Technische Fachhochschule Berlin
Studiengang Medieninformatik Online
Luxemburger Straße 10
13353 Berlin

WS 2004/05

Bachelor-Thesis

„SemanticFlash“ Flash-Interface für Semantic Web Anwendungen

Thomas Wollburg
Storkower Weg 2
15566 Schöneiche bei Berlin
tw@imaginers.de

Sommersemester 2005

Betreut durch Prof. Dr. Robert Strzebkowski

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Ersten Kontakt mit dem Semantic Web [1] hatte ich im Jahr 2001. Damals wusste ich nur bedingt, welches Ziel das Semantic Web hatte. Auf den ersten Blick erschien RDF [5] als ein unübersichtliches XML-Format. Nach einiger Einarbeitung wurde mir die Ziele des Semantic Web bewusst. Es geht um Daten, welche Aussagen über andere Daten machen. Es handelte sich also nicht um Dokumente im herkömmlichen Sinne, sondern um Ergänzungen von bestehenden Dokumenten. Diese Dokumente können Internetseiten, Bilder, Videos oder auch Tonaufnahmen sein.

Angesichts der immensen Informationsmenge, welche jeden Tag erzeugt wird, werden Wege und Lösungen zur Verwaltung dieser Informationen immer wichtiger. So wurden im Jahr 2003 jeden Tag rund 14.000.000 Gigabyte an Informationen erzeugt [4], das sind rund 3 Millionen DVDs - jeden Tag. Ein immer größer werdender Teil dieser Informationen ist über das Internet und vernetzte Computer erreichbar. Viele dieser Daten werden von kommerziellen und öffentlichen Institutionen erzeugt.

Jede dieser Institutionen verwendet ein eigenes System, um diese Daten zu speichern. Somit bildet jede dieser Institutionen eine eigene Domäne. Eine Veröffentlichung dieser Daten im Internet bringt dabei immer einen Informationsverlust in Form von strukturellen Verlusten mit sich. Beziehungen von Inhalten zueinander lassen sich in HTML nicht darstellen: es handelt sich immer um einfache Links ohne Richtung oder Wertung.

1.2 Ausgangssituation

Kommerzielle und nichtkommerzielle Institutionen verwalten viele Daten. Jede Institution speichert die Daten in der für sie jeweils besten Art und Weise und einem den verwendeten Applikationen angepassten Format. Immer mehr Daten werden in Form von XML gespeichert. XML ist zwar ein domänenübergreifend lesbares Format, erlaubt aber auch die Verwendung beliebiger Tags [1] um Inhalte auszuzeichnen, wie in Abbildung 1.1 zu sehen. An den gezeigten Beispielen wird auch schnell sichtbar, dass für den Menschen auf den ersten Blick ähnliche Daten für den Computer vollkommen verschiedene Dokumente sind, welche erst durch zusätzliche Arbeit in einem gemeinsamen Umfeld verwendet werden können.

[1] englisch „Tags“ = Markierungen, z.B. <autor>Max Mustermann</autor>.

```xml
<?xml version="1.0"?>
<CarParts>
    <Part>
        <RefNr>123 456</RefNr>
        <Titel>Mutter 32-80</Titel>
        <Kompats>
            <KomRef id="654 876" />
            <KomRef id="932 741" />
        </Kompats>
    </Part>
    <Part>
        <RefNr>123 4574</RefNr>
        <Titel>Mutter 32-90</Titel>
        <Kompats>
            <KomRef id="654 876" />
            <KomRef id="932 741" />
        </Kompats>
    </Part>
</CarParts>
```

```xml
<?xml version="1.0"?>
<Carparts>
    <Carpart>
        <Id>5456</Id>
        <Name>Mutter 60-140</Name>
        <Komteile>
            <Part id="654X" />
            <Part id="932C" />
        </Komteile>
    </Carpart>
    <Carpart>
        <Id>123D</v>
        <Name>Mutter 40-85</Name>
        <Komteile>
            <Part id="876X" />
            <Part id="741S" />
        </Komteile>
    </Carpart>
</CarParts>
```

Abbildung 1.1: XML-Beispiele

Durch diese Flexibilität wird ein domänenübergreifender Datenaustausch erschwert. Manchmal kann aber gerade ein solcher Datenaustausch zentrales Element der täglichen Arbeit sein, z.B. bei Herstellern, die von verschiedenen Zulieferern ähnliche Teile ordern. Im schlimmsten Fall hat jeder Lieferant eine eigene XML-Sprachvariante. Dadurch wird eine intelligente Suche in den Datenbeständen erheblich erschwert, da ja jeweils in einer speziellen Sprache gesucht werden muss. Die Texte sind zwar domänenübergreifend maschinenlesbar, jedoch können die Maschinen die Texte nicht verstehen. Hier setzt das Semantic Web an:

> *Das Semantische Web basiert auf der inhaltlichen Beschreibung digitaler Dokumente mit standardisierten Vokabularien, die eine maschinell verstehbare*

Semantik haben. Damit wird der Übergang von einem „Netz aus Verweisstrukturen“ zu einem „Netz aus Inhaltsstrukturen“ vollzogen. [2]

1.3 Die Idee: „SemanticFlash“

Das Semantic Web entwickelt sich zu einer ausgereiften Technologie. Zahlreiche Partnerschaften zwischen Bildungseinrichtungen und Unternehmen entwickeln Applikationen im Semantic-Web-Umfeld. Das Semantic Web ist schwer zu verstehen. Zahlreiche Fachbegriffe und der abstrakte Sachverhalt erschweren das Verstehen. Wohl auch deshalb ist der Einsatz von Semantic Web-Technologien bisher eher im professionellen Umfeld (Intranet) zu finden.

Auch ich hatte anfangs Probleme, das Semantic Web zu verstehen. Hier hat mir der RDF-Validator vom W3C [11] sehr weiter geholfen. Er ermöglicht eine Ausgabe von RDF in Graphen-Form. Diese grafische Darstellung der Tripel ist sehr anschaulich.

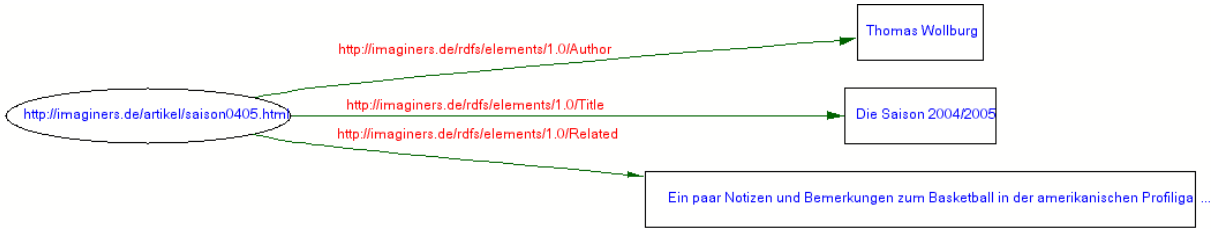

Abbildung 1.2: Graphendarstellung des W3C RDF-Validators

Diese Darstellung und das Wissen um die XML-Fähigkeiten von Flash MX haben die Idee einer Oberfläche zur RDF-Darstellung und Bearbeitung auf Flash-Basis hervor gebracht. Die allgemeine Verbreitung des Flash-Plugins und die plattformübergreifend einheitliche Funktionalität waren ebenfalls ausschlaggebend. Die auch mir persönlich immer wieder begegnenden Probleme mit Java-Applets, welche die wohl einzige technische Alternative darstellen, haben diese Möglichkeit als Umsetzungsalternative für die Idee ausgeschlossen. Das einfache Benutzen ohne zusätzliche Programme und somit im gewohnten Umfeld ist eines der Erfolgskriterien für die Idee.

Ziel der Idee ist es, eine Möglichkeit zu schaffen, das Semantic Web und/oder RDF auch dem Laien ohne die Installation spezieller Software einfach und problemlos anschaulich darstellen zu können.

Das Semantic Web ist bisher eher ein technisches Framework, welches von Fachleuten entwickelt und von anderen Fachleuten verwendet wird. Für eine sinngemäße Nutzung ist

[2] Vorstellung des Smartweb Projetcs auf www.semantic-web.at http://www.semantic-web.at/main.php?prim_nav=11&sec_nav=83&content_id=83&content_typ=link

jedoch eine Verbreitung über diese Fachkreise hinaus notwendig. Bisher wurden jedoch wenig in diese Richtung unternommen. Vielleicht aufgrund der Tatsache, dass diejenigen, welche sich mit dem Semantic Web beschäftigen, meist Fachleute sind und daher meist fachbezogen denken.

2 Fachliches Umfeld

2.1 Informationen im Überfluss

Wir leben in der sogenannten Informationsgesellschaft. Wir leben mit und von Informationen. Jeden Tag strömen Unmengen an Daten auf uns ein. Das Internet umfasst eine schier unvorstellbare Menge an Daten: Texte, Bilder, Videos, Animationen, Tabellen, Graphen. Das Zurechtfinden in diesem Unmengen ist nur durch effiziente Unterstützung auf technischer Ebene möglich. Suchmaschinen sind hier die erste Wahl: Roboter können ohne Pause tausende von Seiten durchsuchen, indizieren und für Suchanfragen nutzen.

Leider führen viele Suchanfragen oft zu Hunderten, Tausenden oder gar Millionen von Treffern. Bisher sind viele für uns selbstverständliche Anfragen für den Computer schlichtweg sinnlos aneinandergereihte Wörter. Die Suchmaschinen suchen nur nach Seiten, in denen alle Wörter auftauchen.

