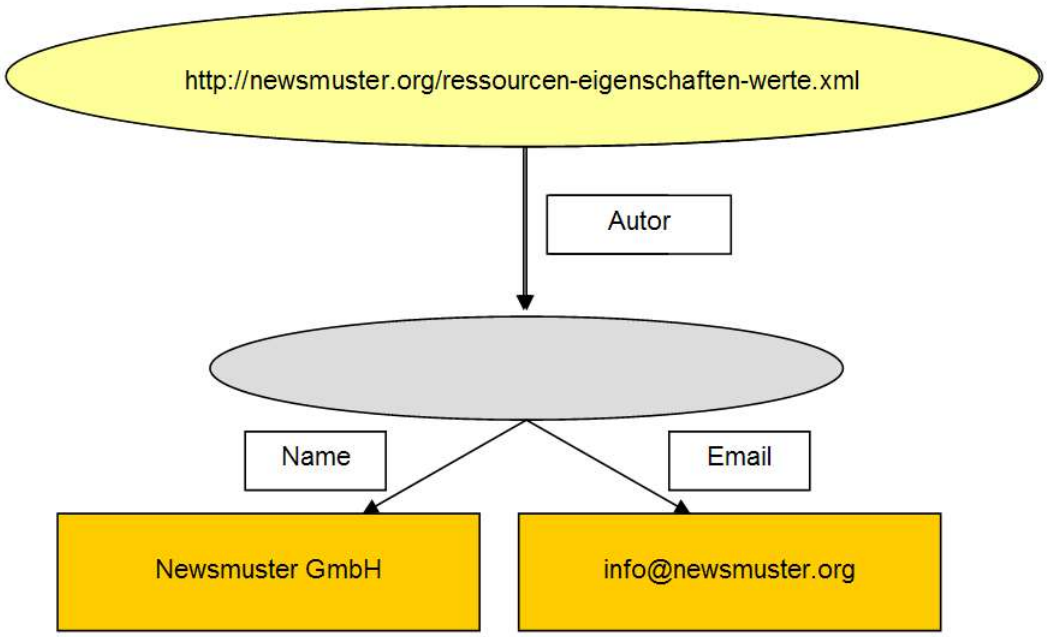


Abbildung 12: Wert der Eigenschaft ist eine Ressource mit zwei eigenen Eigenschaften und Werten

Die Eigenschaft *Autor* wird ihrerseits als Ressource mit den zwei Eigenschaften *Name* und *Email* und deren Werten beschrieben.

So genannte Aussagen werden in RDF als Sätze in Form von Subjekt, Prädikat und Objekt interpretiert, wobei die Ressource als Subjekt, die Eigenschaft als Prädikat und die Werte der Eigenschaften als Objekt zu verstehen sind. Folgender Satz ist aus der vorigen Abbildung zu lesen: „Das XML-Dokument hat einen Verfasser, der durch den Namen und die Email beschrieben wird.“ Die nächste Abbildung ist zu lesen als: „Die Website www.newsmuster.org hat den Ersteller (mit dem Wert) Seminargruppe.“

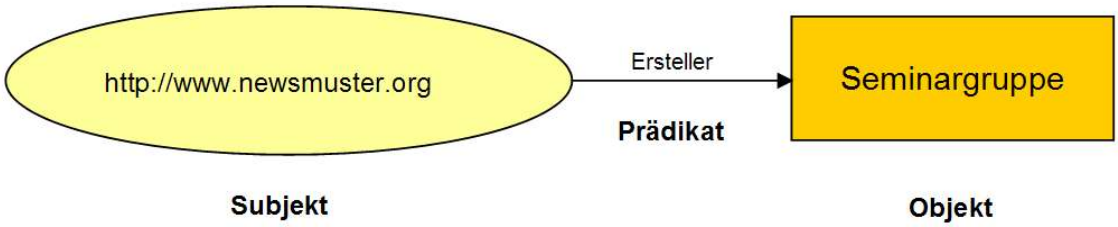


Abbildung 13: Aussage in Form eines einfachen RDF-Tripels aus Subjekt, Prädikat und Objekt

Wie man an diesem Beispiel sehen kann, zeigen die (gerichteten) Kanten immer vom Subjekt ausgehend auf das Objekt der Aussage.

Lösen Sie die RDF Beispiele 1 und 2 im Übungsbeispielteil auf Seite 82 und 82.

5.3 RDF-Syntax

Das RDF-Modell bietet eine abstrakte Vorlage zum Definieren und Benutzen von Metadaten. Für die praktische Erstellung und den Austausch von Metadaten ist allerdings eine konkrete Syntax erforderlich.

Für die Darstellung der Syntax der RDF-Datenmodelle wird das vom W3C standardisierte XML verwendet, das durch seine Darstellung in reiner Textform leicht erstellbar, durchsuchbar und plattformunabhängig austauschbar ist. RDF verwendet die XML-Namensräume, womit eine eindeutige Definition von Elementen und die Zuordnung von Elementen zu einem bestimmten Schema erreicht werden. Diese Zugehörigkeit zu einem Schema wird durch ein zuvor definiertes Präfix vor dem Element festgelegt.

```
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

Wurzelelement:

```
<rdf:RDF>  … Beschreibung …  </rdf:RDF>
```

***about*- und *ID*-Attribut, Description:**

Die zu beschreibende Ressource wird mit einem von zunächst zwei Attributen des Description-Elementes referenziert: *about* und *ID*. Die *ID* wird verwendet, wenn die Ressource keinen URI besitzt. Das ist dann der Fall, wenn man mit dem RDF-Schema Begriffe definieren und in eine Begriffshierarchie einer Anwendungsdomäne einordnen will. Dann nimmt man den Namen des Begriffs als Wert des *ID*-Attributes. Diese *ID* ergibt dann zusammen mit dem URI des definierten Dokuments den URI der Ressource.

```
<rdf:Description ID="Begriff">
  <xyz:Name>Name</xyz:Name>
```

```
</rdf:Description>
```

Beschreibt man Ressourcen, die über einen URI identifizierbar sind, wird das *about*-Attribut verwendet und man setzt als Wert den URI. Damit kann man Begriffe, die schon an anderer Stelle beschrieben wurden, wieder aufnehmen und bei Bedarf auch um weitere Beschreibungen ergänzen. Der Inhalt des Description-Elementes besteht aus beliebig vielen Eigenschaften.

```
<rdf:Description
      [about-Attribut oder ID-Attribut]
>
      [Eigenschaftselement]
      [Eigenschaftselement]
            …
</rdf:Description>
```

Rahmen eines RDF-Dokumentes:

```
<?xml version="1.0"?>
  <rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xyz="http://www.newsmuster.org/unserschema/">

     [ ... ]

   </rdf:RDF>
```

Das Schema „xyz“ ist dabei ein von www.newsmuster.org selbst festgelegtes Schema.

Die **Beschreibung der Ressourcen** mit ihren Eigenschaften werden innerhalb der Description-Tags vorgenommen:

```
<rdf:Description about="www.newsmuster.org/redaktion">
     <xyz:Creator>Max Mustermann</xyz:Creator>
<rdf:Description>
```

Mehrere Eigenschaften, die sich auf dieselbe Ressource beziehen:

```
<rdf:Description about="www.newsmuster.org/redaktion">
     <xyz:Creator>Max Mustermann</xyz:Creator>
     <xyz:Title>Redaktion von www.newsmuster.org</xyz:Title>
     <xyz:Date>01-01-2005</xyz:Date>
<rdf:Description>
```

Hat ein Description-Element kein about-Attribut, so wird dadurch eine neue Ressource repräsentiert, die ein Ersatz für eine physische Ressource ohne festlegbaren URI sein kann.

5.3.1 Abbreviated Syntax – kompakte Syntax

Es wurde eine kompakte Schreibweise entwickelt, damit die Elementinhalte nicht als „freier Text“ codiert sind und dann direkt sichtbar werden, wenn z.B. die RDF-Beschreibung in ein html-Dokument eingebettet wird. Mit der kompakten Schreibweise ist es möglich, Informationen in so genannten Eigenschaftsattributen abzulegen. Die Attribute werden dann genau so benannt, wie die Eigenschaftselemente, die sie ersetzen. Das mehrfache Auftreten gleich benannter Attribute in einem Element bzw. die Verschachtelung von Elementen lässt sich auf diese Weise allerdings nicht abbilden. [vgl. Ecks04, S. 245]

Man kann den Code eines RDF-Dokuments auf drei verschiedenen Arten abkürzen:

1) Darstellung von Eigenschaften als Attribute des Description-Tag:

```
<rdf:Description about="www.newsmuster.org/redaktion"
  xyz:Creator="Max Mustermann" />
```

2) Verschachtelung mehrerer Descriptions, wenn das Objekt eines Statements auch eine andere Ressource ist und deren Werte der Eigenschaft Literale sind:

```
<rdf:Description about="www.newsmuster.org/redaktion"
  xyz:Creator rdf:resource="www.newsmuster.org/mitarbeiter/906090" />
</rdf:Description>

                                    +

<rdf:Description about="www.newsmuster.org/mitarbeiter/906090">
  <xyz:Name>Anna Kournikova</xyz:Name>
  <xyz:Email>sportredaktion@newsmuster.org</xyz:Email>
</rdf:Description>

                                    =
<rdf:Description about="www.newsmuster.org/redaktion">
  <xyz:Creator>
     <rdf:Description about="www.newsmuster.org/mitarbeiter/906090">
     <xyz:Name>Anna Kournikova</xyz:Name>
     <xyz:Email>sportredaktion@newsmuster.org</xyz:Email>
     </rdf:Description>
```

```
  </xyz:Creator>
</rdf:Description>
```

3) der Klassentyp, der im Schema definiert ist, und im *rdf:type*-Attribut angegeben wird, kann direkt als Elementname im Description-Tag verwendet werden, statt im Attribut genannt zu werden.

```
<rdf:Description about=”www.newsmuster.org/mitarbeiter/906090”>
  <rdf:type resource=”http://www.newsmuster.org/unserschema#Person”>
  <xyz:Name>Anna Kournikova</xyz:Name>
  <xyz:Email>sportredaktion@newsmuster.org</xyz:Email>
</rdf:Description>

<xyz:Person about=”http://www.newsmuster.org/mitarbeiter/906090“>
  <xyz:Name>Anna Kournikova</xyz:Name>
  <xyz:Email>sportredaktion@newsmuster.org</xyz:Email>
</rdf:Description>
```

5.4 Container

Für den Fall, dass man mehrere gleichartige Beschreibungen zusammenfassen möchte, bietet RDF das Konzept der *Container* an. Besitzt eine Ressource mehrere Eigenschaften gleichen Typs, können alle dazugehörigen Eigenschaftswerte (Literale oder Ressourcen) zu einem Container zusammengefasst werden.

Man unterscheidet s verschiedene Containertypen:

- Sequenzen (Sequenz)
- Multimengen (Bag)
- Alternativen (Alternative)

Bei einer **Sequenz** handelt es sich um eine geordnete Liste von Ressourcen oder Literalen, d.h. die Reihenfolge der Elemente spielt eine Rolle. Bei Büchern z.B. werden die Autoren immer in einer bestimmten Reihenfolge angegeben.

Eine **Multimenge** ist eine ungeordnete Liste von Ressourcen oder Literalen, d.h. die Reihenfolge ist unerheblich – beispielsweise eine Lehrveranstaltung, die von mehreren Teilnehmern besucht wird.

Eine **Alternative** lässt die Auswahl eines ihrer Elemente zu – beispielsweise wählt man auf einer Webseite die Sprache „Deutsch“ aus mehreren Möglichkeiten. Im Gegensatz zu Sequenz und Multimenge sind keine Duplikate in der Liste erlaubt.

5.4.1 Darstellung von Containern

Für die Darstellung eines Containers wird eine weitere Ressource benötigt. Mit dem Eigenschafts-Typ *rdf:type* wird eine Instanz entsprechend einer der drei Containertypen deklariert. Die Zugehörigkeit der Elemente zum Container wird über eine Nummerierung der Eigenschaften hergestellt.

Beispiel:
Die RDF-Aussage lautet: „Die Autoren des Buches Geographic Information Systems and Science lauten Paul A. Longley und Michael F. Goodchild.“

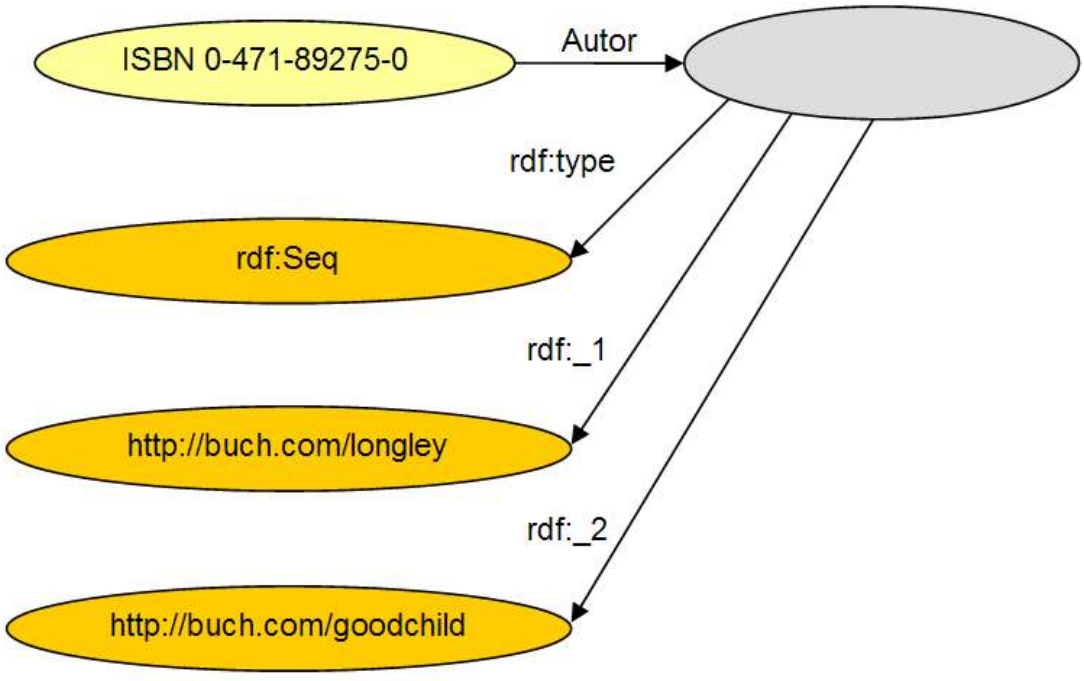


Abbildung 14: Statement mit einem Sequenz-Container

Lösen Sie in Zusammenhang mit der Abbildung 14 das RDF Beispiel 3 im Übungsseitenteil auf Seite 83.

Der Container wird wie eine Ressource mit den Containerelementen als Eigenschaften dargestellt. Anstelle der Nummerierung verwendet man den Elementnamen *li* (list item).

Statt des Containers könnten auch voneinander unabhängige Aussagen formuliert werden, die dasselbe Subjekt besitzen – jedoch kann dabei der inhaltliche Zusammenhang verloren gehen.

Beispiel:
Die Aussage „Der Vorstand bestehend aus dem Vorsitzenden, dem 1. Stellvertreter und dem 2. Stellvertreter hat das Budget 2006 angenommen.“ verliert bei drei einzelnen Aussagen die Information, dass alle Mitglieder des Vorstandes (Containerelemente) das Budget einstimmig angenommen haben. Stattdessen hätten dann der Vorsitzende, der 1. Stellvertreter und der 2. Stellvertreter unabhängig voneinander das Budget angenommen.

Wenn mehrere Ressourcen die gleichen Eigenschaften besitzen, kann in einer Aussage auch das Subjekt durch einen Container dargestellt werden.