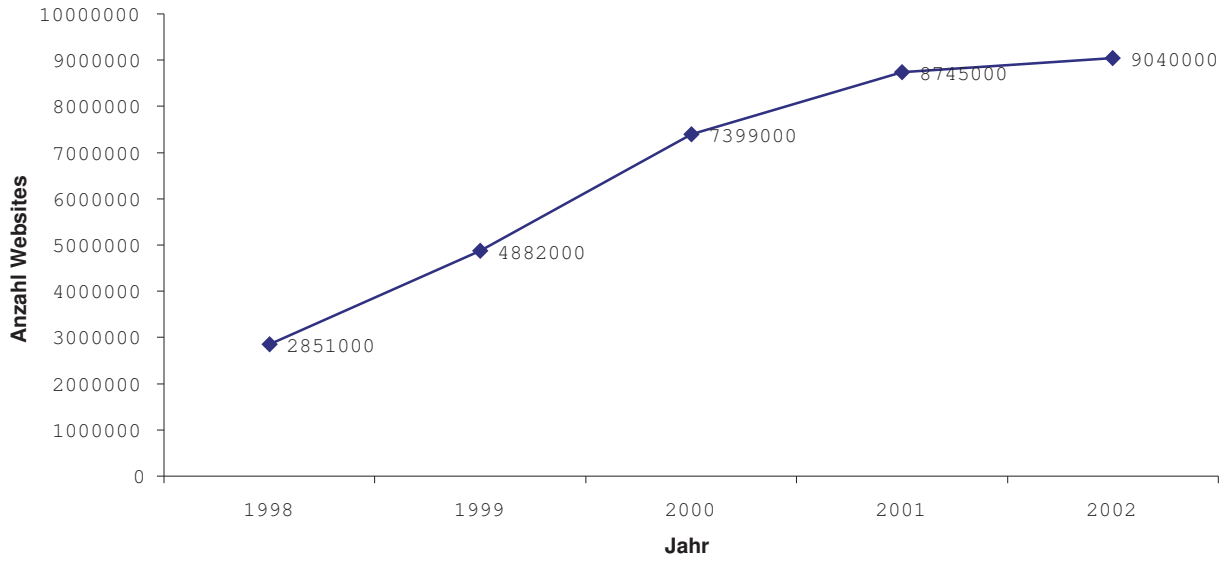

Abbildung 2.1: Entwicklung der Größe des Internet; Quelle www.oclc.org (05.06.2005)

Das Internet ist vor allem in der zweiten Hälfte der Neunziger Jahre rapide gewachsen. Von 1998 zu 1999 wurde ein Zuwachs von über 50% verzeichnet. Das Wachstum hat seitdem allerdings nachgelassen, wohl auch, weil die Gesamtmenge der gespeicherten Informationen mittlerweile gigantische Ausmaße erreicht hat. So hat eine Studie im Jahr 2003 festgestellt [1], dass 2002 fünf Exabyte an Information erzeugt wurden, 92% davon auf Festplatten.

2.1.1 Metadaten

Hier setzen Metadaten an. Metadaten sind Daten, die über Daten berichten. Metadaten kennt man im Internet durch die META-Tags für Suchmaschinen. Diese META-Tags stellen den ersten Schritt in Richtung Datenorganisation dar. Leider wurden die Tags sehr oft dahingehend missbraucht, Seiten in den Suchmaschinen weiter vorn zu positionieren. Deshalb werden diese Meta-Tags heute von den sogenannten Ranking-Mechanismen, welche die Positionierung und Relevanz von Seiten in Suchmaschinen bestimmen, nur wenig beachtet.

Listing 2.1: META-Tag-Beispiel

```
<meta http-equiv="content-language" content="de" />
<meta http-equiv="content-type" content="text/html; CHARSET=UTF-8" />
<meta name="author" content="Thomas Wollburg" />
<meta name="copyright" content="Thomas Wollburg" />
<meta name="description" lang="de" content="Fotografie..." />
<meta name="publisher" content="Thomas Wollburg" />
```

Bekannt sind Metadaten im Internet durch die sogenannten Meta-Suchmaschinen. Diese in den HTML-Code eingebetten Meta-Tags liefern einfache Informationen zum Seiteninhalt. Es existieren Tags für den Autor, den Titel, wann eine Suchmaschine diese Seite wieder besuchen soll. Einige sind in Listing 2.1 zu sehen [13].

2.1.2 Die Notwendigkeit eines umfassenderen Ansatzes

Das Semantic Web [1] ist der nächste Schritt zu einer strukturierteren Informationsgesellschaft. Mit RDF [5] steht ein umfassendes technisches Framework zur Verfügung. Es können echte Beziehungen zwischen Daten gespeichert werden. Somit kann ein Computer wissen, ob man nach einem Konzept für Onlineshops sucht oder nach dem Konzept eines bestimmten Onlineshops. Für einen möglichst reibungslosen systemübergreifenden Datenaustausch müssen alle Beteiligten Systeme die gleiche Sprache sprechen. Diese Basis bietet RDF. Es kann unter anderem in XML dargestellt und somit leicht von Computern gelesen werden.

[1] How much Information 2003 [4]

2.2 Das Semantic Web

2.2.1 Semantik

Semantik bedeutet in der Sprachenwissenschaft Bedeutungslehre. Semantik befasst sich also mit Sinn und Bedeutung von Zeichen (oder im weiteren Sinne eben auch von Bildern). Eine der Hauptaufgaben der Semantik ist es, natürliche Sprache in logische Formeln umzusetzen. Diese Umsetzung in Formeln geschieht im Semantic Web unter Zuhilfenahme von RDF und wird im Anfangsstadium sicher auch ausschließlich manuell durch Menschen geschehen.

2.2.2 Wozu ein Semantic Web

Das Semantic Web soll die Unmengen an Informationen im Internet verfügbar machen. Dieses soll nicht nur durch technische Erreichbarkeit, sondern vor allem durch strukturierte Daten geschehen. Im Semantic Web sucht man nicht nach „Cook“ sondern man sucht in einem bestimmten Kontext nach Cook: der Koch, die Cook-Inseln, oder Thomas Cook. Das „Neue“ dabei ist nicht, dass ein Mensch entsprechende Artikel voneinander unterscheiden kann. Neu ist, dass auch ein Suchmaschinenroboter bei der Indizierung eines Inhaltes semantische Daten erkennt und zur Bewertung der durchsuchten Inhalte mit erfasst. Diese kann er dann bei Suchanfragen durch den Menschen an ihn berücksichtigen.

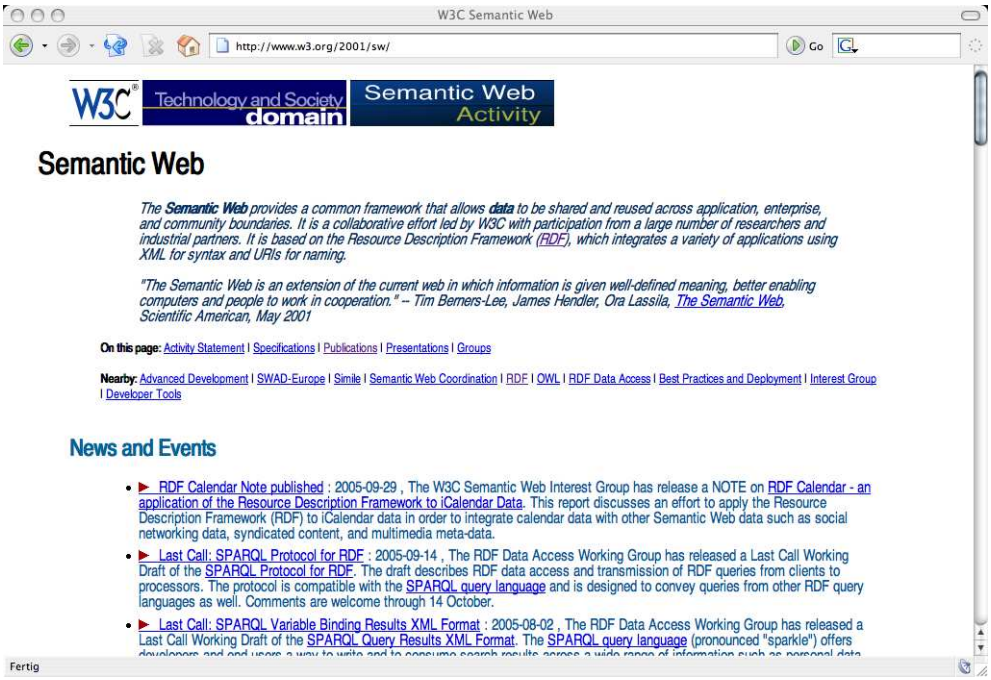

Abbildung 2.2: Offizielle Homepage des Semantic Web

2.2.3 Das Semantic Web kurz erklärt

Das Semantic Web benötigt keine neuen Darstellungswerkzeuge beim Endnutzer, es benötigt auch keine neue Infrastruktur bei den Providern. Es lässt sich nahtlos in das vorhandene Internet, insbesondere ins vorhandene WWW integrieren. Das Semantic Web erfordert lediglich auf Seiten der Inhaltslieferanten und Suchmaschinen die Erzeugung und Integration von Metadaten. Diese Integration erfolgt sehr langsam. Auch die Tatsache, dass das Semantic Web neben anderen vom Internet Miterfinder Tim Berners Lee [2] vorangetrieben wird, kann diese Prozeß nicht wirklich beschleunigen.

Auch politische Entscheidungen „großer“ Unternehmen, vor allem der Suchmaschinen fördern oder behindern die Entwicklung des Semantic Web. So hat Peter Norvig, Director of Search Quality beo Google, die Notwendigkeit und Verbreitung des Semantic Web zwar heruntergespielt [15]:

> A friend of mine just asked can I send him all the URLs on the web that have dot-RDF, dot-OWL, and a couple other extensions on them; he couldn't find them all. I looked, and it turns out there's only around 200,000 of them. That's about 0.005% of the web. We've got a ways to go.

andererseits werden im „Summer Of Code“ Projekt von Google auch Semantic Web Applikationen gefördert [16].

Der Zugang zum Semantic Web erfordert Oberflächen, welche die technische Umsetzung der Kontextsensitivtät der menschlichen Sprachen intuitiv verfügbar machen. Die interaktive Darstellung von Graphen ist ein wichtiger Schritt in diese Richtung. Durch die Verbindung der einzelnen Elemente miteinander lassen sich ihre Beziehungen schnell und einfach verstehen.

Listing 2.2: Beispiel für Seite „RDF-Dokumente“

```
1  <?xml version="1.0"?>
2  <rdf:RDF
3    xmlns:ima="http://imaginers.de/rdfs/elements/1.0/"
4    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
5       <rdf:Description rdf:about="http://imaginers.de/index.php?a=2">
6            <ima:Author>Thomas Wollburg</ima:Author>
7            <ima:Title>RDF-Dokumente</ima:Title>
8            <ima:Related rdf:Resource="http://imaginers.de/index.php?a=1" />
9       </rdf:Description>
10 </rdf:RDF>
```

Listing 2.2 zeigt ein einfaches Beispiel in XML-Darstellung. Abbildung 2.3 zeigt die dazu gehörige Graphendarstellung.

Die dritte offizielle Methode, RDF darzustellen ist die „Notation 3“ [14] auch kurz als N3 bezeichnet. Wie in den anderen Notationen auch, bestehen hier alle Informationen aus Tripeln:

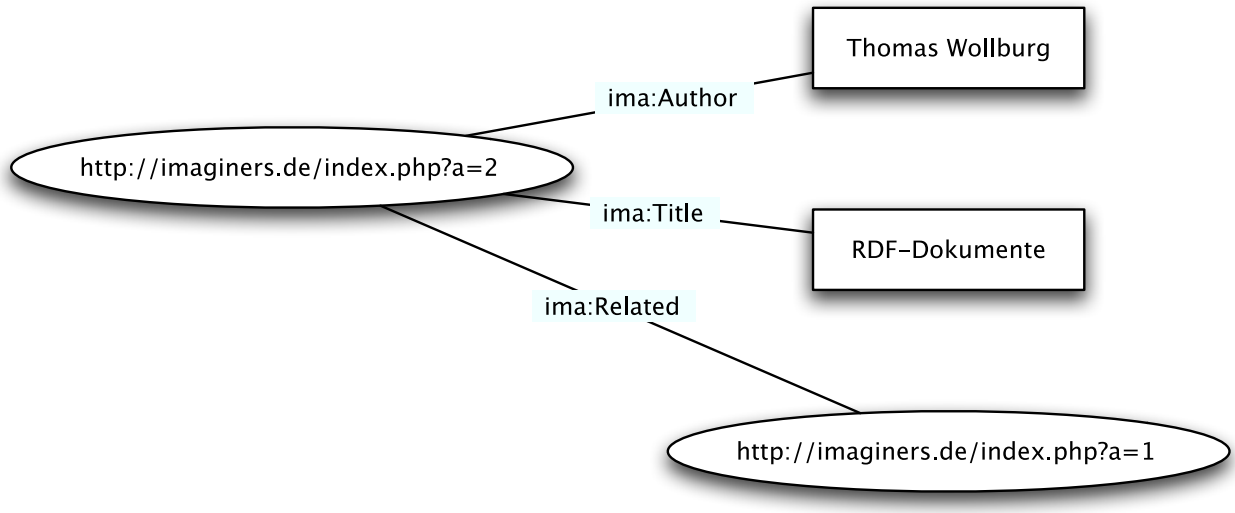

Abbildung 2.3: Beispiel „RDF-Dokumente“

Listing 2.3: Beispiel in Notation 3

```
<http://imaginers.de/index.php?a=2>
    <http://imaginers.de/elements/Title> "RDF–Dokumente"
```

dabei kann auch eine kurze Notation verwendet werden, z.B. wenn sich die Angaben innerhalb eines Dokumentes auf Teiles desselben beziehen:

Listing 2.4: Beispiel in Notation 3 (Kurznotation)

```
<#ChapterOne> <#Follows> <#Introduction>
```

Zusätzlich sollen verwandte Daten leichter aufzufinden sein. Zusammenhängende Dokumente erleichtern die Suche nach Wissen. Beziehungen lassen sich im Semantic Web auf verschiedenste Art und Weise darstellen. So lassen sich zum Beispiel Dokumente anhand eines Autors als zusammengehörig darstellen. Eine andere Möglichkeit ist die direkte Benennung von verwandten Inhalten.

3 Aufgabenstellung

3.1 Ziele

Ziel des „SemanticFlash“-Projektes ist es, RDF-Dokumente grafisch aufbereitet darzustellen. Dazu sollen zunächst verfügbare RDF-Dokumente dargestellt werden und dann nach Auswahl eines Dokumentes eine Detailansicht. Diese Detailansicht soll die Möglichkeit bieten, einfache Literal-Werte zu bearbeiten.

Das abstrakte Semantic Web soll dem Menschen näher gebracht werden. Normalerweise wird RDF in XML dargestellt. Diese Darstellung ist auch sinnvoll, da XML schnell und einfach sowie plattformunabhängig von Computern verarbeitet werden kann. Jedoch können nur die wenigsten Menschen XML effizient erfassen. Zumal RDF durch die Verwendung von Qnames die Lesbarkeit zusätzlich erschwert. Qnames sind qualifizierten Namen, welche den Namensraum des Tags beinhalten:

```
<mns:Author>Max Mustermann</mns:Author>
```

im Beispiel is „mns“ der Namensraum und „Author“ das Tag.

RDF lässt sich auch in Graphenform darstellen. So hat das W3C [10] auch den RDF-Validator [11] neben der textualen Tripeldarstellung auch mit einer Graphenausgabe versehen. Die für die meisten Menschen unverständlichen Tagverschachtelungen in XML sollen grafisch aufbereitet dem Menschen näher gebracht werden.

3.2 Grenzkriterien

Folgende Funktionen müssen implementiert werden, um eine grundlegende Funktionalität zu gewährleisten:

- *Auflistung von auf dem Server verfügbaren Dokumenten*

  Es sollen zunächst alle RDF-Dateien aufgelistet werden, welche sich in einem bestimmten Verzeichnis befinden. Zu Darstellung der Auswahl-Liste werden die Dateinamen der Dokumente angezeigt.

- *Einlesen und Parsen*

  Das ausgewählt Dokument soll direkt aus Flash heraus vom Server gelesen werden. Danach muss es geparst werden. Das Parsen soll das Sicherstellen, dass das Dokument syntaktisch korrekt (valide) ist und angezeigt werden kann. Das Parsen soll weiterhin

für die Darstellung relevanten Informationen in der lokalen Kopie des RDF auf Clientseite (innerhalb des Flash-Films) speichern. Es sollen zunächst nur Grundlegende RDF-Kontrukte dargestellt werden können. Eine einfache Erkennung von Tripeln ist zunächst ausreichend. Es werden zum Beispiel keine Sequenzen unterstützt.

- *Darstellen als Graph*

  Das geparste Dokument soll, sofern es valide ist, als Graph dargestellt werden. Die Darstellung soll sich dabei am W3C RDF Validator orientieren: Resourcen werden als Ellipsen, Prädikate als gerichtete Linien (mit einem Pfeil in Richtung Objekt) und Literale als Rechtecke dargestellt werden.

- *Hilfe*

  Aufgrund der Komplexität des Sachverhaltes ist eine Hilfe, welche den Graphen erklärt, unerlässlich. Die Hilfe soll einfach zu finden sein und parallel zum Graphen angezeigt werden können.

3.3 Wunschkriterien

- *Bearbeitungsfunktion für Literale*

  Literalwerte, welche nur als Objekte von Tripeln auftauchen können, sollen bearbeitbar sein. Die Bearbeitung soll dabei intuitiv erfolgen: ein Doppelklick zum Öffnen einen Editors. Der Editor soll dann den gewählten Wert anzeigen und über einen Speichern- und einen Abbrechen-Button verfügen. Optional ist auch ein Zurücksetzen-Button sinnvoll, welcher dann ein Abbrechen und erneutes Öffnen in einem Klick zur Verfügung stellen würde.

- *Erstellung neuer Tripel*

  Es soll die Möglichkeit geschaffen werden, auf Grundlage der in einem Dokument vorhanden Resourcen neue Tripel zu schaffen. Die vorhandene Resource soll dabei als Subjekt verwendet werden. Die verfügbaren Prädikaten sollen separat eingelesen werden. zunächst ist nur das Hinzufügen von Literal-Werten angedacht.

- *Erkennung und Darstellung von Sequenzen*

  Es sollen RDF Bags erkannt werden. Diese Art von Containern kann Sequenzen beinhalten. Die dazu notwendigen Blank Nodes müssen dann ebenfalls korrekt dargestellt werden.

3.4 Abgrenzungskriterien

Folgende Funktionen soll SemanticFlash nicht erfüllen:

- *Erstellen von RDF-Dokumenten*

  Das Erstellen von RDF-Dokumenten sollten die Systeme übernehmen, welche die Inhalte erzeugen, im Internet sind das meist Content-Management-Systeme. Die Erstellung von RDF sollte automatisch bei der Erstellung der zu beschreibenden Inhalte erfolgen.

- *Suche in RDF-Dokumenten*

  Die zentrale Funktion rund um RDF und Metadaten ist das Auffinden von Inhalten. Dazu müssen Suchprogramme geschaffen werden, welche RDF interpretieren können. SemanticFlash ist ein reines Darstellungs- (und Berbeitungs-) Werkzeug.

4 Das Resource Description Framework

4.1 Allgemeines

Das Resource Description Framework ist ein XML-basierte Spezifikation zur Darstellung von Metadaten über Webseiten und andere Objekten. RDF wird aktuell in verschiedenen Siutationen eingesetzt: so gibt es zm Beispiel RSS 1.0 [1] (welche im Gegensatz zu RSS 2.0 auf RDF aufsetzt), es gibt das DSpace-Projekt [7] vom MIT und FOAF (Friend of a Friend) [2], um nur einige zu nennen.

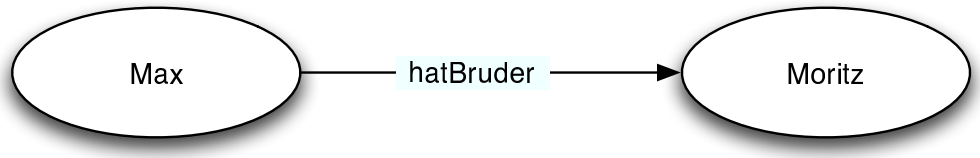

Abbildung 4.1: Einfaches RDF-Beispiel

RDF stellt Objekte zueinander in Beziehung. Es soll für Menschen selbstverständliche Sätze wie: „Moritz ist der Bruder von Max“ auch für Computer verständlich machen. Ein einfaches Beispiel dazu ist Abbildung 4.1, hier hat man als Subjekt Max, das Objekt ist Moritz und das Prädikat ist „hat Bruder“. Solche Aussagen lassen sich wie in Abbildung 4.2 miteinander verbinden. So kann das Objekt des einen Statements das Subjekt eines weiteren sein. Ein Subjekt kann auch gleichzeitig mit mehreren Objekten verbunden werden.

RDF ist eine Spezifikation, welche vor allem dabei hilft, Inhalte von verschiedenen Organisationen zueinander kompatibel zu machen. Es kann immensen Nutzen bringen, aber auch nur eine technische Last erzeugen. Vor dem Einsatz sollte man daher eine genauso Analyse der Situation vornehmen. Ein Einsatzgebiet könnte zum Beispiel die Kommunikation eines Autoherstellers mit seinen Lieferanten sein, welche ja sicher alle verschiedene Software einsetzen und auch unterschiedliche Ansichten haben, was Daten sind und was nicht.