```
<rdf:Description about="http://www.newsmuster.org/redaktion">
   <xyz:Creator> Arabella Kiesbauer </xyz:Creator>
</rdf:Description>
<rdf:Description about="http://www.newsmuster.org/gesellschaft">
   <xyz:Creator>Arabella Kiesbauer</xyz:Creator>
</rdf:Description>
```

Zusammengefasst in einen Container:

```
<rdf:Bag ID="Newssites">
   <rdf:li resource="http://www.newsmuster.org/redaktion" />
   <rdf:li resource="http://www.newsmuster.org/gesellschaft" />
</rdf:Bag>

<rdf:Description aboutEach="#Newssites">
   <xyz:Creator>Arabelle Kiesbauer</xyz:Creator>
</rdf:Description>
```

Das *aboutEach*-Attribut wird verwendet, damit sich die Aussage auf die einzelnen Containerelemente bezieht. Die einzelnen Aussagen sind syntaktisch gleich mit der zusammengefassten Aussage.

Es kommt vor, dass Aussagen einerseits über eine gesamte Website, aber andererseits auch über viele Elemente einer einzelnen Seite getroffen werden sollen. Dazu verwendet man das *aboutEachPrefix*-Attribut. Dabei wird jede Ressource, bei der der URI mit dem Präfix beginnt, angesprochen.

```
<rdf:Description aboutEachPrefix="www.newsmuster.org/">
   <xyz:Creator>Arabella Kiesbauer</xyz:Creator>
</rdf:Description>
```

5.5 Reification - Aussagen über Aussagen

Beispiel:

“Herr Mayer ist der Ersteller der Website www.mayer.at.“

Macht man nun eine Aussage über diese Aussage, kann das so aussehen:

„Herr Huber sagt, dass Herr Mayer der Ersteller der Website www.mayer.at ist.“

In RDF wird darüber etwas ausgesagt, was Herr Huber sagt – über die Website selbst wird keine Aussage gemacht. Für die Abbildung in einem Modell werden fünf Eigenschaften benötigt: subject, predicate, object, type und attributedTo.

Lösen Sie nun das Beispiel 4 im Übungsbeispielteil auf Seite 83.

5.6RDF-Schema (RDFS)

Mit Hilfe des RDF-Schema können Begriffe semantisch zueinander in Beziehung gesetzt werden, wobei es verschiedene Beziehungsarten gibt. So können Begriffe, aus denen die Elemente- und Attributnamen gebildet werden, und die, die in den Beschreibungen der Ressourcen Verwendung finden sollen, erklärt werden. [siehe Ecks04, S. 259]

Das RDF-Schema ist ein Vokabular zur Formulierung von Ontologien in RDF und lässt sich nur bedingt mit dem XML-Schema vergleichen. Es erweitert RDF um Konstrukte zur Kreierung von Schemata. RDFS liegt die Idee des mengentheoretischen Klassenmodells zugrunde, wobei Klassen und Eigenschaften se-

parat von einander modelliert werden. Das Klassenkonzept macht es möglich, eine formale Beschreibung der Semantik der verwendeten RDF-Elemente festzulegen. Neben RDFS gibt es weitere Ontologiebeschreibungssprachen wie F-Logic, DAML+OIL und OWL.

5.6.1 Klassen und Eigenschaften

RDFS definiert eine Reihe von Klassen und Eigenschaften, die zur Erstellung von Schemata benutzt werden können.

Lösen Sie dazu das Beispiel 5 im Übungsbeispielteil auf Seite 84.

Ein Schema legt für jede Eigenschaft fest:

- welche Werte erlaubt sind
- welche Bedeutung die Eigenschaft hat
- welche Ressource die Werte besitzen darf
- welche Beziehung zu anderen Eigenschaften besteht

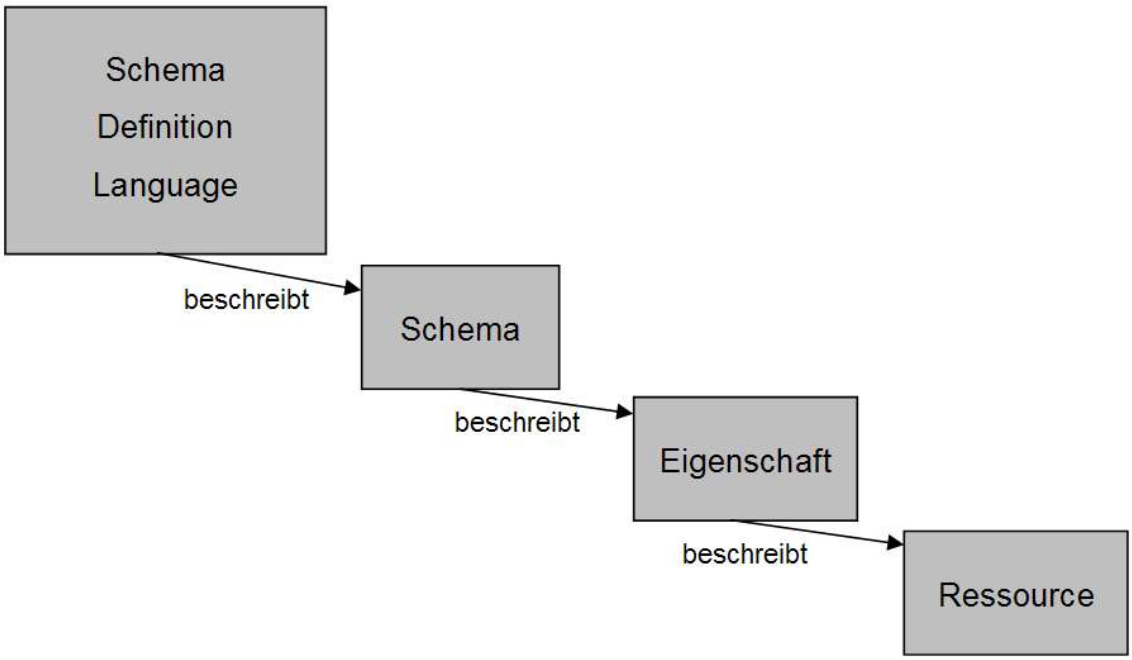


Abbildung 15: RDF-Schema

Kernklassen des RDF-Schema:

- **rdfs:Resource**: alle in RDF beschriebene Dinge sind Instanzen dieser Klasse und haben daher automatisch (implizit) eine *rdf:type*-Eigenschaft mit dem Wert *rdfs:Resource*

- **rdfs:Class**: ist die Zuweisung einer Ressource zu einem Klassentypen; das Konzept ist ähnlich dem der Klassen in objektorientierten Sprachen. Jede Klasse, die definiert wird, muss also eine *rdf:type*-Eigenschaft mit dem Wert *rdfs:Class* haben
- **rdfs:Property**: Instanzen dieser Kernklasse sind alle Klassen, die als Eigenschaft die Beziehung zwischen Klassen ausdrücken; dazu gehört neben der Eigenschaft *rdf:type* auch die Eigenschaften *rdfs:subClassOf*, *rdfs:subPropertyOf*, r*dfs:seeAlso* und *rdfs:isDefinedBy*

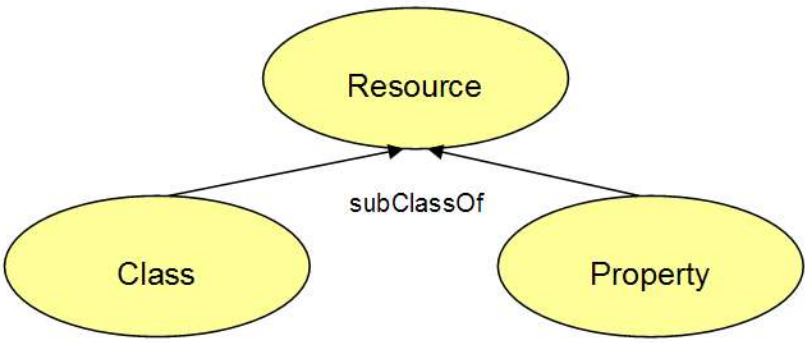


Abbildung 16: Resource, Class, Property

Kern-Properties des RDF-Schema:

- **rdfs:subClassOf**: beschreibt die Beziehung zwischen Ober- und Unterklassen und entspricht dem Vererbungsmechanismus in objektorientierten Sprachen
- **rdfs:subPropertyOf**: damit kann eine Spezialisierung der Eigenschaft ausgedrückt werden. Wenn die Eigenschaft *ist_Mutter_von* eine Untereigenschaft von *ist_Elternteil_von* ist, dann gilt für „Anna *ist_Mutter_von* Karin“ auch „Anna *ist_Elternteil_von* Karin“.

Kern-Constraints des RDF-Schema:

- **rdfs:range**: damit wird der Wertebereich einer Eigenschaft festgelegt; eine Eigenschaft kann höchstens eine rdfs:range-Eigenschaft haben
- **rdfs:domain**: damit wird der Definitionsbereich einer Eigenschaft festgelegt; eine Eigenschaft kann höchstens eine rdfs:domain-Eigenschaft haben

Schemata sind statisch und ändern sich nicht, denn eine Schemadefinition wird im Internet von sehr vielen verschiedenen Personen benützt und Änderungen in einem Schema würden Änderungen in einer großen Anzahl anderer Dokumente nach sich ziehen sowie Anpassungen der Anwendungen.

Wenn doch Änderungen erforderlich sind, muss ein neues Schema erstellt erden, das das alte Schema erweitert. Bei den Teilen, die ohne Änderung übernommen werden, muss deutlich werden, dass sich die Bedeutung nicht geändert hat [vgl. Stum02, S.10]. Das geschieht mit rdfs:subClassOf

```
<rdf:Description rdf:ID="XY"
     rdfs:subClassOf="http://beispiel.org/altesschema#XY"/>
```

Beispiel:

Die Mitarbeiter eines Unternehmens können verschiedenen Projekten zugeordnet werden. Eine Ressource „Hr. Mayer“ kann sowohl eine Instanz der Klasse „Projekt A“, als auch eine Instanz der Klasse „Projekt B“ sein. Darüber hinaus können auch Subklassen gebildet werden – das können z.B. Arbeitsgruppen sein, die innerhalb der Projekte gebildet werden. Analog zu den Klassen von Objekten können auch Eigenschaften und Subeigenschafen definiert werden. Man kann z.B. die Eigenschaft „hat_Fortbildungskurs“ und dazu die Subeigenschaften „hat_einen_Fortbildungskurs“, „hat_zwei_Fortbildungskurse“, „hat_drei_Fortbildungskurse“, usw. definieren.

Damit hat man die Möglichkeit, einfache hierarchische Strukturen zu erzeugen, aus denen zusätzliche, nicht explizite Information gewonnen werden kann. Hat ein Projekt z.B. das Thema „Effizienzsteigerung“ und eine Arbeitsgruppe dieses Projekts bearbeitet das Thema „Prozessoptimierung“, so kann man daraus ableiten, dass „Prozessoptimierung“ ein Unterthema von „Effizienzsteigerung“ ist. Ein weiteres Beispiel ist eine Klasse mit den Eigenschaften „hat_Fortbildungskurs“ und „erhält_den_Bildungsbonus“. Wird dieser Klasse eine Subklasse mit der Eigenschaft „hat_einen_Fortbildungskurs“ zugeordnet, dann hat jede Instanz dieser Subklasse automatisch auch die Eigenschaft „erhält_den_Bildungsbonus“, ohne das das explizit angeführt werden muss.

Das RDF-Schema bietet darüber hinaus die Möglichkeit, Eigenschaften zu beschreiben. Dafür definiert man für Subjekte und Objekte, zu welchen

Klassen sie gehören müssen. Eine Eigenschaft „ist_Vorgesetzter_von“ als Subjekt kann dann z.B. nur Instanzen einer Klasse „Person“ und als Objekt nur Instanzen einer Klasse „Mitarbeiter“ zulassen. Damit können sinnlose Aussagen wie „ein Kopierer ist_Vorgesetzter_von einem Firmenwagen“ ausgeschlossen werden.

Das im RDF-Modell benützte Vokabular kann mit dem RDF-Schema typisiert und durch Vererbungen von Eigenschaften ausgedrückt werden. Die formale Beschreibung der Semantik ermöglicht es, Informationen aus den im Internet zur Verfügung stehenden Daten abzuleiten. Über Vererbungsbeziehungen können implizite Attribute explizit gemacht werden.

Das RDF-Schema ist eine einfache Modellierungssprache, die zur Formulierung von einfachen Ontologien dient. Ressourcen und ihre Beziehungen können allerdings untereinander nicht sehr detailliert dargestellt werden – dafür kommen die Ontologiesprachen zum Einsatz.

5.7Dublin Core Element Set

Eigenschaften die Dinge beschreiben, sind eine Teilmenge der Ressourcen, sie können daher auch selbst beschrieben werden und müssen darüber hinaus einen Unified Resource Identifier (URI) haben. Wird über bestimmte Themengebiete kommuniziert, besteht allerdings die Möglichkeit, dass die Kommunikationspartner jeweils unterschiedliche Vorstellungen von den enthaltenen Konzepten haben und daher ist es wichtig, sich auf bestimmte Definitionen zu einigen. Durch die Verwendung einheitlicher URI´s für bestimmte Eigenschaften kann diesem Umstand Rechnung getragen werden.

Im so genannten „Dublin Core Element Set“ wurde ein Namensraum mit allgemeinen Eigenschaften zur Kategorisierung und Beschreibung von Webseiten erstellt.

Das „Dublin Core Element Set“ besteht aus 15 Datenfeldern:

1 Title
2 Creator
3 Subject
4 Description
5 Publisher
6 Contributor
7 Date
8 Type
9 Format
10 Identifier
11 Source
12 Language
13 Relation
14 Coverage
15 Rights

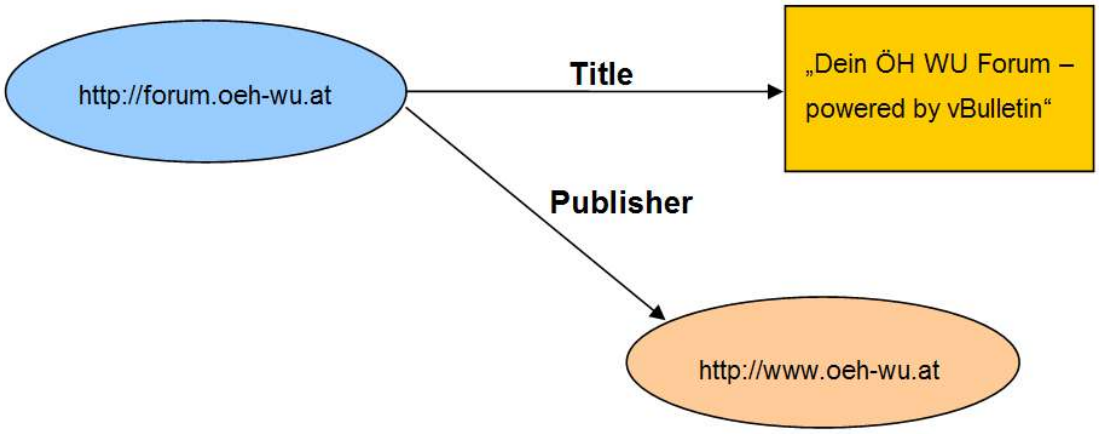


Abbildung 17: RDF, Dublin Core Elemente

Die Internetseite http://forum.oeh-wu.at hat den Titel „Dein ÖH WU Forum – powered by vBulletin“ und den Herausgeber http://www.oeh-wu.at. Die Eigenschaften Titel (Title) und Herausgeber (Publisher) sind dem „Dublin Core Element Set“ entnommen.

5.8 RSS

RSS ist eine XML-basierte Familie von Dateiformaten und steht für die folgenden Standards:

- Rich Site Summary (RSS 0.9x)
- RDF Site Summary (RSS 1.0)
- Really Simple Syndication (RSS 2.0)
- Really Simple Syndication (RSS 3.0)

RSS wird verwendet, um Artikel oder deren Kurzbeschreibungen auf Webpräsenzen (insbesondere Nachrichtenmeldungen) zu speichern und in maschinenlesbarer Form bereitzustellen. Ein sogenannter RSS-Feed oder Newsfeed besteht aus einer XML-Datei, welche nur den Inhalt – beispielsweise einer Nachrichtenseite – bereithält, aber keinerlei Layouts oder Design beinhaltet. Viele Seiten, die regelmäßig Artikel publizieren, generieren eine solche RSS-Datei mit den neuesten Artikeln und veröffentlichen diese auf ihrer Webpräsenz [siehe Wiki m].

Lösen Sie das Beispiel 6 im Übungsbeispielteil auf Seite 84.

5.9 RDF Abfragesprachen

Um Daten aus RDF auszulesen und damit Weiterverarbeitungsmöglichkeiten zu bieten, ist eine eigene Abfragesprache nötig. Nachdem es bereits einige Abfragesprachen wie „RQL“, „N3“ usw. gibt, hat sich das W3C das Ziel gesetzt, einen einheitlichen Standard für RDF Abfragen zu schaffen[5]. Diese hat den Namen SPARQL (**S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage[6]). Die Schaffung der Spezifikationen ist zurzeit (August 2005) im Gange und befindet sich

[5] Nähere Informationen zu bereits bestehenden RDF Abfragesprachen können dem Report „A Comparison of RDF Query Languages“ von Peter Haase, Jeen Broekstra, Andreas Eberhart und Raphael Volz 2004, entnommen werden, in dem die RDF Abragesprachen RQL, SeRQL, TRIPLE, RDQL, N3 und Versa beschrieben und miteinader verglichen werden.

[6] In früheren Versionen stand SPARQL für **S**imple **P**rotocol **a**nd **R**DF **Q**uery **L**anguage und wurde erst später zu dem endgültig rekursiven Akronym **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage

derzeit im Status „Last Call Working Draft“ der am 21. Juli 2005 veröffentlicht wurde. Es sind nur noch kleinere Änderungen bis zur endgültigen Version zu erwarten.

Mittels SPARQL können aus einem RDF Graphen Informationen über den URI, dem Knoten und natürlich den Daten des RDF Graphen ausgelesen werden. Des Weiteren können durch SPARQL RDF Subgraphen extrahiert und neue RDF Graphen, die auf den Informationen des abgefragten Graphen basieren, erstellt werden.

Anhand des folgenden Beispiels wird eine einfache SPARQL Abfrage dargestellt :
Ausgangsdaten sind folgendes RDF Triple:

```
<http://www.newsmuster.org/leitartikel>
        <http://purl.org/dc/elements/1.1/title> "Semantic Web"
```

Das RDF Triple sagt aus, dass „<http://www.newsmuster.org/leitartikel>" den Titel "Semantic Web" hat. Der Ausdruck „<http://purl.org/dc/elements/1.1/title>" ist eine spezifische Definition für den Titel eines Werkes der Dublin Core Metadata Initiative. Damit wird sichergestellt, dass es eindeutig feststellbar ist, dass es sich um den Titel eines Werkes handelt, und nicht um irgendeinen anderen Titel, wie zum Beispiel die Titelbezeichnung eines Menschen oder einfach um die Aneinanderreihung der Buchstaben „t“ „i“ „t“ „l“ und „e“. Will man nun den Titel von <http://www.-newsmuster.org/leitartikel> wissen, sieht die SPARQL Abfrage folgendermaßen aus:

```
SELECT ?title

WHERE
{
    <http://www.newsmuster.org/leitartikel>   <http://purl.org/dc/ele-
ments/1.1/title> ?title .
}
```

Als Ergebnis erhält man "Semantic Web".

5.10 RDF in der Praxis

Der Einsatz von RDF in der Praxis erfährt derzeit eine steigende Tendenz, da immer mehr Beispiele die Verbesserungen die durch RDF möglich sind zeigen. So ist es etwa Vodafone durch den Einsatz von RDF auf deren mobilen Vodafone Live Portal, auf dem Klingeltöne, Spiele und Handybilder zum Download angeboten werden, den Klingeltonumsatz um 20% zu steigern. Gleichzeitig wurde durch die semantische Verknüpfung der Daten eine auf die Kundenbedürfnisse zugeschnittene Auswertung der Suchanfragen möglich, sodass die Anzahl der Seiten, die besucht werden bis ein Download getätigt wird, halbiert werden konnte. Auch die Forum Seite von Nokia http://www.forum.nokia.com setzt auf RDF [W3Ce].

Abbildung 18: Vodafone Live Portal

5.11 Exkurs: Graphentheorie

Die Graphentheorie stellt ein Teilgebiet der Mathematik dar, das sich mit Eigenschaften von Graphen und ihren Beziehungen untereinander beschäftigt. Als einen Graphen bezeichnet man ein Gebilde aus Knoten (Punkte) die durch Kanten (Linien) miteinander verbunden sind. Die Form der Knoten und Kanten

kann hierbei frei gewählt werden. Mathematische Strukturen werden durch Graphen grafisch darstellbar. Ein Graph besteht aus der Menge der enthalten Konten V (V für „vertex“ engl. für Knoten) und der Menge der Kanten E (E für „edge“ engl. für Kante).

$$G = (V, E)$$

In der Graphentheorie wird einerseits zwischen ungerichteten und gerichteten Graphen sowie andererseits zwischen Graphen mit und ohne Mehrfachkanten unterschieden. Bei gerichtete Graphen haben die Kanten im Gegensatz zu ungerichteten Kanten eine Orientierung. Diese Orientierung wird meist mittels eines Pfeils dargestellt. Mehrfachkanten sagen aus, dass mehrere Kanten zwischen den gleichen Knoten verlaufen und diese zusätzlich bei gerichteten Graphen dieselbe Orientierung haben. In den folgenden Abbildungen 19 bis 22 werden die einzelnen Graphenformen dargestellt [wiki g] .

Abbildung 19: Ungerichteter Graph ohne Mehrfachkanten

Abbildung 20: Gerichteter Graph ohne Mehrfachkanten

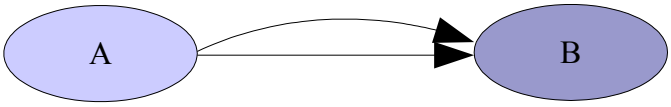


Abbildung 21: Gerichteter Graph mit Mehrfachkanten

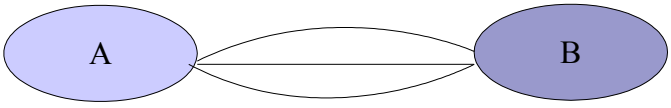


Abbildung 22: Ungerichteter Graph mit Mehrfachkanten

5.11.1 RDF Triple in Graphentheorie

Wenn wir nun den Aufbau von RDF Graphen anhand der Abbildung 5: Aufbau von RDF-Aussagen betrachenten können wir feststellen, dass es sich um einen gerichteten Graphen ohne Mehrfachkanten handelt. Der Knoten „Ressource“ zeigt durch die Kante „Eigenschaft“ auf den Knoten „Wert“.

Abbildung 23: Graph aus Abbildung 5: Aufbau von RDF-Aussagen

6 Ontologien

6.1 Was sind Ontologien

Eine Ontologie im IT - Sinn ist eine geschaffene Sprache die aus einem spezifischen Vokabular, um einen Teil der Wirklichkeit (Wissensbereich) zu beschreiben, und Regeln über die Verwendung des Vokabulars besteht. [Grub93]. Ontologien regeln, welche Eigenschaften, Funktionen und Beziehungen Objekte im beschriebenen Wissensbereich selbst und zu anderen Objekten haben. Kurz gesagt: Ontologien spezifizieren Konzepte!

Ontologien müssen die Beschreibung von Wissensgebieten so darstellen, dass sie von anderen verstanden und verwendet werden können, da ohne einheitliche Sprache keine Information ausgetauscht werden kann.

Ontologien bauen auf dem Konzept von Taxonomien und Thesauren auf. Taxonomien sind hierarchische Klassifikationen von Objekten der realen Welt, die typischer Weise in einer Baumstruktur dargestellt werden. (z. B. „Tennis ist eine Subklasse von Sportart“). Bei Thesauren werden Objekte beliebig miteinander in Beziehung gesetzt (z. B. „A ist ein B“, „A ist verwandt mit B“). Taxonomien und deren Unterschied zu Ontologien wird im Kapitel 6.9 Taxonomie vs. Ontologie näher behandelt.

6.2 Anforderungen an die Ontologie für das Semantic Web

Ontologien für das Semantic Web bauen auf den bereits in den vorherigen Kapiteln beschriebenen Konzepten von XML und RDF auf. Diese Einordnung kann in der folgenden Grafik nochmals ersehen werden.

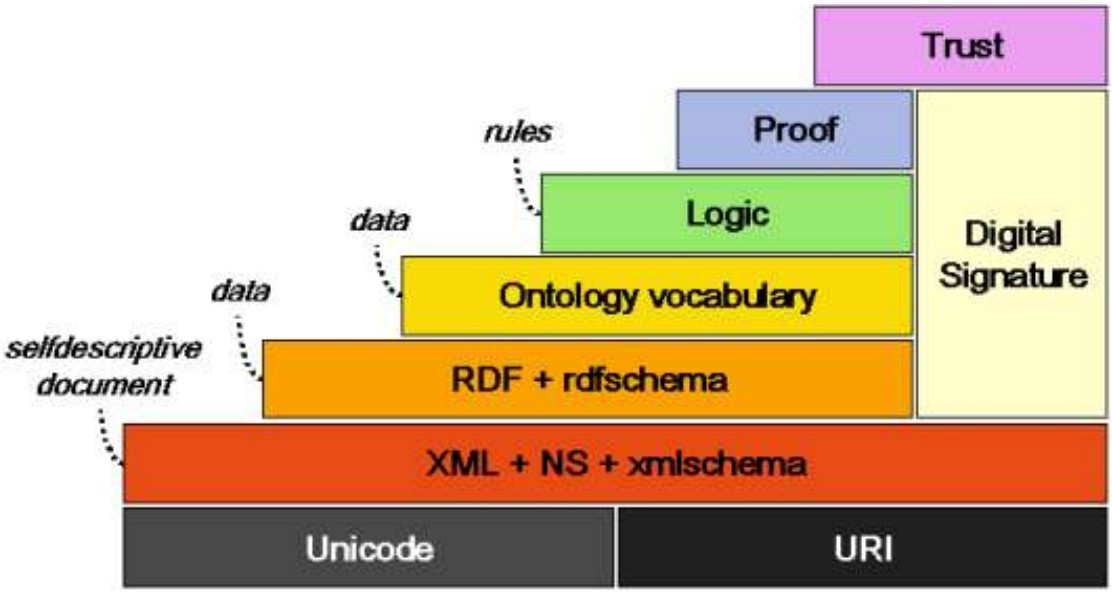


Abbildung 24: Schichtenmodell von Berners-Lee [KoMi01]

XML übernimmt die Aufgabe benutzerdefinierte Schemata zu definieren, und RDF wird als flexibler Ansatz verwendet, um Daten abzubilden. Nun werden Ontologien gebraucht, die die in den Dokumenten verwendeten Ausdrücke ausdrücklich und formell beschreiben. Dies geschieht durch zusätzliche Sprachausdrücke einhergehend mit formaler Semantik. Die Wichtigkeit von Ontologien für das Semantic Web liegt darin, dass zur Schaffung von leistungsfähigen Web – Applikationen, die Daten interpretieren und miteinander in Beziehung setzen können, die Basissemantik von RDF nicht ausreicht.

Um diese Anforderungen für das Semantic Web zu erfüllen, muss die Ontologie Sprache für das Semantic Web einige Punkte erfüllen [vgl. W3Cb]:

- Ontologien sollten öffentlich zugänglich sein und zusätzlich durch andere Ontologien erweiterbar sein.
- Einerseits sollen Ontologien ein weites Spektrum an Wissen darstellen, anderseits sollen auch effiziente Möglichkeiten bestehen, daraus Schlüsse zu ziehen. Darum muss ein Mittelweg zwischen Ausdruckstärke und Skalierbarkeit gefunden werden.
- Die Sprache soll einfach zu erlernen sein, und ein klares Konzept und Ausdrucksweise haben.
- Ontologien müssen eindeutig identifizierbar sein. (URI)

- Verschiedene Ontologien können ein und dasselbe Konzept durch die eindeutige Unterscheidbarkeit auf verschiedenste Art und Weise darstellen.
- Jede Ontologie muss Metadaten über den Ersteller, Erstelldatum usw. integrieren.
- Die geschaffene Sprache muss es ermöglichen verschiedene Versionen einer Ontologie zu vergleichen und in Verbindung zu bringen.
- Klassen müssen wie Instanzen behandelt werden können

6.3 Web Ontology Language

Zur Einbindung von Ontologien in das Semantic Web wurde unter Einhaltung der Anforderungen der W3C Web Ontology Working Group die Web Ontology Language (kurz: OWL[7]) geschaffen. Die Spezifikationen für OWL wurden am 10. Februar 2004 verabschiedet. [W3Ca]

Es wird aufbauend auf der Syntax von XML und semantischen Sprachausdrücken eine Sprache geboten, die es ermöglicht Bedeutungen für OWL Ontologien zu beschreiben. Dadurch ist OWL für Anwendungen geeignet, die Information weiterverarbeiten und sie nicht nur für Menschen darstellen. Durch die Verwendung eines zusätzlichen Vokabulars und formaler Semantik wird eine größere Maschinenverarbeitungsmöglichkeit von Informationen geboten, als dies mit XML, RDF oder auch RDF Schema möglich wäre. OWL ist in drei Subsprachen, die sich in der Aussagenkräftigkeit unterscheiden, gegliedert:

- OWL Lite
- OWL DL (Description Logic)
- OWL Full

Im Folgenden sollen die drei Subsprachen von OWL kurz vorgestellt werden.

OWL LITE

OWL Lite kann als Basisversion bzw. „light“ Version von OWL gesehen werden. Es wird verwendet um eine einheitliche Hierarchie oder einfache eingeschränkte Abstraktionen darzustellen [vgl. W3Cc].

OWL DL Description Logic

OWL DL ist für jene Nutzer gedacht, die ein Maximum an Ausdrucksstärke möchten, ohne dabei auf vollständige Verarbeitbarkeit und Entscheidbarkeit verzichten zu müssen [vgl. W3Cc].

OWL Full

[7] Die Abkürzung für Web Ontology Language müsste eigentlich WOL sein. Doch als Referenz an die Eule in der Kindergeschichte Winnie Puuh, die in der englischen Originalfassung ihren Namen „Eule“ „OWL“ immer falsch als „WOL“ schreibt, wurde die Vertauschung der Buchstaben umgekehrt. So wird aus Web Ontology Language WOL -> OWL.

OWL Full bietet ein Maximum an Ausdrucksstärke und syntaktische Freiheit von RDF, jedoch keine Garantie für Verarbeitbarkeit und Entscheidbarkeit [vgl. W3Cc].

Da die drei OWL Subsprachen jeweils eine Erweiterung der aussagenschwächeren Subsprache darstellen, ist die aussagenschwächere Sprache eine gültige Version der aussagenstärkeren Version. Jedoch sind Ontologien die in einer aussagenkräftigeren Subsprache geschrieben worden sind, nicht zu aussagenschwächeren kompatibel. (keine „Abwärtskompatibiliät“)

Daraus können folgende Schlüsse gezogen werden:

Jede korrekte OWL Lite Ontologie ist eine korrekte OWL DL Ontologie.

Jede korrekte OWL DL Ontologie ist eine korrekte OWL Full Ontologie

Jede korrekte OWL Lite Ontologie ist eine korrekte OWL Full Ontologie

Jede gültige OWL Lite Schlussfolgerung ist eine gültige OWL DL Schlussfolgerung

Jede gültige OWL DL Schlussfolgerung ist eine gültige OWL Full Schlussfolgerung

Jede gültige OWL Lite Schlussfolgerung ist eine gültige OWL Full Schlussfolgerung

Diese Aussagen sollen mit der nachfolgenden Grafik noch einmal verdeutlicht werden.

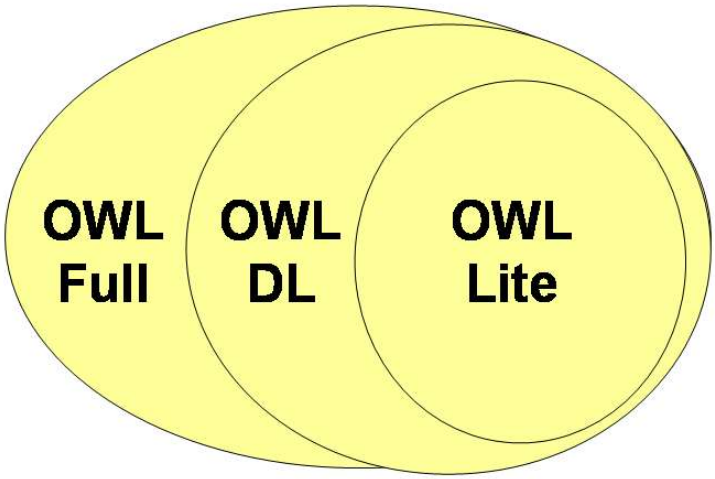


Abbildung 25: Zusammenhang OWL Full, OWL DL und OWL Lite [Schr04]

In der Gegenüberstellung mit RDF können folgende Aussagen getroffen werden. OWL Full kann als Erweiterung von RDF gesehen werden. OWL Lite und OWL DL können nur als Erweiterungen einer eingeschränkten Sicht auf RDF bezeichnet werden.

Daraus können folgende Schlüsse gezogen werden:

- Jedes OWL (lite, DL und Full) Dokument ist ein RDF Dokument.
- Jede RDF Dokument ist ein gültiges OWL Full Dokument.
- Nur bestimmte RDF Dokumente sind gültige OWL DL bzw. OWL lite Dokumente

6.4OWL LITE

Die ausdrucksschwächste Subsprache von OWL ist OWL Lite. Durch sie lassen sich einfache Klassenhierarchien und Restriktionen darstellen. Durch die geringere Komplexität von OWL Lite ist es leicht Thesauri und Taxonomien zu migrieren, sowie Tools zu entwickeln.

OWL Lite bietet Sprachausdrücken die im Bezug zum RDF Schema stehen, Ausdrücke der Property Einschränkungen, Klassen Schnittmenge, Datentypen, (In)Äquivalenz, eingeschränkte Kardinalitäten, Versionierung; Property Characteristika, Header Informationen und Anmerkungen zu den Properties. Die genauen Sprachausdrücke können in [W3Cd] nachgelesen werden.

6.5OWL DL (DESCRIPTION LOGIC)

OWL DL ist für jene Nutzer gedacht, denen die Ausdrucksstärke von OWL Lite nicht genügt und wenn eine vollständige Verarbeitbarkeit und Entscheidbarkeit gegeben sein soll. Es können sämtliche OWL Konstrukte verwendet werden, die jedoch gewissen Beschränkungen unterliegen. OWL DL verlangt Typenunterscheidung. Dies bedeutet, dass eine Klasse nicht gleichzeitig ein Individual oder eine Eigenschaft (Property) sein kann. Desweiteren kann eine Eigenschaft (Property) nicht auch gleichzeitig ein Individual oder eine Klasse sein. Restriktionen können daher mit OWL DL nicht auf die Sprachelemente von OWL selbst angewendet werden. Eine weitere Restriktion ist, dass Eigenschaften (Properties) entweder „ObjectProperties“ „oder DataProperties“ sind [vgl. W3Cd]

OWL DL entspricht einer entscheidbaren terminologischen Logik (Description Logic (DL)) und damit einer Teilmenge der Prädikaten - Logik erster Stufe, erlaubt also auch keine Metabegriffe und keine „semantischen“ Relationen mit Begriffen als Argumenten.

OWL DL erweitert die Sprachausdrücke von OWL Lite um Ausdrücke für Klassen Grundsätze, wählbaren Kardinalitäten, Boolean Kombinationen von Klassenausdrücken und Wertinformation. Die genauen Sprachausdrücke können wiederum unter [W3Cd] nachgelesen werden.

6.6 OWL Full

OWL Full ist für Benutzer gedacht, die maximale Ausdrucksstärke und die syntaktische Freiheit von RDF möchten. Sie müssen dabei jedoch auf die Garantie der Verarbeitbarkeit verzichten. OWL Full erlaubt es einer Ontologie die Bedeutung von vordefiniertem (RDF oder OWL) Vokabular zu erweitern. OWL Full nützt denselben Syntax wie OWL DL. Der Unterschied zwischen den Sprachen liegt darin, dass OWL Full keinen Restriktionen unterliegt. Es muss auf die bereits erwähnten Restriktionen wie Typenunterscheidung oder Eigenschaftsrestriktionen keine Rücksicht genommen werden.

6.7 Owl Erklärungsbeispiel

Im Folgenden ist eine einfache Ontologie über den Nachrichtenbereich dargestellt. Es ist zu beachten, dass nur einige der wichtigsten Aspekte beleuchtet werden und kein Anspruch auf Vollständigkeit der Anwendungsmöglichkeiten von OWL gestellt werden kann.

Die Zeile [6] identifiziert den Namespace der Ontologie. Zeile [7] identifiziert die Basis – URI dieses Dokuments. Diese wurde frei mit „http://www.ontologien.at/news.owl“ gewählt. Von Zeile 3 bis 6 werden die fix definierten Namespaces für OWL [5], RDF [3], RDF Schema [4] und XML Schema Datentypen definiert. Unter [8] könnte man einen Namen oder eine Referenz für die Ontologie angeben. Wenn wie in unserem Fall nichts angegeben ist, was auch der Standardfall ist, ist die URI des Dokuments der Name. Es sind die Klassen Autor und Nachricht (mit den Subklassen Sport und Politik) in [9 – 19] deklariert. Als Restriktion wurde definiert, dass eine Nachricht von einem Autor geschrieben wird [20 – 30]. „Staberl“ [37] und „Dr. Schneckerl“ [38] sind Instanzen der Klasse Autor. Von [40 – 50] wird je eine Sport- und eine Politiknachricht instanziert. Detailierte Informationen zum Aufbau von OWL können unter der URL http://www.w3.org/TR/2004/REC-owl-guide-20040210/ gefunden werden.