Die bereits erwähnten in HTML eingebetteten Metadaten [13] können als Vorläufer von RDF betrachtet werden. So kann die Tatsache, dass ein Meta-Tag in einem bestimmten Dokument steht, zusammen mit dem Schlüssel-Wert-Paar des Meta-Tags als Tripel betrachtet werden. Diese Meta-Tags waren allerdings eher dazu gedacht, Informationen über das Dokument für Menschen zur Verfügung zu stellen. Obwohl diese Informationen bei der Anzeige des Dokuments

[1] RSS - Really Simple Syndication http://web.resource.org/rss/1.0/
[2] FOAF http://www.foaf-project.org/

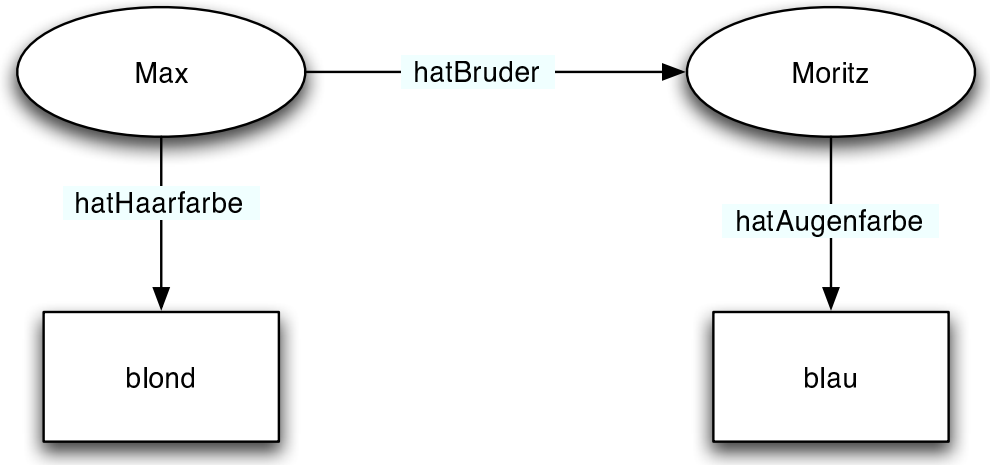

Abbildung 4.2: RDF-Beispiel

im Browser nicht erscheinen, sollten sie doch von Suchmaschinen benutzt werden, relevante Treffer zu finden.

4.2 Was macht RDF

RDF beschreibt Resourcen. Es nutzt dazu einen objektorientierten Ansatz: jede Resource ist die Instanz einer bestimmten Klasse. Für Resourcen gibt es in RDF die Klasse rdf:Resource. Daneben gibt es eine Reihe weiterer Klassen [6], z.B. für Zeichenketten, Datentypen und Eigenschaften. Dieser Ansatz erleichtert die Erstellung von objektorientierten Applikationen, welche RDF benutzen. Man kann z.B. eine Java- oder PHP-Klasse verwenden, um ein RDF-Dokument zu repräsentieren.

RDF an sich ist eine Sammlung syntaktischer Regeln. RDF Schema [6] ergänzt RDF um „semantische“ Angaben. Es ermöglicht semantische Aussagen indem es bestimmten RDF-Klassen und -Eigenschaften Bedeutungen gibt. Es ist selbst sind in RDF formuliert. RDF definiert Klassen, Eigenschaften und Werte, mit RDF Schema kann man applikationsspezifische Klassen, Eigenschaften und Werte definieren und so den abstrakten Eigenschaften von RDF spezifische Bedeutungen geben. Anstelle von RDF Schema lassen sich theoretisch auch andere Erweiterungen von RDF einsetzen.

4.3 RDF und das Semantic Web

Das Semantic Web ist einer der möglichen Anwendungsfälle für RDF. Neben dieser Anwendung kann RDF auch in Netzwerken eingesetzt werden. Durch die selbstbeschreibenden Eigenschaften von RDF sind aber alle RDF-Installationen potentiell im Semantic Web erfassbar.

Wenn RDF und Semantic Web in Zusammenhang auftauchen, gehört auch immer die Web Ontology Language [9] dazu. Mit einer Ontologie ist eine explizite formale Spezifikation gemeint. Diese legt fest, wie Objekte, Konzepte und deren Attribute und Eigenschaften in einem bestimmten Umfeld dargestellt werden. Die OWL ist somit vor allem für Anwendungen gedacht, welche Inhalte verarbeiten sollen und nicht für Anwendungen, welche Inhalte anzeigen.

5 Das Semantic Web

5.1 Was ist das Semantic Web

Die einfach deutsche Übersetzung „semantisches Internet“ ist genau wie der englische Begriff erklärungsbedürftig. Das Semantic Web ist kein neues Protokoll und keine neue Technologie. Das Semantic Web ergänzt das bestehende Internet um domänenübergreifend lesbare Metadaten.

Für den Menschen verständliche Informationen sollen auch für Computer „verständlich“ gemacht werden. Dieses Verständnis auf Seiten des Computer soll es widerum dem Menschen erleichtern, Informationen mit Hilfe das Computers zu finden.

Dieser Ansatz ist auch unbedingt notwendig, da die bereits erwähnte Flut an Informationen ein „Verstehen“ der Daten durch Computer notwendig macht - der Computer muss wissen, was der Mensch von ihm verlangt.

Im Moment suchen Suchmaschinen in einfachen Verfahren nach Häufigkeiten und Verlinkungen. Einige Ansätze schlagen dem Benutzer „verwandte Begriffe“ vor, um die oft mehr vier- oder fünfstelligen Suchergebnisse einzugrenzen. Es wird auch versucht, den gesuchten Kontext zu erkennen, es fehlt solchen Ansätzen allerdings an Semantik: bei der Suche nach dem Konzept für die Erstellung eines Onlineshops möchte man keine Onlineshops ansehen, die nach einem tollen, nicht genannten Konzept erstellt wurden.

Obwohl RDF immer wieder im Zusammenhang mit dem Sematic Web erwähnt wird, wurde das Semantic Web nicht explizit an RDF gebunden. RDF und OWL sind viel mehr Implementationen des Konzepts, für welches das Semantic Web steht.

Das Suchen im Internet unterscheidet sich nicht wesentlich vom Suchen im „normalen“ Internet. Das war eine der großen Vorgaben vor der Umsetzung. Für den Endbenutzer ändert sich nichts, nur die Ergebnisse werden besser. Im Hintergrund stellt sich die notwendige Arbeit als Erweiterung bei den Inhaltsanbietern und Implementierungen von RDF-Parsern bei den Suchdienstanbeitern. Insgesamt lässt sich den Abbildungen 5.1 und 5.2 entnehmen, dass auch die Komplexität des Gesamtsystemes Internet nicht wesentlich erhöht wird.

5.2 Begriffsklärungen

Hier sollen zunächst einige der im Umfeld des Semantic Web verwendeten Begriffe erläutert werden.

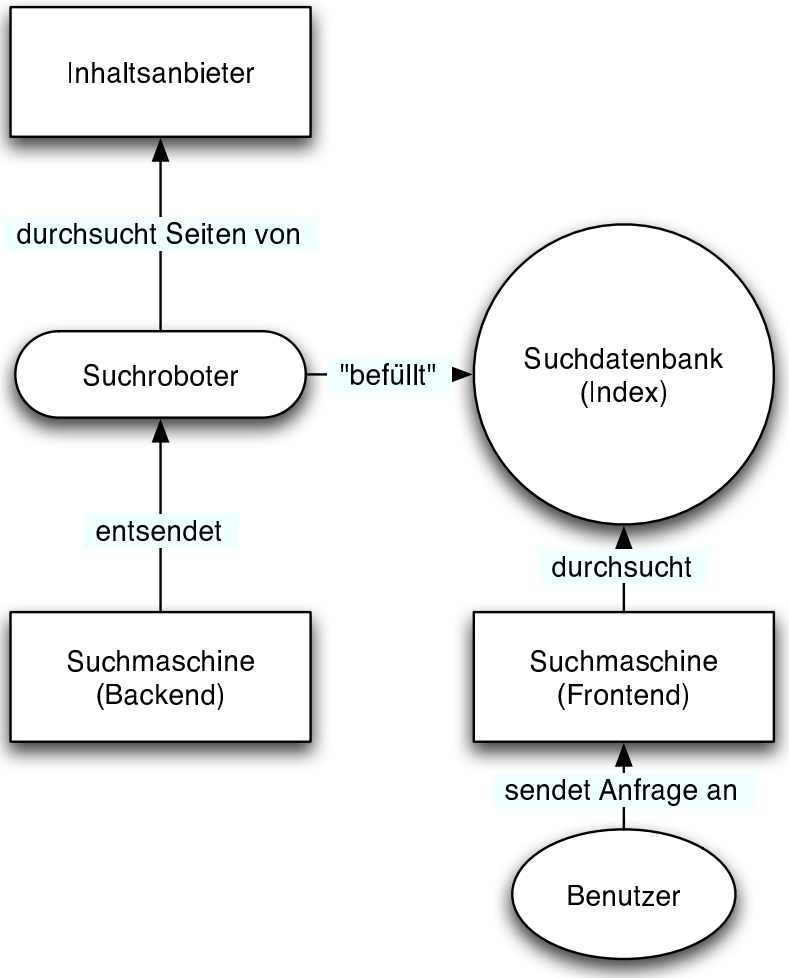

Abbildung 5.1: Suchen im Internet

5.2.1 Ontologie

Der Begriff Ontologie ist der Philosophie entlehnt. Dort beschäftigt sich die Ontologie mit dem Sein an sich. In der Informatik ist eine Ontologie ein formales System zur Beschreibung von Wissen. Es sind Konzepte und Relationen definiert. Eine Ontologie stellt einen Mechanismus zur Verfügung, mit welchem man Objekte und ihre Beziehung zueinander beschreiben kann.

Diese Beschreibung ist immer auf ein bestimmtes Interessengebiet beschränkt, um die Komplexität in einem endlichen Rahmen zu halten. Ontologien sind also Begriffssysteme. Sie unterscheiden sich von einfachen Vokabularien durch Relationen der definierten Begriffe zueinander - zwischen den einzelnen Bestandteilen gelten Regeln [1].