```
1<?xml version="1.0"?>
2<rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
 4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 5  xmlns:owl="http://www.w3.org/2002/07/owl#"
 6   xmlns="http://www.ontologien.at/news.owl#"
 7 xml:base="http://www.ontologien.at/news.owl">

8 <owl:Ontology rdf:about=""/>

 9 <owl:Class rdf:ID="Sport">
10   <rdfs:subClassOf>
11     <owl:Class rdf:ID="Nachricht"/>
12   </rdfs:subClassOf>
13 </owl:Class>
14 <owl:Class rdf:ID="Autor"/>
15 <owl:Class rdf:ID="Politik">
16   <rdfs:subClassOf>
17     <owl:Class rdf:about="#Nachricht"/>
18   </rdfs:subClassOf>
19 </owl:Class>

20 <owl:Class rdf:about="#Nachricht">
21   <rdfs:subClassOf>
22     <owl:Restriction>
23       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLS-
chema#int"
24       >1</owl:cardinality>
25       <owl:onProperty>
26         <owl:ObjectProperty rdf:ID="wurdegeschriebenvon"/>
27       </owl:onProperty>
28     </owl:Restriction>
29   </rdfs:subClassOf>
30 </owl:Class>
31
32 <owl:ObjectProperty rdf:about="#wurdegeschriebenvon">
33   <rdfs:domain rdf:resource="#Nachricht"/>
34   <rdfs:range rdf:resource="#Autor"/>
35 </owl:ObjectProperty>
36
37 <Autor rdf:ID="Staberl"/>
38 <Autor rdf:ID="Dr.Schneckerl"/>
39
40 <Sport rdf:ID="EM2008">
41  <wurdegeschriebenvon rdf:resource="#Dr.Schneckerl"/>
42   <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
43   >Das EM Stadion in Klagenfurt wird gebaut.</rdfs:comment>
```

```
44 </Sport>

45 <Politik rdf:ID="EUVerfassung">
46   <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
47   > Franzosen sagen NON zur EU-Verfassung</rdfs:comment>
48   <wurdegeschriebenvon rdf:resource="#Staberl"/>
49 </Politik>
50</rdf:RDF>
```

6.8 Vertiefendes Tutorial

Zur Vertiefung soll anhand des folgenden Beispiels einer einfachen Computer Hardware Ontologie gezeigt werden, wie eine Ontologie erschaffen wird. Es werden die Schritte jeweils kurz erläutert und durch dazugehörigen Code ergänzt. Zum Zwecke des besseren Verständnisses wird auf Komplexität verzichtet und die vereinfachte Annahme getroffen, dass ein PC nur aus Mainboard, CPU und Grafikkarte besteht.

Um eine eindeutige interpretierbare Ontologie die von Softwareagenten genutzt werden kann zu erstellen, sind die formale Syntax und die formale Semantik von OWL anzuwenden.

Namespaces

Als erstes muss definiert werden, welches spezifische Vokabular genützt wird. Dazu wird ein „XML namespace“ deklariert. Dieser macht gemäß dem Prinzip der Namespaces die Ontologie eindeutig.

```
<rdf:RDF
1xmlns = "http://www.meineonto.at/pc.owl#"
2xml:base = "http://www.meineonto.at/pc.owl">
3xmlns:owl ="http://www.w3.org/2002/07/owl#"
4xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

Die erste Zeile (1) identifiziert den Namespace der Ontologie. Die Zeile 2 identifiziert die Basis – URI dieses Dokuments. Von Zeile 3 bis 6 werden die (fix definierten) Namespaces für OWL, RDF, RDF Schema und XML Schema Datentypen definiert.

Ontologie Header

Als nächstes werden im Header der Ontologiename [4], Bemerkungen [1], Hinweis auf ältere Versionen [2] oder auch die Einbindung von weiteren Ontologien [3] durchgeführt. Weiters könnt unter anderem noch Inkompatibilität mit „owl:incompatibleWith“ oder Versioninformation mit „owl:versionInfo“ angegeben werden. Unter rdf:about [0] könnte man einen Namen oder eine Referenz für die Ontologe angeben. Wenn wie in unserem Fall durch " "nichts angegeben ist, was auch der Standardfall ist, ist die URI des Dokuments der Name.

```
0 <owl:Ontology rdf:about="">
1  <rdfs:comment>An example OWL ontology</rdfs:comment>
2  <owl:priorVersion
rdf:resource="http://www.meineonto.at/bspalt.owl"/>
3  <owl:imports
rdf:resource="http://www.meineonto.at/grundonto.owl"/>
4  <rdfs:label>Computer Ontology</rdfs:label>
```

Klassen

Nun werden durch die Definition von Klassen Taxonomiebäume geschaffen. Dies geschieht durch `<owl:Class rdf:ID="Klassenname"/>` zur Klassendeklaration durch `<rdfs:subClassOf rdf:resource="#ÜbergeordneteKlasse"/>` um Unterklassen, und damit transitive Abhängigkeiten zu deklarieren. In unserem Beispiel werden als erstes die Klassen Systemkomponenten [1] und Computer [2] geschaffen. Danach werden in [3-11] die Klassen Mainboard, Grafikkarte und CPU jeweils mit Hinweis, das es sich um Subklassen von Systemkomponenten [4, 7 und 10] deklariert.

```
1 <owl:Class rdf:ID="Systemkomponenten"/>
2 <owl:Class rdf:ID="Computer"/>

3  <owl:Class rdf:ID="Mainboard">
4    <rdfs:subClassOf rdf:resource="#Systemkomponenten"/>
5  </owl:Class>

6  <owl:Class rdf:ID="Grafikkarte">
7    <rdfs:subClassOf rdf:resource="#Systemkomponenten"/>
8  </owl:Class>

9  <owl:Class rdf:ID="CPU">
10    <rdfs:subClassOf rdf:resource="#Systemkomponenten"/>
11 </owl:Class>
```

Instanzen

Um Instanzen zu erzeugen werden diese als Individuen der Klasse durch <Klasse rdf:ID="Instanzenname"/> instanziert. Beispielhaft wird unter [1] ein „Intel3Ghz“ Prozessor als Instanz eines CPU's sowie die Grafikkarte „GeForce6600GT“ [2] als auch das Mainboard „AsusA8N“ [3] angelegt

```
1 <CPU rdf:ID="Intel3Ghz"/>
2 <Grafikkarte rdf:ID="GeForce6600GT"/>
3 <Mainboard rdf:ID="AsusA8N"/>
```

Eigenschaften (Properties)

Um Fakten über Klassenmitglieder aber auch über Instanzen anzugeben, müssen diese durch Eigenschaftsdefinitionen in Beziehung gebracht werden. In [1] wird die Objekteigenschaft (Relationen zwischen Instanzen von zwei Klassen) „hatCPU“ deklariert. Die Domain [2] (Grundmenge) und Range [3] (Wertebereich) Deklaration bringt die Instanzen von „Computer“ und „CPU“ in Beziehung. Genauso geschieht es für „hatMainboard“ und „hatGrafikkarte“ [5-12].

```
1<owl:ObjectProperty rdf:ID="hatCPU">
2    <rdfs:domain rdf:resource="#Computer"/>
3    <rdfs:range rdf:resource="#CPU"/>
4  </owl:ObjectProperty>
5 <owl:ObjectProperty rdf:ID="hatMainboard">
6    <rdfs:range rdf:resource="#Mainboard"/>
7    <rdfs:domain rdf:resource="#Computer"/>
8  </owl:ObjectProperty>
9  <owl:ObjectProperty rdf:ID="hatGrafikkarte">
10    <rdfs:domain rdf:resource="#Computer"/>
11    <rdfs:range rdf:resource="#Grafikkarte"/>
12  </owl:ObjectProperty>
```

Nun kann definiert werden, dass ein Computer aus genau einen CPU [1-9], einer Grafikkarte [10 -18] und einem Mainboard [19 – 27] besteht. Dazu wird die Restriktion „cardinality“ verwendet. Dadurch ist nur ein Wert, in unserem Fall „1“ erlaubt.

```
0 <owl:Class rdf:ID="Computer">
1    <rdfs:subClassOf>
2      <owl:Restriction>
3      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLS-
chema#int"
4        >1</owl:cardinality>
5        <owl:onProperty>
6          <owl:ObjectProperty rdf:ID="hatCPU"/>
```

```
7         </owl:onProperty>
8       </owl:Restriction>
9     </rdfs:subClassOf>
10     <rdfs:subClassOf>
11       <owl:Restriction>
12         <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLS-
chema#int"
13         >1</owl:cardinality>
14         <owl:onProperty>
15           <owl:ObjectProperty rdf:ID="hatGrafikkarte"/>
16         </owl:onProperty>
17       </owl:Restriction>
18     </rdfs:subClassOf>
19     <rdfs:subClassOf>
20       <owl:Restriction>
21         <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLS-
chema#int"
22         >1</owl:cardinality>
23         <owl:onProperty>
24           <owl:ObjectProperty rdf:ID="hatMainboard"/>
25         </owl:onProperty>
26       </owl:Restriction>
27     </rdfs:subClassOf>
28   </owl:Class>
```

Nun können wir die Computerinstanz „HeimPC“ [0] mit den Eigenschaften hatCPU [2], hatMainboard [2] und hatGrafikkarte [3] beschreiben.

```
0<Computer rdf:ID="HeimPC">
1     <hatCPU rdf:resource="#Intel3Ghz"/>
2     <hatMainboard rdf:resource="#AsusA8N"/>
3     <hatGrafikkarte rdf:resource="#GeForce6600GT"/>
4   </Computer>
```

Unsere (fertige) Computerontologie sieht folgendermaßen aus:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns="http://www.meineono.at/pc.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
 xml:base="http://www.meineono.at/pc.owl">
 <owl:Ontology rdf:about=""/>



    <owl:Class rdf:ID="Systemkomponenten"/>
```

```
   <owl:Class rdf:ID="Mainboard">
   <rdfs:subClassOf rdf:resource="#Systemkomponenten"/>
 </owl:Class>
 <owl:Class rdf:ID="CPU">
   <rdfs:subClassOf rdf:resource="#Systemkomponenten"/>
 </owl:Class>
 <owl:Class rdf:ID="Grafikkarte">
   <rdfs:subClassOf rdf:resource="#Systemkomponenten"/>
 </owl:Class>
 <owl:Class rdf:ID="Computer">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="hatCPU"/>
       </owl:onProperty>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLS-
chema#int"
       >1</owl:cardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLS-
chema#int"
       >1</owl:cardinality>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="hatGrafikkarte"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>

   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLS-
chema#int"
       >1</owl:cardinality>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="hatMainboard"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
 </owl:Class>
 <owl:ObjectProperty rdf:about="#hatCPU">
   <rdfs:domain rdf:resource="#Computer"/>
   <rdfs:range rdf:resource="#CPU"/>
```

```
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:about="#hatGrafikkarte">
   <rdfs:domain rdf:resource="#Computer"/>
   <rdfs:range rdf:resource="#Grafikkarte"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:about="#hatMainboard">
   <rdfs:range rdf:resource="#Mainboard"/>
   <rdfs:domain rdf:resource="#Computer"/>
 </owl:ObjectProperty>

     <Grafikkarte rdf:ID="GeForce6600GT"/>
     <CPU rdf:ID="Intel3Ghz"/>
     <Mainboard rdf:ID="AsusA8N"/>

 <Computer rdf:ID="HeimPC">
   <hatCPU rdf:resource="#Intel3Ghz"/>
   <hatMainboard rdf:resource="#AsusA8N"/>
   <hatGrafikkarte rdf:resource="#GeForce6600GT"/>
 </Computer>
</rdf:RDF>
```

Es handelt sich hierbei um eine OWL Lite Ontologie.

Nehmen wir nun an, dass wir unsere Ontologie dahin erweitern, das nun auch Computer (z.B.: Server), welche mit zwei Prozessoren betrieben werden, unterstützt werden. Dadurch würde die Restriktion das ein Computer genau eine CPU hat durch

```
<owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</owl:maxCardin-
ality>
```

ersetzt werden. Das hat zur Folge, dass die Ontologie nun eine OWL DL Ontologie ist, da bei OWL Lite die Kardialitäten auf „0“ und „1“ beschränkt sind.

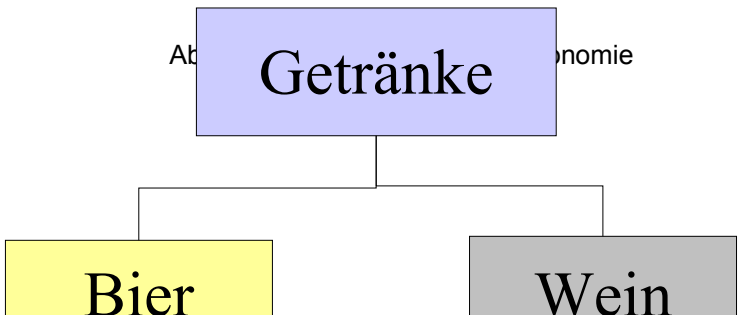
6.9 Taxonomie vs. Ontologie

Der Unterschied zwischen Ontologien und Taxonomien liegt darin, dass Taxonomien Objekte lediglich hierarchisch klassifizieren. Ontologien bieten neben

der reinen hierarchischen Klassifikation der Objekte auch die Möglichkeit, die Beziehungen, Restriktionen und Eigenschaften der Objekte darzustellen. Als Beispiel wird folgend eine Taxonomie für Getränke, die sich in Bier und Wein und Wein wiederum in Weißwein und Rotwein gliedert dargestellt.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.newsmuster.org/getreanke.owl#"
  xml:base="http://www.newsmuster.org/getreanke.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Bier">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Getränke"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Wein">
    <rdfs:subClassOf rdf:resource="#Getränke"/>
  </owl:Class>
  <owl:Class rdf:ID="Rotwein">
    <rdfs:subClassOf rdf:resource="#Wein"/>
  </owl:Class>
  <owl:Class rdf:ID="Weisswein">
    <rdfs:subClassOf rdf:resource="#Wein"/>
  </owl:Class>
</rdf:RDF>
```

Daraus lässt sich folgender Taxonomiebaum ableiten:



Diese einfache Taxonomie kann nun zu einer Ontologie dadurch erweitert werden, dass die Restriktion eingeführt wird, dass Weisswein besonders gut zu Fischgerichten schmeckt. In der nachfolgenden Ontologie wird dies dargestellt. „Fischgerichte“ ist eine Subklasse von „Essen“ und die Restriktion „passtzu-Fisch“ gibt an, dass nur Weissweine zu Fisch schmecken. Als Beispiel wird die Fischgerichte Instanz „Forelle“ instanziert, die als passenden Wein „Vetliner“ zugeordnet bekommt.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.newsmuster.org/wein.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.newsmuster.org/wein.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Bier">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Getränke"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Wein">
    <rdfs:subClassOf rdf:resource="#Getränke"/>
  </owl:Class>
  <owl:Class rdf:ID="Fischgerichte">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Essen"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XML-
Schema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="passtzuFisch"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Rotwein">
    <rdfs:subClassOf rdf:resource="#Wein"/>
  </owl:Class>
```