[1] Wikipedia Ontologie (Informatik)

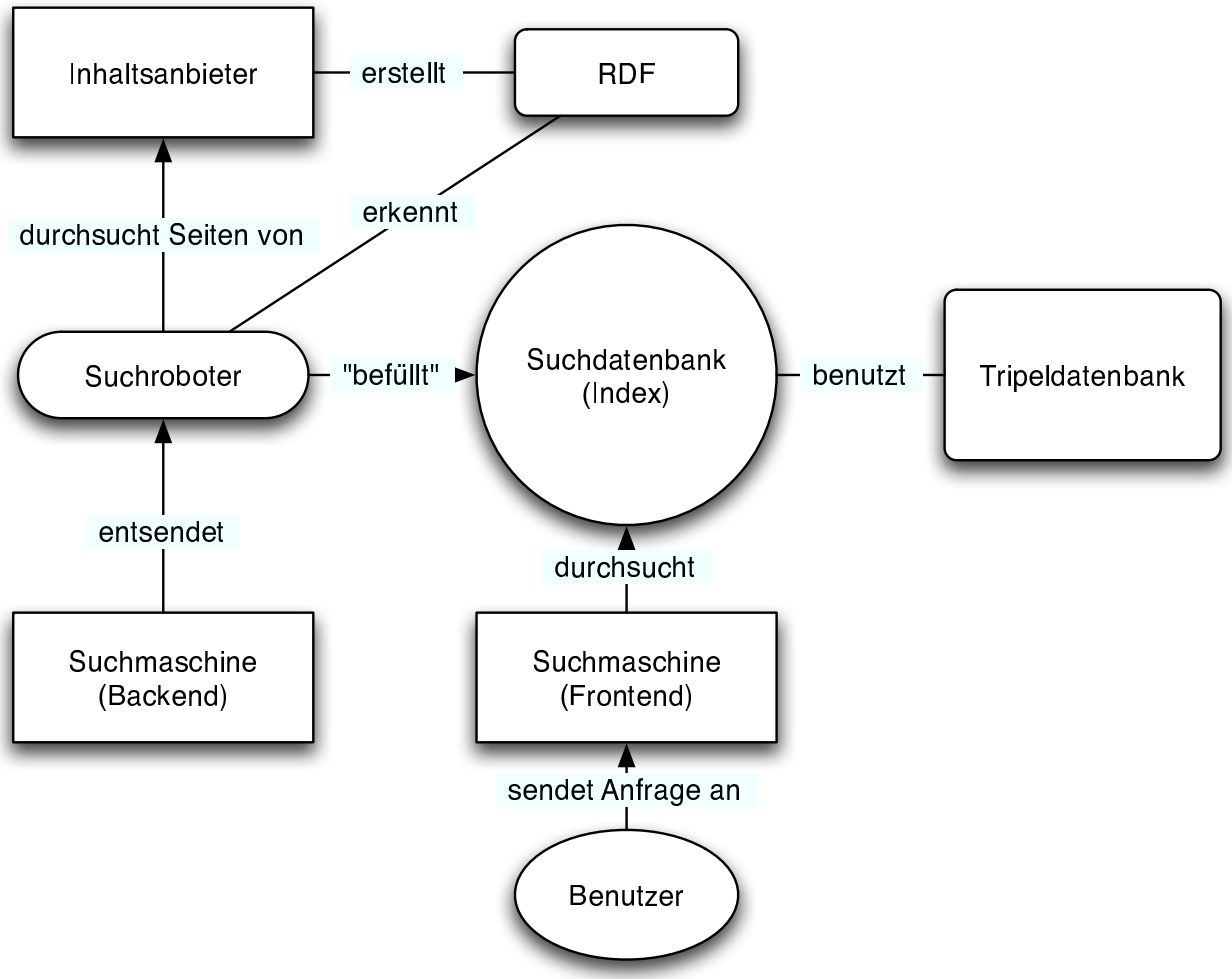

Abbildung 5.2: Suchen im Semantic Web

Ontologien sind einer der zentralen Schlüssel bei der Schaffung von Inhalten, welche sich in das Semantic Web integrieren. In immer mehr Bereichen werden Ontologien, welche früher nur im Bereich der Künstlichen Intelligenz eine Rolle spielten, eingesetzt. So arbeiten heute Suchkataloge wie Yahoo! oder Versandhäuser wie Amazon

5.2.2 Tripel

RDF verwendet die sogenannte Tripeldarstellung. Gegenüber einer einfachen dualen Schlüssel - Wert - Darstellung sind Tripel in der Lage, Objekte miteinander in Beziehung zu setzen. So kann ein Schlüssel-Wert-Paar zum Beispiel „Material: CoolMax“ [3] darstellen. Die Darstellung des Sachverhaltes „Dieses Poloshirt besteht aus CoolMax“ jedoch benötigt ein Tripel. Bei diesem Tripel ist Poloshirt das Subjekt (die Sache, die es zu beschreiben gilt), das Prädikat (der

Sachverhalt des Subjektes, welcher beschrieben wird) ist Material und das Objekt (der Wert des Prädikates) ist CoolMax.

5.3 Die Technologien im Überblick

Neben RDF als syntaktischer Grundlage verhilft vor allem die Ontology Web Language [9] dem Semantic Web zur Funktionalität. OWL ist die Spezifikation des W3C, um in RDF implementierten Sprachen formal so darzustellen, dass auch „fremde“ Agenten diese Sprache verstehen: so ist OWL quasi die semantische Ergänzung zu RDF, welches die syntaktische Grundlage im Semantic Web bildet.

5.4 Die Vorläufer der aktuellen Technologien

Der Vorgänger von OWL ist DAML+OIL [8]. DAML+OIL stellt den Zusammenschluss von der „Defense Advanced Research Projects Agency“ (eine Einrichtung des US-Militär) und dem „Ontology Inference Layer“ (entwickelt im Rahmen eines europäischen Forschungs-Projektes) dar.

DAML+OIL entstand nach RDF und ergänzte dieses um zusätzliche Modellierungs-Möglichkeiten. OIL ist dabei kompatibel mit RDF Schema. In OIL sind verschiedene Sprachebenen definiert. Für alle diese Sprachebene existieren RDF Schema-Definitionen [2].

In OWL sind drei Sprachebenen definiert: OWL Lite, OWL DL (Descriptions Logics) und OWL Full. Dabei steigt die Komplexität von einer Ebene zur nächsten. Die einfachste Form ist OWL Lite. Danach folgt OWL DL und letztendlich OWL Full. In vielen Bereichen entwickeln Domänen-Experten Ontologien, um Ihre fachspezifischen Informationen zu veröffentlichen und untereinander zu kommentieren.

[2] http://www.ontoknowledge.org/oil/

6 Die Umsetzung im Projekt

6.1 Einführung

Als Plattform dient www.imaginers.de. Technische Basis bildet PHP mit einer Flash MX Benutzerschnittstelle für die Eingabe der Daten. Der Grund für die Verwendung von Flash liegt in den ausgereiften XML-Funktionalitäten und den Darstellungsvorteilen gegenüber klasischem HTML, vor allem im Bereich der dynamischen Daten und der Tripeldarstellung.

6.2 Beschreibung des Tools

Das Flash-Interface soll eine einfache Möglichkeit bieten, in RDF/XML gespeicherte Informationen darzustellen und zu bearbeiten bzw. zu erstellen. Es werden die XML-Fähigkeiten von Flash genutzt. Die Möglichkeit, XML-Strings direkt aus Flash per HTTP zu versenden und die seit Version 6 integrierte Unterstützung des XML-Standard-Zeichensatzes UTF-8 sprechen für eine solche Umsetzung.

6.3 Konzept des Tools

Das Tool besteht aus den beiden Hauptbestandteilen Frontend und Backend. Im Frontend Flash MX zur Umsetzung von Darstellung und Logik verwendet. Im Backend wird PHP die Schnittstelle zum Frontend bilden und auf den Datenspeicher zugreifen. Das Tool ist nicht in der Lage, jede mögliche Darstellungsform von RDF zu interpretieren. Es wird ein einfaches überschaubares RDF-Format verwendet.

Wie in Abbildung 6.1 zu sehen, erfolgen alle Zugriffe auf das Backend über ein zentrales Schnittstellenskript (index.php). Dieses Skript verarbeitet dann die Anfrage und liefert den angeforderten Inhalt zurück. Bei Anfragen vom Browser wird dabei HTML zurückgeliefert. Auf Anfragen des Flash-Plugins werden entweder die RDF-Dokumente ausgegeben oder, zum Beispiel als Antwort auf eine Speicheranfrage, eine XML-formatierte Antwort.

Die für das Frontend entwickelte RDF-Klasse wird im Flash-Plugin des Browsers ausgeführt. Daher ist die Ausführung von der verfügbaren Pluginversion abhängig. Für eine korrekte Anzeige ist ein Flashplayer 7-Plugin notwendig, welches für verschiedenste Browser auf unterschiedlichen Plattformen verfügbar ist.