```
  <owl:Class rdf:ID="Weisswein">
    <rdfs:subClassOf rdf:resource="#Wein"/>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#passtzuFisch">
    <rdfs:range rdf:resource="#Weisswein"/>
  </owl:ObjectProperty>
  <Weisswein rdf:ID="Vetliner"/>
  <Fischgerichte rdf:ID="Forelle">
    <passtzuFisch rdf:resource="#Vetliner"/>
  </Fischgerichte>
</rdf:RDF>
```

6.10 OWL Tools

Zur Validierung und zur Feststellung der Zugehörigkeit zu den einzelnen Subsprachen werden im Internet Online – Validatoren angeboten. Exemplarisch sei hier der WonderWeb OWL Ontology Validator genannt, der unter der URL: „http://phoebus.cs.man.ac.uk:9999/OWL/Validator“ aufrufbar ist.

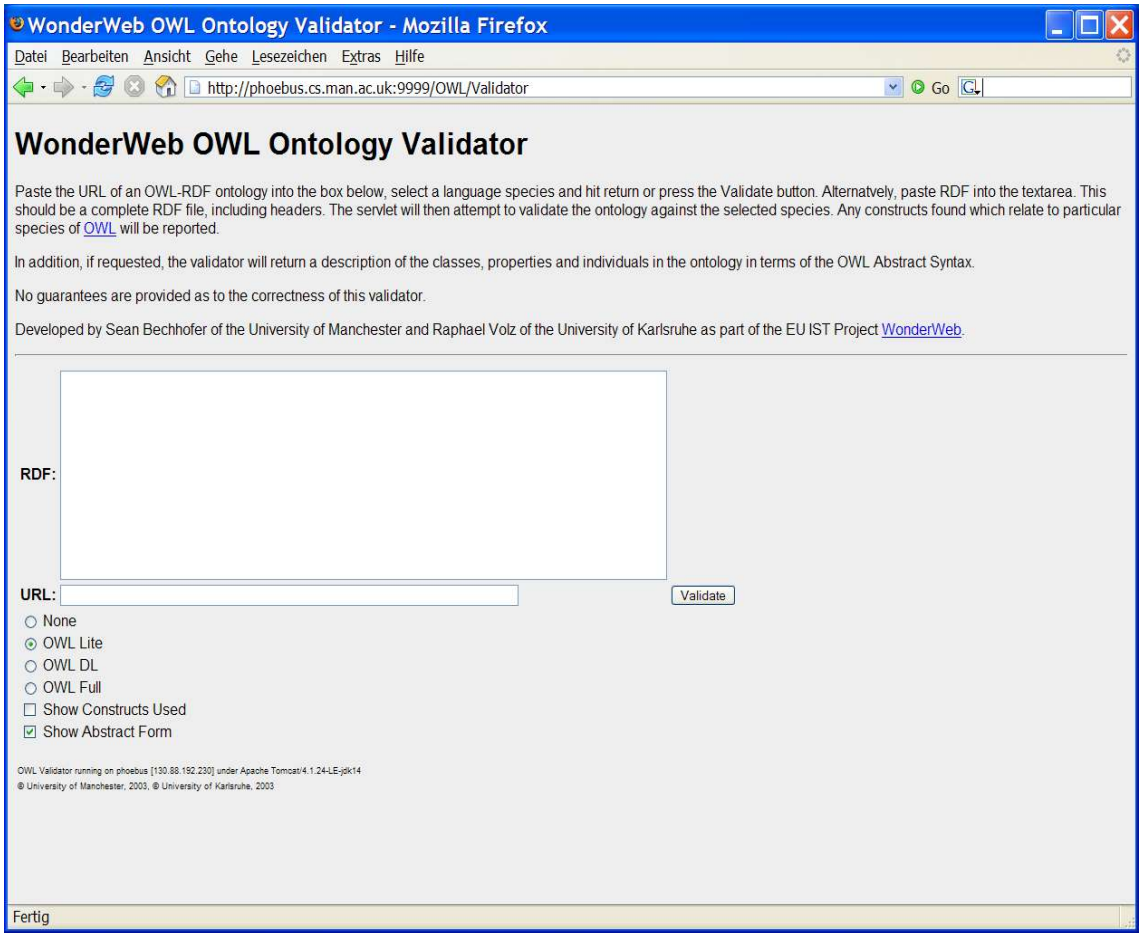


Abbildung 27: Sreenshot des WonderWeb OWL Ontology Validator [Vali]

Zur Erstellung von Ontologien gibt es auch Hilfstools. Hierbei sei vor allem Protégé als Open Source Ontologie Editor erwähnt. Dieser wird unter der URL „http://protege.stanford.edu/“ zum freien Download angeboten.

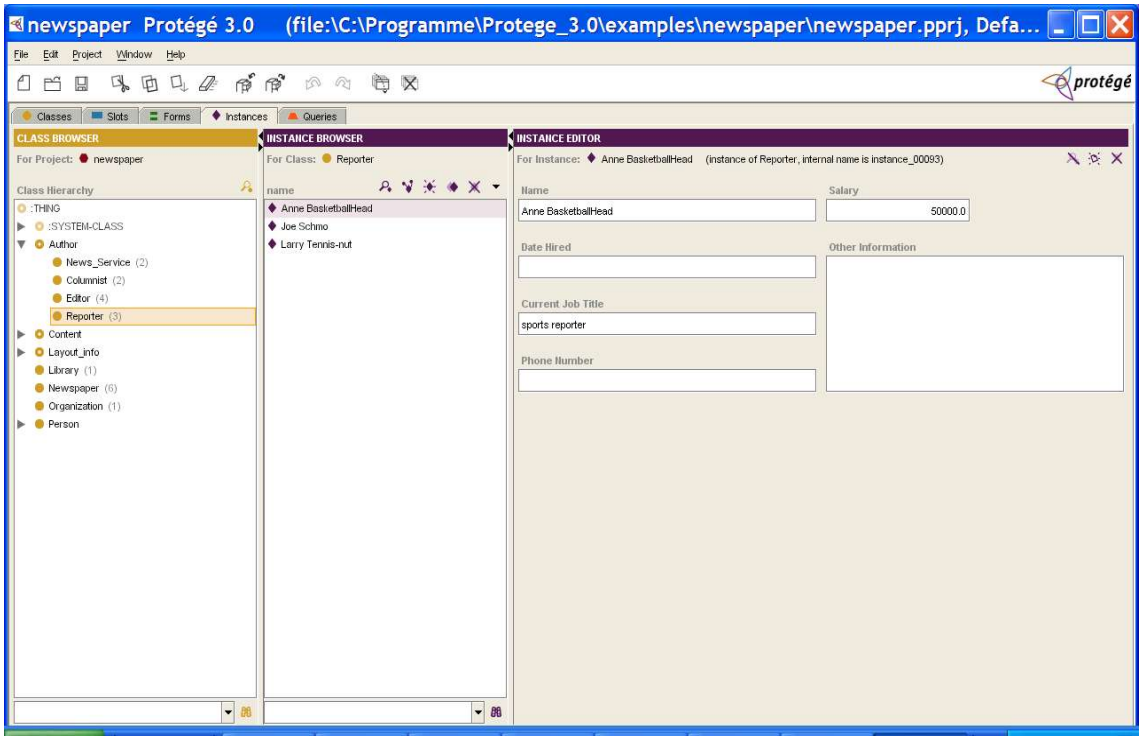


Abbildung 28: Screenshot von Protégé 3.0 [Prot]

6.11 Ontologien in der Praxis

Obwohl OWL in der Praxis noch wenig Verbreitung gefunden hat, sind sich doch viele EDV Entwicklungsverantwortliche von bedeutenden Firmen wie Boeing, HP, IBM Research, Nokia, Sun usw. einig, dass OWL ein bedeutsamer und für die Zukunft sehr wichtiger Standard ist [W3Cf]. Der geringe Einsatz von OWL mag auch daran liegen, da oft die Möglichkeiten von RDF noch nicht zur Gänze ausgreizt sind, und ein Umstieg auf OWL erst in einiger Zeit nötig sein wird.

Eine der interessantesten Semantic Web Anwendungen ist zurzeit die Firefox Extension „Piggy Bank“. Mittels „Piggy Bank ist es bereits „heute“ möglich einen

„semantischen Browser“ zu verwenden. Dazu werden die bereits bestehenden Informationen ausgelesen und mittels RDF gespeichert. Danach ist es möglich die Daten der verschiedensten Websites sinnvoll miteinander zu kombinieren. So kann zum Beispiel die Adresse einer Firma durch die Kombination mit einer „Planseite“ auf einer Landkarte dargestellt werden, obwohl dies auf der ursprünglichen Firmenseite nicht möglich war. Leider ist die Anwendung von „Piggy Bank“ derzeit nur auf wenigen Webseiten möglich. Beispielhaft sei die Anwendung von „Piggy Bank“ an der Informationsseite http://www.orf.at/ gezeigt.

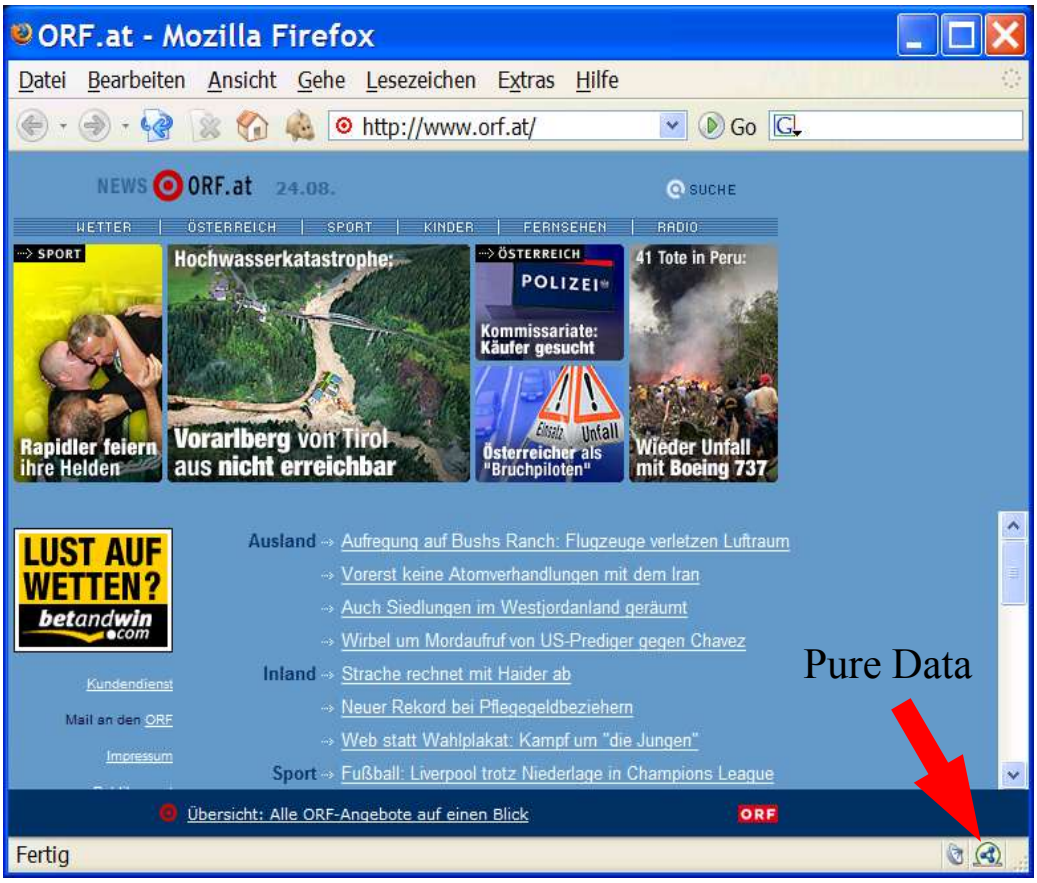


Abbildung 29: Screenshot von http://www.orf.at/ vom 24. August 2005

Durch einen „Klick“ auf das „Pure Data“ Symbol werden die Daten neu aufbereitet und als „reine Information“ (Pure Data) dargestellt. Danach kann die Seite nach diversen Kriterien gruppiert und untersucht werden, was die Informationsgewinnung für den Benützer vereinfacht. Unter der URL http://simile.mit.edu/piggy-bank/ kann die Firefox - Extension „Piggy Bank“ heruntergeladen werden.

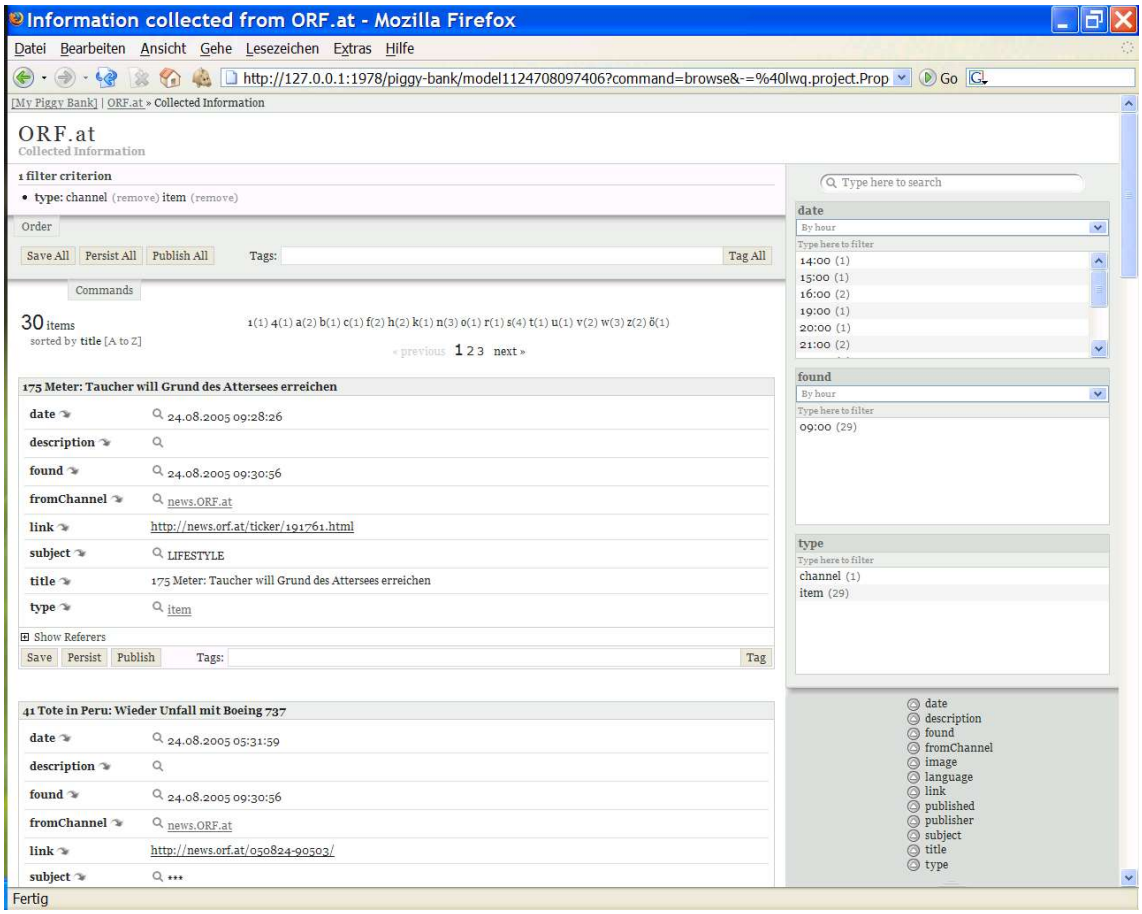


Abbildung 30: Screenshot von http://www.orf.at/ , „Pure Data“ Ansicht vom 24. August 2005

6.12Exkurs: Beschreibungslogik

Beschreibungslogiken sind Sprachen zur Repräsentation von Wissen. Wie der Name “Beschreibungslogik” bereits sagt, handelt es sich einerseits um Konzepte um eine Sache zu beschreiben, und andererseits um eine auf Logik basierende Semantik. Beschreibungslogiken sind eine Untermenge der Prädikatenlogik, die wiederrum eine Untermenge der Logik ist. Mittels Prädikatenlogik können einfache logische Aussagen wie zB. „Alle Weißweine schmecken gut zu Fisch“ oder „Die Forelle ist ein Fisch“ und logische Schlussfolgerungen, in unserem Beispiel „Weißwein schmeckt zu Forelle“, getroffen werden [Nard02] [Pann04].

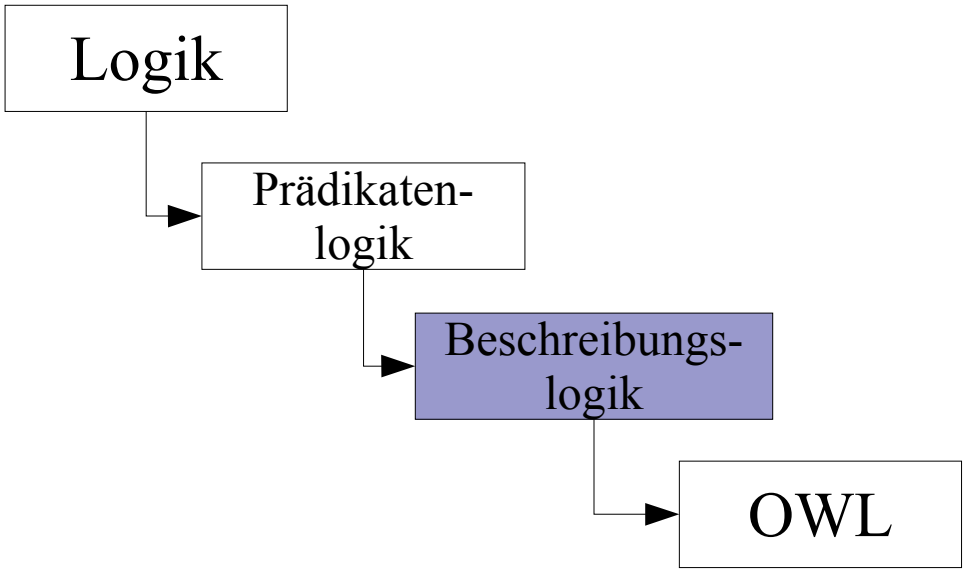


Abbildung 31: Einordnung der Beschreibungslogik

7Ausblick

Vier Jahre sind vergangen seit Tim Berners Lee's Artikel im „Scientific American“ veröffentlicht wurde. In dieser Zeit ist viel passiert und vieles hat sich verändert. Auf der jährlich stattfindenden International Semantic Web Conference (ISWC) werden Ideen, Technologien und Probleme, kurz, verschiedenste Aspekte, die mit dem Semantic Web in Verbindung stehen (oder stehen könnten / sollten) vorgestellt, diskutiert und ausgetauscht. An dieser Stelle möchten die Autoren einige davon herausgreifen.

Die vorgestellte Schichteneinteilung ist, zumindest was die oberen Schichten (Logic bis Trust) betrifft, bereits nicht mehr aktuell. Warum dennoch das ursprüngliche Schichtenmodell als Grundlage für diese Unterlagen herangezogen wurde, ist schnell erklärt: die Autoren wollten den Lesern die ursprüngliche Idee und Konzeption des Semantic Web näher bringen und einen Einblick in die grundlegende Funktionsweise der zur Umsetzung angedachten Technologien bieten. Dabei sollte eine Wertung weitgehend vermieden und dem Leser selbst überlassen werden. Diese Unterlagen sind gewissermaßen als „Sprungbrett“ für eine weitere Auseinandersetzung mit dem Thema gedacht.

Bereits bei der zweiten Schicht (XML) beginnen sich manche Geister zu scheiden. Andere Technologien sind im Gespräch, die andere Schichten inkludieren könnten. Spätestens auf der Ebene der Ontologien jedoch teilen sich die Ansichten nicht nur auf der technologischen, sondern auf der konzeptionellen Ebene, von den darüber liegenden ganz zu schweigen.

Drei Technologien die in Zusammenhang mit dem Semantic Web stehen und neue Aspekte einbringen sollen hier kurz umrissen werden:

SPARQL (Protocol And RDF Query Language)

Um die in RDF verfügbare Information besser nutzbar zu machen, soll eine standardisierte Abfragesprache entwickelt werden, die Information aus den RDF-Graphen ausliest. Stand: 3. Public Working Draft (19. April 2005) (http://www.w3.org/TR/rdf-sparql-query/ ;letzter Abruf am 2005-06-23)

XHTML 2.0

Um semantische Aspekte und neue Technologien besser einbinden zu können, wird XHTML 2.0 einige Neuerungen und Änderungen gegenüber seinem Vor-

gänger beeinhalten: das Abgehen von der strikten Abwärtskompatibilität und der Vorsatz weniger layout- und mehr strukturorientiert zu sein, sind nur zwei von mehreren wichtigen Punkten. Stand: 7. Public Working Draft (27. Mai 2005) (http://www.w3.org/TR/xhtml2/ ; letzter Abruf am 2005-06-23)

VoiceXML 2.1

VoiceXML will unterschiedliche Signale nutzbar machen, um damit Vorteile des WWW in neuen Bereichen nutzbar zu machen. Die angedachte Palette ist groß: von der Navigation durch eine Seite mittels Tastenton (per Telefon) bis hin zur Aufnahme von gesprochenen Worten, könnten gänzlich neue Anwendungen für diese Technologie entstehen. Stand: Candidate Recommendation (13. Juni 2005). (http://www.w3.org/TR/2005/CR-voicexml21-20050613/ ; letzter Abruf am 2005-06-23)

Diese drei Beispiele zeigen, dass sich das Internet, mehr oder weniger offensichtlich, weiter verändern wird. Ob und in welcher Weise die Semantik verstärkt einfließen wird, bleibt abzuwarten.

8Übungsbeispiele

8.1XML

Beispiel 1: Finden Sie den Fehler in diesem Beispiel:

```
<autor name=Maier>
```

Beispiel 2: Finden Sie die Fehler in diesem Beispiel:

```
<artikel>
  <titel>Meteorologen prognostizieren Rekordsommer</titel>
  <autor>Wetterfrosch</autor>
  <datum>14. Mai 2005 <datum>
</Artikel>
```

Beispiel 3: Erstellen Sie eine XML-Datei, die zu folgender Beschreibung passt:

Die Datei „Kontakte.xml“ soll mindestens drei Einträge aufweisen. Jeder Eintrag soll den Namen, den Vornamen, das Alter und die e-mail-Adresse umfassen.

Beispiel 4: Erstellen Sie zu folgender XML-Datei eine passende Schema-Datei:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<autoren xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:no-
NamespaceSchemaLocation="BSP4.xsd">
 <autor>
  <name>Maier</name>
  <vorname>Hans</vorname>
  <alter>38</alter>
  <berufserfahrung>5 Jahre</berufserfahrung>
 </autor>
 <autor>
  <name>Mayer</name>
  <vorname>Max</vorname>
  <alter>47</alter>
  <berufserfahrung>25</berufserfahrung>
  <wichtigeWerke>XML</wichtigeWerke>
 </autor>
</autoren>
```

Beispiel 5: Erstellen Sie zu folgender Schema-Datei eine passende XML-Datei:

```
?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="schlagzeilen">
  <xsd:annotation>
   <xsd:documentation>Schema-Datei zu Beispiel 5</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="schlagzeile" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="titel">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="55"/>
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="redakteur" type="xsd:string"/>
       <xsd:element name="datum" type="xsd:date"/>
       <xsd:element name="bereich">
        <xsd:simpleType>
         <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="politik"/>
          <xsd:enumeration value="kultur"/>
          <xsd:enumeration value="sport"/>
          <xsd:enumeration value="wetter"/>
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="anmerkung" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Beispiel 6: Erstellen Sie eine einfache XSL-Transformation, die eine HTML-Tabelle ausgibt, die folgende Informationen enthält: Schlagzeile, Bereich und Datum (Beziehen Sie sich auf Beispiel 5).

Beispiel 7: Erstellen Sie zu folgender XML-Datei eine Schema-Datei, die die Auswahl an Städten auf die neun österreichischen Hauptstädte beschränkt und die zulässt, dass kein Wert für das Element „himmel“ angegeben wird. Erstellen Sie danach eine XSL-Datei, die alles in einer Tabelle ausgibt (leere Felder sind erlaubt).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="wetter.xsl"?>
<messungen xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wetter.xsd">
  <wert>
    <stadt>Eisenstadt</stadt>
    <temp>33</temp>
    <datum>2005-06-24</datum>
  </wert>
  <wert>
    <stadt>Wien</stadt>
    <himmel>Heiter</himmel>
    <temp>32</temp>
    <datum>2005-06-24</datum>
  </wert>
  <wert>
    <stadt>Inssbruck</stadt>
    <himmel>Stark bewölkt</himmel>
    <temp>17</temp>
    <datum>2005-06-24</datum>
  </wert>
  <wert>
    <stadt>Graz</stadt>
    <temp>26</temp>
    <datum>2005-06-24</datum>
  </wert>
</messungen>
```

Zusatzbeispiel:

a)

Erstellen Sie eine XML-Datei mit dem Namen student.xml. in dieser Datei sollen Daten über Studenten gespeichert werden können. Schreiben Sie eine DTD und fügen sie 3 Datensätze Ihrer Wahl ein.