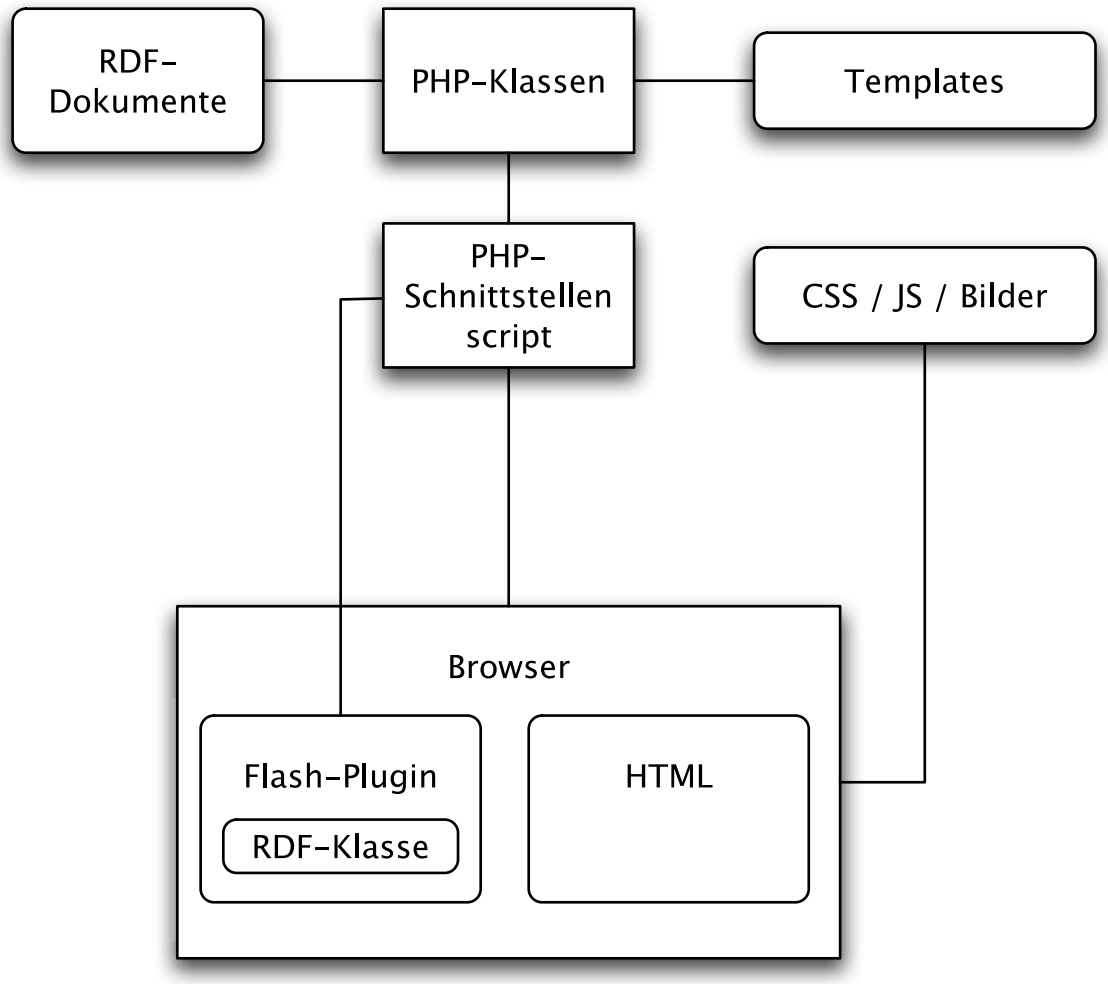

Abbildung 6.1: Aufbau Semantic Flash (Frontend/Backend)

6.3.1 Was soll das Tool leisten

Gerade bei einem solch komplexen Thema sind die Möglichkeiten für Verbesserungen an der Oberfläche und im Workflow schier unbegrenzt. Deshalb wird diese Version des Tools ein klar abgegrenztes Funktionsspektrum haben. Ein modularer objektorientierter Aufbau ermöglich die problemlose Erweiterung der bestehenden Funktionalität zu einem späteren Zeitpunkt.

Zunächst einmal geht es um die Darstellung der Machbarkeit der interaktiven RDF-Darstellung mit Hilfe einer frei verfügbaren Software. Dies wurde erreicht. Es wurde eine intuitive Bearbeitungsfunktion für Literalwerte integriert. Durch die frei Beweglichen Subjekte und Objekte der Graphen kann der Benutzer das Informationsnetz beliebig anordnen. Da Menschen am besten lernen, wenn sie etwas tun, wird hierdurch ein optimaler Lernprozess erzielt.

6.4 Umsetzung

6.4.1 Verwendete Sprachen und Autorensysteme

Für die Umsetzung des Tools wurde auf Standards gesetzt:

- XHTML: wie beinahe alle anderen Webseiten auch, wurde für die Gestaltung der Website HTML verwendet. Ergänzt wird das HTML durch Cascading Stylesheets (CSS) und Javascript (JS)
- PHP 4: zur Realisierung der Logik und Verarbeitung der Benutzeranfragen wurde die freie Scriptsprache PHP in der Version 4 verwendet. Es wurde objektorientiert programmiert. Aufgrund der Tatsache, dass viele Webspace-Provider noch kein PHP 5 unterstützen, wurde die alte etablierte Version 4 der Sprache verwendet.
- Flash 7: Für die Darstellung des RDF-Gaphen wurde Macromdia Flash MX 20004 Professional eingesetzt. Das Flash-Plugin ist weit verbreitet und die Funktionen der Version 7 erfüllen alle für die Umsetzung benötigten Anforderungen.

6.4.2 RDF-Dokumenten-Klasse

Den Anfang der Umsetzung bildet die Erstellung eine RDF-Dokumenten-Klasse in ActionScript. ActionScript ist die in Macromedia Flash integriert Scriptsprache zur Programmierung logischer Abfolgen und zur Steuerung von Oberflächenelementen. In der verwendeten Version 2.0 von Actionscript besteht die Möglichkeit, objektorientiert Anwendungen zu entwickeln.

Über diese Klasse werden alle logischen Schritte laufen: einlesen einer RDF-Datei, Extraktion der Tripel, Bearbeiten und Speichern bzw. senden der Datei zum Server, um sie dort speichern zu lassen.

6.4.3 PHP-Klassen im Backend

6.4.3.1 Das Backend allgemein

Das in PHP 4 geschriebene Backend besteht aus verschiedenen Teilen. Dabei werden funktionale und darstellende Teile strikt getrennt. Der logische Teil besteht dabei aus:

- Konfigurationsdatei
- Klassen zur Verarbeitung der Logik
- Schnittstellenskript (Zentraler Controller)

Durch diese Architektur läßt sich die Anwendung schnell und einfach erweitern. Für die Darstellung der Website wurden

- (Smarty-) Templates
- Grafiken
- Zentrales Stylesheet (CSS)
- JavaScript

ausgearbeitet. Die Website basiert auf XHTML. Für die Farben, Abstände, Formate wurde ein Stylesheet entwickelt. Für die Kommunikation des Semantic-Flash mit dem Browser wurden JavaScript-Funktionen geschrieben.

6.4.3.2 Klasse für die grafische Oberfläche

Für die Darstellung der Website imaginers.de wurde eine eigene Klasse entwickelt, welche die Anfragen der Benutzer verarbeitet. Für die Erzeugung des fertigen HTML wurde die Template-Engine „Smarty“[1] verwendet.

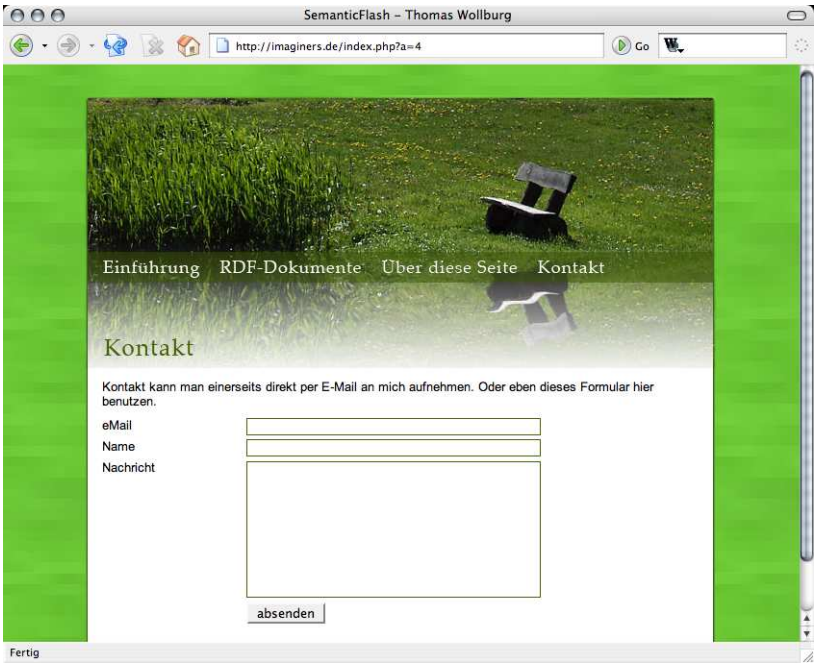

Abbildung 6.2: Semantic Flash Kontaktseite

[1] http://smarty.php.net

6.4.3.3 RDF-Klasse für das Backend

Als passendes Gegenstück zur Klasse auf der Clientseite wurde auch im Backend eine RDF-Klasse entwickelt. Diese dient hauptsächlich zum Speichern der RDF-Dokumente. Vor dem Speichern wird dabei der übergebene XML-String mit Hilfe der Klasse „ARC_rdfxml_parser“ vom appmosphere-CMS [2] validiert. Ist das Dokument valide, so wird es für das Speichern aufbereitet.

Das das Dokument vom Flash-Plugin als Zeichenkette ohne Umbrüche zum Server gesendet wird, wird es vor dem Speichern noch mit Einrückungen und Umbrüchen versehen. Hier wird die PEAR-Klasse [3] „XML_Beautifier“ [4] verwendet. Zusätzlich wird noch der Standard-XML-Header (<?xml version="1.0"?>)eingefügt, da Flash diesen nicht mit zum Server sendet. Nach dem Speichern wird dann eine entsprechende Nachricht zum Server gesandt.

6.4.4 Grafische Oberfläche

Die grafische Oberfläche wird im ersten Stadium noch recht rudimentär sein. Die Ausgabe der Tripel soll sich an der Darstellung des W3C RDF Validator [11] orientieren. Ergänzt wird die Tripeldarstellung durch eine Menüleiste und den Editor.

6.4.4.1 Funktionen in der Menüleiste

Die Menüleiste enthält die Hauptmenüpunkte “Datei„ und “Hilfe„.

Der Menüpunkt „Datei“ stellt die zum Speichern und Schließen des Tools notwendigen Aktionen zur Verfügung. Das Verhalten des Menüs orientiert sich dabei an üblichen Standards. Über den Menüpunkt „Hilfe“ wird die integrierte Hilfefunktion verfügbar gemacht.

6.4.4.2 Der Editor

Der Editor ermöglicht es, (Literal-) Werte direkt zu bearbeiten. Der Editor hat dabei neben einem Button zum Übernehmen der Änderungen auch einen Button zum Abbrechen und einen zum Zurücksetzen des Inhaltes. Der Editor ist, wie die Infoboxen beim Speichern auch, modal umgesetzt, d.h. die anderen Funktionen des Tools sind nicht erreichbar, solange der Editor geöffnet ist. So können Mißverständnisse und Fehler auf der Benutzerseite vermieden werden.

6.4.4.3 Interaktion des Tools mit dem Browser

Das Flash-Plugin kommuniziert via Javascript mit dem Browser. Daher muss ein Bearbeiter in jedem Fall Javascript aktiviert haben. Über diese Kommunikation kann das Tool sich zu Beispiel selbst schließen.