Die Datei soll folgendes Aussehen haben: Einem Student soll einen Vorname, einen Nachname, eine Matrikelnummer und ein Studium zugeordnet werden können. Zusätzlich soll dem Studium ein Attribut mit dem Namen Art zugeordnet werden, welches beispielsweise die Werte „bakk“, „mag“ oder „dr“ haben könnte, je nachdem um welchen Studientyp es sich handelt.
Hinweis: für Attribute verwenden Sie folgende Syntax:
In der DTD: <!ATTLIST studium art CDATA #IMPLIED>
In der XML-Datei: <studium art ="mag">IBW</studium>

b)

Erstellen sie eine Datei student2.xml, die der Datei Student.xml gleicht, aber keine DTD enthält. Erstellen Sie stattdessen eine passende XML-Schema-Datei. (student2.xsd)Berücksichtigen Sie auch die Datentypen, beispielsweise kann eine Matrikelnummer nur eine Zahl sein (z.B. Integer). Fügen Sie ein Feld für das Geburtsdatum hinzu.

Hinweis: Attribute in XML-Schema: <xs:attribute name="art" type="xsd:string"/>

8.2RDF

Beispiel 1:

In die Aussage aus soll ein weiterer Aspekt hinzugefügt werden. Stellen Sie folgende Aussage grafisch dar: „Die Website www.newsmuster.org hat als Ersteller die Seminargruppe, die aus Hans, Karl und Rudi besteht.“

Beispiel 2:

Erweitern Sie das Beispiel 1 insofern, dass die anonyme Ressource durch eine eindeutige URI identifiziert werden kann.

Beispiel 3:

Erstellen sie die XML-Notation zur Container-Darstellung aus Abbildung 14.

Beispiel 4:

Erstellen Sie die Syntax zur folgenden Abbildung, die ein Datenmodell einer Aussage über eine Aussage darstellt.

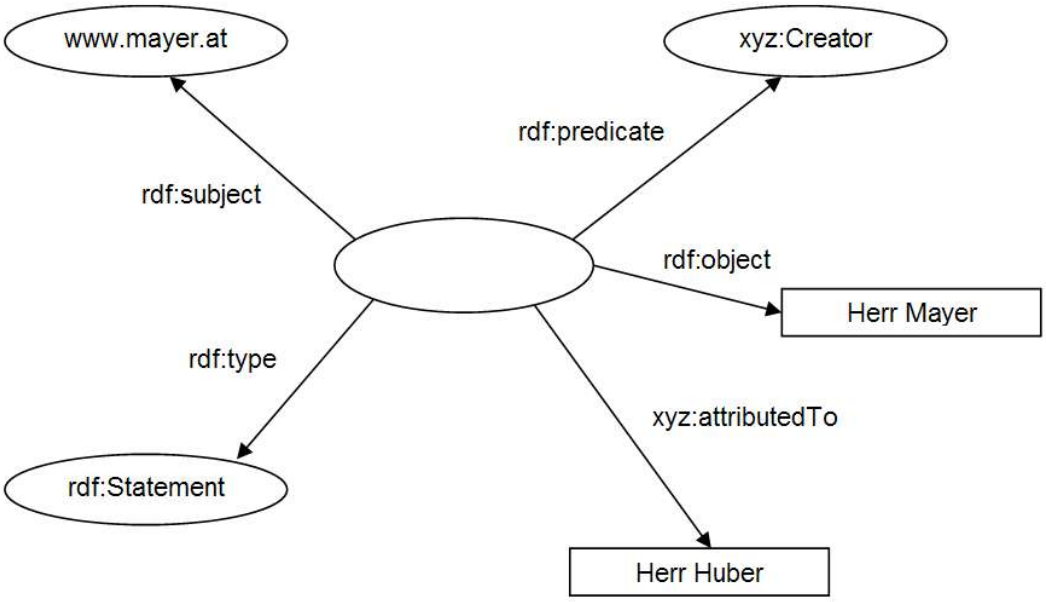


Abbildung 32: Datenmodell einer Aussage über eine Aussage

Beispiel 5:

Erstellen Sie eine Datei, die den Begriff „XML“ als Klasse und die Begriffe „XML-Schema“ und „RDF“ als Unterklassen von „XML“ definiert.

Beispiel 6:

Erstellen Sie eine RSS-Datei, die unterschiedliche Nachrichtenmeldungen anbietet. Beziehen Sie sich dabei auf die fiktiv erstelle Website http://www.-newsmuster.org.

Zusatzbeispiel:

Erstellen Sie eine RDF-Datei, die die Webseite http://www.newsmuster.org/sport/meisterschaftsende.html beschreibt. Beschreiben Sie den Autor mittels Website, Name und Email. Verwenden Sie dabei das fiktive Schema xyz.

8.3OWL

Beispiel 1: Definieren sie die Klasse „Bälle“ mit den Subklassen „Fussball“ und „Handball“ sowie die Klasse „Leder“ mit den Instanzen „Naturleder“ und „Kunstleder“. (Nehmen sie Namespace der Ontologie „http://www.beispiel.at/ball.owl“ an.

Beispiel 2: Nun erweitern sie die Ontologie um die Eigenschaft „istaus“ (owl:ObjectProperty), dass ein Fussball aus Leder sein muss. (verwenden sie dazu die Restriktion „owl:hasValue“).

Beispiel 3: Instanzieren sie nun von der Klasse “Fussball” den “EMStar2008” der aus „Naturleder ist.

9Lösungen

9.1XML

Beispiel 1:

```
<autor name=“Maier“>
```

Bei Attributen muss der Wert immer in Anführungszeichen oder Hochkommas stehen.

Beispiel 2:

```
<artikel>
  <titel>Meteorologen prognostizieren Rekordsommer</titel>
  <autor>Wetterfrosch</autor>
  <datum>14. Mai 2005 </datum>
</Artikel>
```

1. Der Slash beim Schließ-Tag wird bei manueller Eingabe häufig vergessen.
2. XML ist case-sensitive. D.h.: artikel ≠ Artikel

Beispiel 3:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<kontakte>
  <eintrag>
    <name>Iksemel</name>
    <vorname>Detede</vorname>
    <alter>5</alter>
    <email>beispiel@test.com</email>
  </eintrag>
  <eintrag>
    <name>Mustermann</name>
    <vorname>Helga</vorname>
    <alter>29</alter>
    <email>muster@muster.com</email>
  </eintrag>
  <eintrag>
    <name>Altmutter</name>
    <vorname>Thomas</vorname>
    <alter>26</alter>
    <email>h0000000@wu-wien.ac.at</email>
  </eintrag>
</kontakte>
```

Beispiel 4:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="autoren">
    <xsd:annotation>
      <xsd:documentation>Schema-Datei zu Beispiel 4</xsd:documentati-
      on>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="autor" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
              <xsd:element name="name" type="xsd:string"/>
              <xsd:element name="vorname" type="xsd:string"/>
              <xsd:element name="alter" type="xsd:int"/>
              <xsd:element name="berufserfahrung" type="xsd:string"/>
              <xsd:element name="wichtigeWerke" type="xsd:string"
               minOccurs="0"/>
              </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Beispiel 5:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<schlagzeilen xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="F:\Eigene Dateien\Studium\SoSe05\SemWe-
bXML\BSP5.xsd">
      <schlagzeile>
            <titel>Heißer Sommer erwartet</titel>
            <redakteur>Muster</redakteur>
            <datum>2005-03-23</datum>
            <bereich>wetter</bereich>
      </schlagzeile>
      <schlagzeile>
            <titel>Vienale eröffnet</titel>
            <redakteur>Vorlage</redakteur>
            <datum>2005-04-15</datum>
            <bereich>kultur</bereich>
      </schlagzeile>
</schlagzeilen>
```

Beispiel 6:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"/>
 <xsl:template match="/">
   <html>
     <head><title>Transformation</title></head>
     <body>
       <table border="1">
         <tr>
           <td>Schlagzeile</td>
           <td>Bereich</td>
           <td>Datum</td>
         </tr>
         <xsl:for-each select="schlagzeilen/schlagzeile">
           <tr>
             <td><xsl:value-of select="titel"/></td>
             <td><xsl:value-of select="datum"/></td>
             <td><xsl:value-of select="bereich"/></td>
           </tr>
         </xsl:for-each>
       </table>
     </body>
   </html>
 </xsl:template>
</xsl:stylesheet>
```

Beispiel 7: Die Schema-Datei:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="messungen">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="wert" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="stadt">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="Wien"/>
                    <xsd:enumeration value="St.Pölten"/>
                    <xsd:enumeration value="Eisenstadt"/>
```

```
                        <xsd:enumeration value="Linz"/>
                        <xsd:enumeration value="Graz"/>
                        <xsd:enumeration value="Inssbruck"/>
                        <xsd:enumeration value="Klagenfurt"/>
                        <xsd:enumeration value="Salzburg"/>
                        <xsd:enumeration value="Bregenz"/>
                      </xsd:restriction>
                    </xsd:simpleType>
                  </xsd:element>
                  <xsd:element name="himmel" type="xsd:string"
                  minOccurs="0"/>
                  <xsd:element name="temp" type="xsd:integer"/>
                  <xsd:element name="datum" type="xsd:date"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Die XSL-Datei:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
      <html>
      <head><title>Wetterwerte</title></head>
      <body>
       <table border="2">
  <tr>
   <td>Stadt</td>
   <td>Wetter</td>
   <td>Temperatur</td>
   <td>Datum</td>
  </tr>
      <xsl:for-each select="messungen/wert">
   <tr>
    <td><xsl:value-of select="stadt"/></td>
    <td><xsl:value-of select="himmel"/></td>
    <td><xsl:value-of select="temp"/></td>
    <td><xsl:value-of select="datum"/></td>
   </tr>
  </xsl:for-each>
```

```
 </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Zusatzbeispiel:

a)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE studenten [
  <!ELEMENT studenten (student)+>
  <!ELEMENT student (vorname, nachname, adresse)>
  <!ELEMENT vorname (#PCDATA)>
  <!ELEMENT nachname (#PCDATA)>
  <!ELEMENT matrikelnummer (#PCDATA)>
  <!ELEMENT studium (#PCDATA)>
  <!ATTLIST studium art CDATA #IMPLIED>
]>
<studenten>
  <student>
    <vorname>Lukas</vorname>
    <nachname>Helm</nachname>
    <matrikelnummer>0253563</matrikelnummer>
    <studium art ="bakk">WINF</studium>
  </student>
  <student>
    <vorname>Thomas</vorname>
    <nachname>Altmutter</nachname>
    <matrikelnummer>0053410</matrikelnummer>
    <studium art ="mag">IBW</studium>
  </student>
  <student>
    <vorname>Dieter</vorname>
    <nachname>Steffen</nachname>
    <matrikelnummer>9953501</matrikelnummer>
    <studium art="dr">BW</studium>
  </student>
</studenten>
```

b)

student2.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<studenten xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student2.xsd">
  <student>
    <vorname>Lukas</vorname>
    <nachname>Helm</nachname>
    <matrikelnummer>0253563</matrikelnummer>
    <studium art ="bakk">WINF</studium>
  </student>
  <student>
    <vorname>Thomas</vorname>
    <nachname>Altmutter</nachname>
    <matrikelnummer>0053410</matrikelnummer>
    <studium art ="mag">IBW</studium>
  </student>
  <student>
    <vorname>Dieter</vorname>
    <nachname>Steffen</nachname>
    <matrikelnummer>0251281</matrikelnummer>
    <studium art="dr">BW</studium>
  </student>
</studenten>
```

student2.xsd:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="studenten">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="student" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="vorname" type="xsd:string" />
                <xsd:element name="nachname" type="xsd:string" />
                <xsd:element name="matrikelnummer" type="xsd:int" />
                <xsd:element name="studium" type="xsd:string" />
                   <xs:complexType>
                     <xs:attribute name="art" type="xsd:string"/>
                   </xs:complexType>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
```

9.2RDF

Beispiel 1:

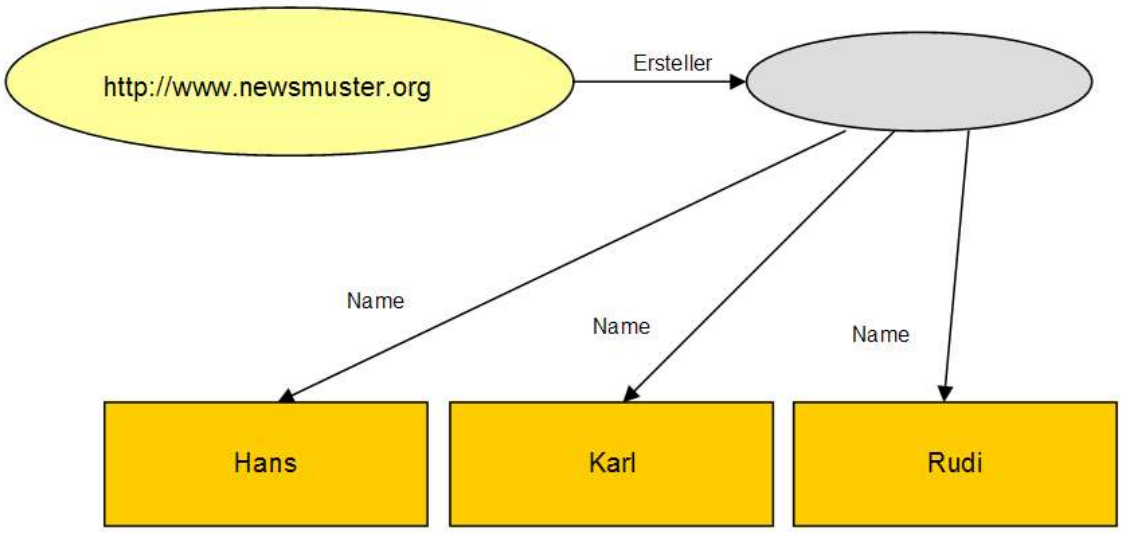


Abbildung 33: Lösung zu, RDF-Beispiel 1

Beispiel 2:

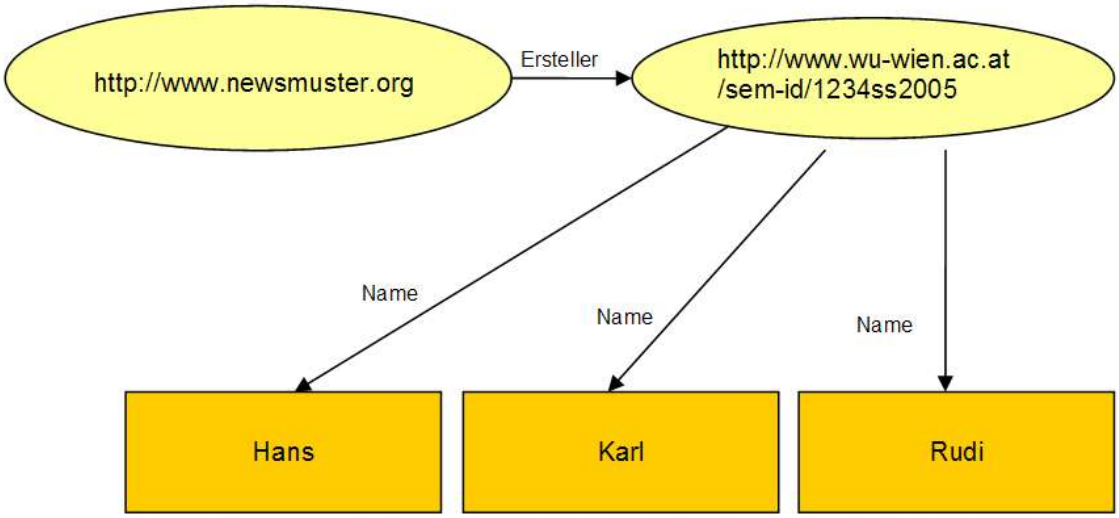


Abbildung 34: Lösung zum RDF-Beispiel 2

Die Aussage ist so zu lesen: „Die Mitglieder der Seminargruppe mit der Identifikationsnummer 1234ss2005 haben den Namen Hans, Karl und Rudi. Die Ressource http://newsmuster.org wurde von diesen Personen erstellt."