[2] http://www.appmosphere.com/pages/en-arc
[3] http://pear.php.net
[4] http://pear.php.net/package/XML_Beautifier

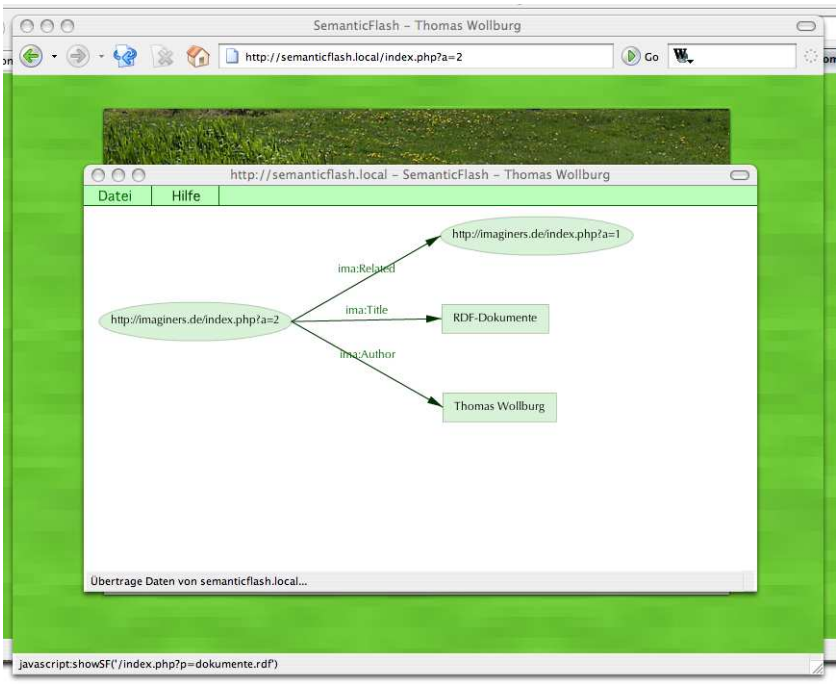

Abbildung 6.3: Semantic Flash RDF-Darstellung

Abbildung 6.4: Semantic Flash Menüleiste

6.5 Das verwendete Vokabular

Das für die Umsetzung verwendete Vokabular ist kurz gehalten und an die Anforderungen einer allgemeinen Internetseite angelehnt. Zentrales Element ist dabei ein Artikel, der mit Attributen Titel, Autor, Erstellungsdatum, Datum der letzten Änderung versehen ist. Durch das flexible Konzept von RDF kann das Vokabular bei Bedarf einfach erweitert werden, sowohl in den Details als auch im Umfang.

6.6 Die verwendete Syntax

Neben einem Vokabular wurde auch eine Syntax festgelegt, welche nicht alle Möglichkeiten von RDF ausschöpft. Dies ist notwendig, um den Aufwand für die Implementierung in Flash in einem begrenzten Rahmen zu halten. Prinzipiell ist es aufgrund der Fähigkeiten von Actions-

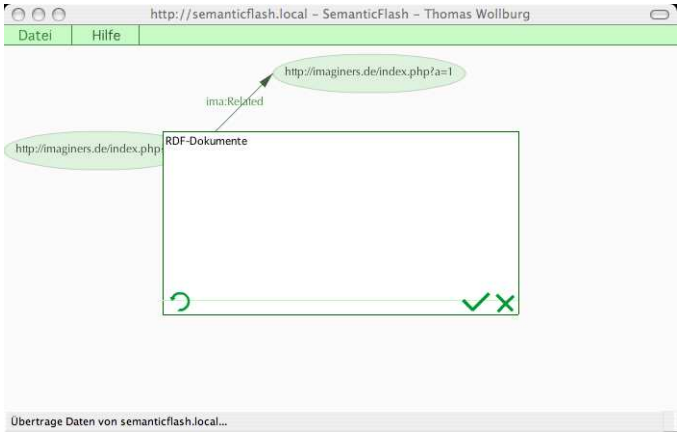

Abbildung 6.5: Semantic Flash Editor für Literalwerte

cript möglich, sämtliche in RDF gegebenen Möglichkeiten auch umzusetzen, jedoch kann der Aufwand für eine entsprechende Implementierung sehr gross werden, da Flash zum Beispiel keinen Zugriff via XPath [5] auf das XML ermöglicht.

6.7 Probleme bei der Umsetzung

Die in der Einführung erwähnten XML-Funktionalitäten von Flash sind anfangs doch etwas gewöhnungsbedürftig. So ist der erste des Knoten eines validen XML-Dokumentes nicht der Wurzelknoten, sondern die XML-Doctype-Deklaration. Auch der Zugriff auf Elemente über Flash-XML-Attribute „childNodes“ und „attributes“ erweist sich durch den „for(Objekt in Liste)“ Zugriff als doch recht unterschiedlich im Vergleich zu anderen Sprachen. Nach einer gewissen Einarbeitungsphase kommt man dann aber doch gut voran.

Auch die fehlerhafte XML-Behandlung von Flash hat einiges an Arbeit und Suchen gekostet. Die Implementierung der Funktion „hasChildNodes()“ schien zunächst fehlerhaft.

So wurde für folgende XML-Fragment aus dem Beispiel 3.12 aus [12]

```
<pstcn:series>
  <rdf:Description rdf:ID="monsters">
    <pstcn:seriesTitle>A Tale of Two Monsters</pstcn:seriesTitle>
  </rdf:Description>
</pstcn:series>
```

für das Element „pstcn:series“ für das erste Kind angegeben, dass es sich um einen Textknoten handelt, was ja offensichtlich nicht der Fall ist.

[5] XPath stellt Methoden zur Verfügung, um schnell und einfach auf Inhalte von XML-Dokumenten zuzugreifen

Nach ein paar Tests hat sich herausgestellt, dass Flash Zeilenumbrüche fehlerhaft interpretiert, solange man die Option "ìgnoreWhite“ nicht explizit auf „true“ setzt.

6.8 Mögliche zukünftige Änderungen

6.8.1 Serverseitige Aufbereitung des RDFs

Das Parsen von RDF in Flash ist direkt an die XML-Fähigkeiten von Flash gebunden. Da der Zugriff dabei sehr umständlich sein kann, könnte es sinnvoll sein, das RDF serverseitig aufzubereiten und in einem vereinfachten Format an Flash zu senden. Dieser Schritt würde jedoch die Komplexität des Tools weiter erhöhen, da ja die in Flash vorgenommen Änderungen zurück in das ursprünglich gelieferte RDF gebracht werden müssen.

Ein großer Vorteil dieser Architekturänderung wäre die Möglichkeit, die besseren RDF-fähigkeiten von verfügbarer Software für die Serverseite zu nutzen (z.B. Jena). Ein damit verbundener möglicher Nachteil wäre allerdings eine erhöhte Serverlast.

6.8.2 Verbesserung der Kommunikation zwischen Server und Client

Die direkte Übergabe des zu bearbeitenden RDF könnte durch eine Session-Übergabe ersetzt werden. Daraufhin kann der Flash-Client dann unter Angabe der Session den Namen des gewählten RDF und weitere Parameter holen. Diese Parameterübergabe kann wahlweise über die Flash-Funktion „loadVars()“ oder über ein XML-Format und XML:sendAndLoad realisiert werden.