Beispiel 3:

```
<rdf:Description about="ISBN0-471-89275-0">
   <xyz:author>
      <rdf:Sequence>
         <rdf:li resource="http://buch.com/longley" />
         <rdf:li resource="http:// buch.com/goodchild" />
      </rdf:Sequence>
   </xyz:author>
</rdf:Description>
```

Beispiel 4:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xyz="http://www.mayer.at/rdfschema#">
   <rdf:Description>
      <rdf:subject resource="http://www.mayer.at"/marketing />
      <rdf:predicate
           resource="http:// www.mayer.at/rdfschema#Creator" />
      <rdf:object>Herr Mayer</rdf:object>
      <rdf:type
```

```
    resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement" />
      <xyz:attributedTo>Herr Huber</xyz:attributedTo>
   </rdf:Description>
</rdf:RDF>
```

Beispiel 5:

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

<rdf:Description ID=”XML”>
    <rdf:type resource=”http://www.w3.org/2000/01/rdf-schema#Class”/>
    <rdfs:subClassOf rdf:resource=”http://www.w3.org/2000/01/
                                                rdf-schema#Resource”/>
</rdf:Description>

<rdf:Description ID=”XML-Schema”>
    <rdf:type resource=”http://www.w3.org/2000/01/rdf-schema#Class/”>
    <rdfs:subClassOf rdf:resource=”#XML”/>
</rdf:Description>

<rdf:Description ID=”RDF”>
    <rdf:type resource=”http://www.w3.org/2000/01/rdf-schema#Class”/>
    <rdfs:subClassOf rdf:resource=”#XML”/>
</rdf:Description>
</rdf>

[Ecks04, S. 260]
```

Innerhalb des RDF-Elements, das zwei Namensraum-Deklarationen für die Namensräume von RDF/XML-Syntax und RDF-Schema enthält, werden drei Description-Elemente verwendet. Diese enthalten als Eigenschaftselemente *type* und *subClassOf*. Durch die Definitionen erhält man einen Oberbegriff „XML“, von dem die beiden untergeordneten Begriffe „XML-Schema“ und „RDF“ abgeleitet werden können. Dadurch wird es z.B. ermöglicht, dass bei der Suche bei Angabe von „XML“ auch Dokumente gefunden werden können, die nicht „XML“, dafür aber „XML-Schema“ oder „RDF“ enthalten [siehe Ecks04, S.260].

Beispiel 6:

```
<?xml version=”1.0” encoding=”utf-8”?>

<rdf:RDF xmlns=”http://purl.org/rss/1.0/”
    xmlns:dc=”http://purl.org/dc/elements/1.1/”
```

```
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <channel rdf:about="http://www.newsmuster.org/home.rss">
    <title>News Muster</title>
    <description>Die Seite, die vielseitige Nachrichten
anbietet...</description>
    <link>http://www.newsmuster.org/</link>

    <dc:date>2005-06-01T09:00:00Z</dc:date>

    <items>
        <rdf:Seq>
            <rdf:li
rdf:resource="http://www.newsmuster.org/2005/138.html" />
            <rdf:li rdf:resource="http://www.newsmuster.org/2005/139.-
html" />
            <rdf:li rdf:resource="http://www.newsmuster.org/2005/140.-
html" />
        </rdf:Seq>
    </items>

  </channel>

  <item rdf:about="http://www.newsmuster.org/2005/138.html">
    <title>Liverpool gewinnt die Champions League!</title>
    <description>25. Mai 2005: Liverpool besiegt den AC Milan im
    Elfmeterschießen […]
    </description>
    <link> http://www.newsmuster.org/2005/138.html </link>
    <dc:date>2005-05-25</dc:date>
  </item>

  <item rdf:about="http://www.newsmuster.org/2005/139.html">
    <title>Frankreich sagt Nein zur EU-Verfassung</title>
    <description>29. Mai 2005: Die Volksabstimmung zur EU-Verfassung
    brachte inFrankreich ein klares Nein […]</description>
    <link> http://www.newsmuster.org/2005/139.html </link>
    <dc:date>2005-05-29</dc:date>
  </item>

  <item rdf:about="http://www.newsmuster.org/2005/140.html">
    <title>Probleme bei Airbus A380</title>
    <description>1. Juni 2005: Nach der Klage der USA wegen der Sub-
ventionen kommt auf
    Airbus das nächste Problem zu […]</description>
```

```
    <link> http://www.newsmuster.org/2005/140.html </link>
    <dc:date>2005-06-01</dc:date>
  </item>

</rdf:RDF>
```

Zusatzbeispiel:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xyz="http://www.newsmuster.org/autorschema.xml">

  <rdf:Description
about=”http://www.newsmuster.org/sport/meisterschaftsende.html”>
    <xyz:Autor>
      <xyz:Website rdf:resource=http:// www.martinmuster.org />
      <xyz:Name>Martin Muster</xyz:Name>
      <xyz:Email>muster@newsmuster.org</xyz:Email>
    </xyz:Autor>
  </rdf:Description>
</rdf:RDF>
```

9.3OWL

Beispiel 1:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.beispiel.at/ball.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.beispiel.at/ball.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Bälle"/>
  <owl:Class rdf:ID="Handball">
    <rdfs:subClassOf rdf:resource="#Bälle"/>
  </owl:Class>
  <owl:Class rdf:ID="Fussball">
    <rdfs:subClassOf rdf:resource="#Bälle"/>
  </owl:Class>
  <owl:Class rdf:ID="Leder"/>
```

```
  <Leder rdf:ID="Naturleder"/>
  <Leder rdf:ID="Kunstleder"/>
</rdf:RDF>
```

Beispiel 2:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.beispiel.at/ball.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.beispiel.at/ball.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Bälle"/>
  <owl:Class rdf:ID="Handball">
    <rdfs:subClassOf rdf:resource="#Bälle"/>
  </owl:Class>
  <owl:Class rdf:ID="Fussball">
    <rdfs:subClassOf rdf:resource="#Bälle"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:hasValue>
          <owl:Class rdf:ID="Leder"/>
        </owl:hasValue>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="istaus"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#istaus">
    <rdfs:range rdf:resource="#Leder"/>
    <rdfs:domain rdf:resource="#Fussball"/>
  </owl:ObjectProperty>
  <Leder rdf:ID="Naturleder"/>
  <Leder rdf:ID="Kunstleder"/>
</rdf:RDF>
```

Beispiel 3:

```
<Fussball rdf:ID="EMStar2008">
    <istaus rdf:resource="#Naturleder"/>
  </Fussball>
```

10 Quellenverzeichnis

[BeHL01] Berners-Lee, Tim; Hendler, James; Lassila, Ora: The Semantic Web; Scientific American, May 2001, pp. 35-42.

[Bern98] Berners-Lee, Tim: Semantic Web Road map; September 2004. http://www.w3.org/DesignIssues/Semantic.html, letzter Abruf am 2005-04-11.

[Case] Patrick Casey, Associated Press

[Born05] Born, Günter: jetzt lerne ich XML, Markt+Technik Verlag, München 2005.

[Choi03] Choi, Joongmin: Semantic Web Overview. Intelligent Systems Laboratory, Dept. of Computer Science and Engineering, Hanyang University, 2003. http://cse.hanyang.ac.kr/~jmchoi/presentations/Ajou-030226.pdf, letzter Abruf am 2005-05-09.

[Ecks04] Eckstein, Rainer; Eckstein, Silke, XML und Datenmodellierung, 2004

[Grub93] Gruber, T. R.; A Translation approach to portable ontology specifications; Knowledge Acquisition; 5: S. 199-220; 1993

[Grub95] Gruber, T. (1995). Towards Principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, (43):907–928.

[HaNe01] Hansen, Hans Robert; Neumann, Gustaf: Wirtschaftsinformatik I, 8. Auflage. Lucius & Lucius, Stuttgart 2001.

[Hüsl05] Hüsler, Martin: Kurzeinführung in XML. Fachhochschule Solothurn Nordwestschweiz, 2005. http://www.hst.fhso.ch/~huesler/internet/XML.pdf, letzter Abruf am 2005-04-18.

[Jeck03] Jeckle, Mario: XML-Namensräume. DaimlerChrysler Forschungszentrum Ulm, 2003. http://www.jeckle.de/files/xmlnsW3CDE.pdf, letzter Abruf am 2005-04-18.

[Jeck04] Jeckle, Mario: e-Business Engineering Vorlesung, 2004. http://www.jeckle.de/vorlesung/eBusinessEng/script.html, letzter Abruf am 2005-06-23.

[KnKo01] Knobloch, Manfred; Kopp, Matthias: Web-Design mit XML, dpunkt.verlag Heidelberg, 2001.

[Knop04] Knopper, Klaus: Einführung in die Informatik. http://www.knopper.-net/bw/gdi/vorlesung/vorlesung2.pdf, letzter Abruf am 2005-06-18.

[KoMi01] Koivunen, Marja-Riitta; Miller, Eric: W3C Semantic Web Activity. W3C 2001. http://www.w3.org/2001/12/semweb-fin/w3csw, letzter Abruf am 2005-04-11.

[Nard02] Nardi Daniele, Ronald J. Brachman, An Introduction to Description Logics 2002

[O.A.a] o.V.; http://www.ibiblio.org/pioneers/lee.html, letzter Abruf am 2005-03-30.

[Palm01] Palmer, Sean B.: The Semantic Web: An Introduction; http://in-fomesh.net/2001/swintro/, letzter Abruf am 2005-03-30.

[Pann04] Panny, Wolfgang: Expertensysteme und Uncertain Reasoning – Logik und AI. Wirtschaftsuniversität Wien, Abteilung für Informa-tionswirtschaft 2004.

[Peny05] Penya, Yoseba: Foliensätze 2 und 3, benutzt im „Grundzüge der In-formationsstrukturierung“-Semiar im SoSe 05 an der WU-Wien; htt-ps://learn.wu-wien.ac.at/ (Abruf am 2005-04-30; eingeschränkter Zugang).

[Popp05] Popp, Nora: Eine Einführung in das Semantic Web. Humboldt Uni-versität Informatik, Fachgebiet Wissensmanagement, Jänner 2005.

[Prot] Protégé Ontology Editor, http://protege.stanford.edu/ , letzter Abruf am 2005-06-18

[Schr04] Schreiber (2004) http://www.cs.vu.nl/~guus/public/2004-webont-zeist/all.htm , letzter Abruf am 2005-06-01

[Treib04] Treiblmaier, Horst: „XML“-Foliensatz, benutzt im „Webengineering 1“-Seminar im WS04/05 an der WU-Wien; https://learn.wu-wi-en.ac.at/ (Abruf am 2004-11-29; eingeschränkter Zugang)

[Unic05] Unicode Home Page: C0 Controls and Basic Latin. http://www.unicode.org/charts/PDF/U0000.pdf, letzter Abruf am 2005-06-18.

[Vali] OWL Validator http://phoebus.cs.man.ac.uk:9999/OWL/Validator, letzter Abruf am 2005-06-24.

[W3C04] World Wide Web Consortium: Der technische Hintergrund; 2004. http://www.w3c.at/Flyer/OnePage%202004.pdf, letzter Abruf am 2005-04-11.

[W3Ca] World Wide Web Consortium: Web Ontology Language (OWL). http://www.w3.org/2004/OWL/, letzter Abruf am 2005-05-08.

[W3Cb] World Wide Web Consortium: Web Ontology Language (OWL). http://www.w3.org/TR/2004/REC-webont-req-20040210/, letzter Abruf am 2005-05-20.

[W3Cc] World Wide Web Consortium: Web Ontology Language (OWL). http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.2, letzter Abruf am 2005-05-20.

[W3Cd] World Wide Web Consortium: Web Ontology Language (OWL). http://www.w3.org/TR/owl-features/ , letzter Abruf am 2005-05-20.

[W3Ce] World Wide Web Consortium: Wood, David; http://www.w3.org/2005/Talks/0512-semweb-dwood, letzter Abruf am 2005-08-20.

[W3Cf] World Wide Web Consortium: Testimonials for W3C's Semantic Web Recommendations - RDF and OWL, http://www.w3.org/2004/01/sws-testimonial , letzter Abruf am 2005-08-20.

[Weim05] Weimer, K ,SVG gibt Gas. Automobil-Bordnetzentwicklung mit XML, SVG, Apache Cocoon und der XML-Datenbank eXist. Automobile boardnet development with XML, SDVG, Apache Cocoon and the XML-database eXist

[Wels03] Welsh, Tom: The Semantic Web – Proof, Trust and Security. Cutter Consortium, 2003. http://www.cutter.com/research/2003/edge030923.html, letzter Abruf am 2005-05-09.

[Wiki a] Wikipedia: URI, URL. http://de.wikipedia.org/wiki/URL, letzter Abruf am 2005-04-11.

[Wiki b] Wikipedia: XML. http://de.wikipedia.org/wiki/XML, letzter Abruf am 2005-04-11.

[Wiki m] Wikipedia: RSS, http://de.wikipedia.org/wiki/RSS, letzter Abruf am 2005-06-07.

[wiki g] http://de.wikipedia.org/wiki/Kategorie:Graphentheorie, letzter Abruf am 2005-08-07.

[Wiki u] Wikipedia: URN, http://de.wikipedia.org/wiki/URN, letzter Abruf am 2005-08-07.

11 Abbildungsverzeichnis

12 Tabellenverzeichnis