7 Anhang

7.1 Anhang I - RDF-Klasse

```
1
2  /**
3
4   RDF Parser Klasse
5   erweitert die in Flash integrierte XML-Klasse
6
7   das erste Kind des root ist immer <?xml ...?>,
8   deshalb muss immer auf der 2. Kind zugegriffen werden,
9   welches dann den eigentlichen Root-Node enthält.
10
11 */
12
13 class RDFP extends XML {
14
15  /**
16    Variablen deklarieren
17  */
18  private var pfad;
19  private var wurzel;
20  private var namespaces;
21  private var tripelliste;
22  private var resourcenliste;
23  private var gezeichnetes;
24  private var tmc;
25  private var antwortxml;
26  private var infomc;
27
28  /**
29    Kontruktor
30  */
31  public function RDFP(){
32    this.ignoreWhite = true;
33    this.namespaces = Array();
34    this.tripelliste = Array();
35    this.resourcenliste = Array();
36    this.gezeichnetes = Array();
```

```
37    this.onLoad = this.init;
38   } // end of RDFP
39
40   /*
41    Dokument initialisieren
42   */
43   public function init(ladeErfolg) {
44    if(ladeErfolg) {
45     this.findRDFRoot();
46    } else {
47     // XML konnte nicht geladen werden...
48    }
49
50   } // end of init
51
52   /*
53    Zurueckgeben, ob Dokument snytaktisch korrektes XML enthält
54    und vollständig geladen wurde
55   */
56   public function isValid(){
57    if(this.status == 0) return true;
58    return false;
59   } // end of isValid
60
61   /*
62   Im Zielfilm werden 3 Methoden erwartet:
63   - drawRDFResource(titel)
64   - drawRDFPredicate(titel,subjekt,objekt)
65   - drawRDFObjekt(titel)
66   man muss natürlich erst einmal Subjekt und Objekt erstellen,
67   um dann das Prädikat dazwischen zu hängen
68   */
69   public function drawTriples(zielfilm){
70    var tripel, tripelnummer, rs;
71    var posnummer, pos, su, pr,ob, sumc, prmc, obmc;
72
73    this.tmc = zielfilm;
74
75    // Tripelliste durchgehen und anzeigen dabei Resourcen
76    // merken und ggf. als Subjekt verwenden, wenn man
77    // es schon als Objekt benutzt hat
78
79    for(tripelnummer in this.tripelliste) {
80     tripel = this.tripelliste[tripelnummer];
```

```
81
82      // Subjekt erstellen und merken
83      su = tripel.subject;
84
85      sumc = this.drawResource(su);
86      this.gezeichnetes[su] = sumc;
87
88
89      for(pr in tripel.pos){
90       ob = tripel.pos[pr];
91       //pr = this.getElementURI(pr);
92
93       // Überprüfen ob Objekt eine Resource ist
94       if(this.isResource(ob)) {
95        obmc = this.drawResource(ob);
96       } else {
97        obmc = this.drawLiteral(ob);
98       }
99       this.gezeichnetes[ob] = obmc;
100      //trace(pr +'___' + ob);
101      prmc = zielfilm.drawRDFPredicate(pr,sumc,obmc);
102      //trace(su + '::' + pr +'__'+ ob);
103     }
104    } // end for each resource...
105
106
107   } // end of drawTriples
108
109   /*
110    Zeichnen zuruecksetzten (in Vorbereitung auf neues Zeichnen)
111   */
112   public function resetDraw(){
113
114    this.tmc.currentX = 0;
115    this.tmc.prevX = 0;
116    this.tmc.currentY = 0;
117    this.tmc.prevY = 0;
118
119    this.tmc.resetDraw();
120
121   } // end of resetDraw
122
123   /*
124   Daten laden
```

```
125   */
126   public function load(quelle){
127    super.load(quelle);
128    this.pfad = quelle;
129   } // end of load
130
131   /*
132    RDF speichern. Zum Server senden und Antwort verarbeiten.
133   */
134   public function saveRDF(){
135    var tripelnummer, tripel;
136    var subject,po,pomc,sumc;
137    var wert, zieltyp,zielort;
138
139    var knoten,kinder,i;
140
141    // wenn keine RDF:Description oder keine Resourcen gefunden
142    // wurden dann einfach so zurück, ohne speichern
143    if(this.resourcenliste.length < 1) return true;
144
145    // GUI sperren
146    // unter root.tb sollte ein Infoclip bereitliegen,
147    // siehe dazu auch unten
148    _root.tb.disableForSaving();
149
150    // Alle Daten aus den grafischen Instanzen holen und
151    // ggf. das XML aktualisieren
152
153    for(tripelnummer in this.tripelliste) {
154
155     tripel = this.tripelliste[tripelnummer];
156
157     knoten = tripel['knoten'];
158
159     // jetzt haben wir wieder eines unserer Tripel
160     // das Objekt enthält, alles, was man zum Speichern braucht.
161
162     // der Subjektfilm lässt sich einfach so finden:
163     // trace("Subjektfilm " + this.gezeichnetes[tripel.subject]);
164
165     // Jetzt die Prädikat-Objekt-Kombinationen
166     for(po in tripel.pos) {
167      sumc =  this.gezeichnetes[tripel.pos[po]];
168
```

```
169       wert = sumc.getValue();
170       zieltyp = tripel.possaveto[po][0];
171       zielort = tripel.possaveto[po][1];
172
173       kinder = knoten.childNodes;
174       for(i=0;i<kinder.length;i++){
175        if(kinder[i].nodeName == zielort){
176         kinder[i].firstChild.nodeValue = wert;
177        }
178       }
179
180     }
181
182     this.wurzel.attributes['rdf:zielpfad'] = this.pfad;
183    }
184
185
186    this.antwortxml = new XML();
187    this.contentType = 'text/xml';
188    this.sendAndLoad(_root.speicherpfad,this.antwortxml);
189
190    this.antwortxml.onLoad = function(ladeErfolg){
191     var meinstatus;
192
193     /*
194       unter _root.tb muss ein Info-Clip bereitliegen:
195       Der Info-Movie-Clip muss 2 Funktionen besitzen:
196       displaySaveSuccess und displaySaveError
197       displaySaveSuccess wird ohne Paramter aufgerufen
198       displaySaveError bekommt eine anzuzeigende
199       Fehlermeldung als Parameter mit
200     */
201
202     // Unter wurzel.status sollte ein OK gefunden werden...
203     meinstatus = this.firstChild.firstChild.firstChild.nodeValue;
204     if( meinstatus == "OK") {
205      _root.tb.displaySaveSuccess();
206     } else {
207      _root.tb.displaySaveError(meinstatus.toString());
208     }
209    };
210
211   } // end of saveRDF
212
```

```
213
214 //#########################################################################
215 // END OF PUBLIC FUNCTIONS ----------------------------------------------
216 //#########################################################################
217
218
219 // GUI - related private Functions ---------------------------------A-
220
221 /*
222 Eine Resource zeichnen, erwartet in this.tmc einen Pointer auf
223 einen MC, welcher drawRDFResource(titel) enthält und einen
224 Pointer auf den erzeugten MC zurückgibt
225 */
226 private function drawResource(titel){
227  if(!this.isDrawn(titel)) {
228   return this.tmc.drawRDFResource(titel);
229  } else {
230   return this.getElementMC(titel);
231  }
232 } // end of drawResource
233 /*
234 Ein Literal zeichnen. Erwartet in this.tcm einen MC, welcher
235 drawRDFLiteral(titel) (@ret Pointer auf den erzeugten MC) enthält
236 */
237 private function drawLiteral(titel){
238  if(!this.isDrawn(titel)) {
239   return this.tmc.drawRDFLiteral(titel);
240  } else {
241   return this.getElementMC(titel);
242  }
243 } // end of drawLiteral
244
245 /*
246 Überprüfen, ob ein Element schon gezeichnet wurde
247 */
248 private function isDrawn(element){
249  var tmp;
250
251  for(tmp in this.gezeichnetes){
252   if(tmp == element) return true;
253  }
254  return false;
255 } // end of isDrawn
256
```

```
257  /*
258  schon gezeichneten MC eines Elements zurückgeben
259  */
260  private function getElementMC(element){
261   return this.gezeichnetes[element];
262  } // end of getElementMC
263
264  // GUI - related private Functions ------------------------------E-
265
266  // XML - related private Functions ------------------------------A-
267
268  /*
269  RDF-Root Node finden
270  */
271  private function findRDFRoot(){
272   var kinder;
273   var i;
274
275   kinder = this.childNodes;
276
277   for(i = 0; i < kinder.length; i++){
278    if(kinder[i].nodeName.toLowerCase() == 'rdf:rdf'){
279     // rdf root gefunden
280     // parsen und tripel suchen
281     this.wurzel = kinder[i];
282     this.parseRDFRoot(kinder[i]);
283
284    } // end if rdf root found
285   } // end for each childNode
286  } // end of findRDFRoot
287
288  /*
289   Das RDF Root Element parsen, d.h. xmlns-Deklarationen
290   suchen und im Objekt merken
291  */
292  private function parseRDFRoot(knoten){
293   var atts, attname,i;
294
295   // RDF Wurzel gefunden.
296   // Attribute lesen
297   atts = knoten.attributes;
298   for (attname in atts) {
299
300    // hier hofft man ja, einige Namespaces zu finden
```

```
301     // andere Attribute an der Wurzel werden im Moment ignoriert
302     if(attname.substr(0,5).toLowerCase() == 'xmlns'){
303      this.namespaces[attname.substr(6)] = atts[attname];
304     }
305    } // end namespace-declaration check
306
307    // Jetzt nach Tripeln suchen
308    this.checkDescriptions(knoten);
309
310   } // end of parseRDFRoot
311
312   /*
313   Nach Descriptions suchen
314   */
315   private function checkDescriptions(knoten){
316    var kinder, atts;
317    var i,j;
318
319    // Jedes Kind des Wurzelknotens durchgehen
320    kinder = knoten.childNodes;
321    for(i=0;i<kinder.length;i++){
322
323     // RDF-Descriptions müssen im Moment auch als solche
324     // im XML bezeichnet werden
325     if(kinder[i].nodeName.toLowerCase() == 'rdf:description'){
326      this.parseDescription(kinder[i]);
327     }
328    } // end for each childNode
329
330    // Debugmeldung
331    _root.debug.t1 += '\nXML durchsucht.';
332
333    // Debugmeldung wenn keine Resourcen im Objekt
334    if(this.resourcenliste.length < 1) {
335     tmpstring = "Es wurde keine rdf:Description gefunden.";
336     _root.tb.showErrorMessage(tmpstring);
337     _root.debug.t1 += '\nEs wurde KEINE rdf:description gefunden!';
338    } else {
339     //for(i in this.resourcenliste)
340     // _root.debug.t1 += 'R: ' + this.resourcenliste[i];
341    }
342   } // end of checkDescriptions
343
344   /*
```

```
345   Eine rdf:Descrption parsen und versuchen, ein Tripel zu finden
346   */
347   private function parseDescription(knoten){
348    var kinder, atts, attname, subkinder;
349    var i,j, tmp;
350    var tripel;
351
352    tripel = Array();
353    tripel['knoten'] = knoten;
354
355    // Attribute durchgehen
356    atts = knoten.attributes;
357    for (attname in atts) {
358     if(attname.toLowerCase() == 'rdf:about'){
359      tripel['subject'] = atts[attname];
360      tripel['saveto'] = Array('att',attname);
361      this.addResource(atts[attname]);
362      tripel['pos'] = Array();
363      tripel['possaveto'] = Array()
364     } else if(attname.toLowerCase() == 'rdf:id'){
365      tripel['subject'] = atts[attname];
366      tripel['saveto'] = Array('att',attname);
367      this.addResource(atts[attname]);
368      tripel['pos'] = Array();
369      tripel['possaveto'] = Array()
370     }
371     //trace(attname + ' : ' + atts[attname]);
372    } // end foreach att
373
374    // childNodes
375    kinder = knoten.childNodes
376    for(i=0;i<kinder.length;i++){
377     if(kinder[i].nodeName != null) {
378
379      // Kinder nach Objekten und Prädikaten durchsuchen
380      if(this.hasValidNameSpace(kinder[i].nodeName)) {
381
382        if(kinder[i].firstChild.nodeType == 3 && this.nodeIsChildless(kinder[i]))
383         tripel['pos'][kinder[i].nodeName] = kinder[i].firstChild.nodeValue;
384         tripel['possaveto'][kinder[i].nodeName] = Array('node',kinder[i].nodeName
385        } else {
386         // Wenn der Knoten Kinder hat, checken, ob eine Description dabei ist
387         // und ggf. Resource zeichnen.
388         subkinder = kinder[i].childNodes;
```

```
389          atts = subkinder[0].attributes;
390
391          for(attname in atts){
392           if(hasValidNameSpace(attname) && this.isResourceIndicator(attname)) {
393            tripel['pos'][kinder[i].nodeName] = atts[attname];
394            tripel['possaveto'][kinder[i].nodeName] = Array('att',attname);
395            this.addResource(atts[attname]);
396           }
397          }
398        } // end ifelse nodeIsChildless
399      }
400
401      //  Attribute abchecken
402      atts = kinder[i].attributes;
403      for(attname in atts){
404       if(hasValidNameSpace(attname)) {
405        tripel['pos'][kinder[i].nodeName] = atts[attname];
406        tripel['possaveto'][kinder[i].nodeName] = Array('att',attname);
407        if(this.isResourceIndicator(attname)) {
408         this.addResource(atts[attname]);
409        }
410        //trace(attname +' : '+ atts[attname]);
411       }
412      }
413
414      // Auch in den Kindern nach Descriptions suchen
415      this.checkDescriptions(kinder[i]);
416     }
417    } // end for each childNode
418
419    // SPOs im Objekt speichern
420    this.tripelliste.push(tripel);
421
422   } // end of parseDescrption
423
424   /*
425   Anhand des Namespaces eine Elementes eine komplette URL
426   zusammensetzen
427   */
428   private function getElementURI(element){
429    var namespace, rest, colpos;
430    var tmp;
431
432    // Position des : finden
```

```
433     colpos = element.indexOf(':');
434     namespace = element.substr(0,colpos);
435     rest = element.substr(colpos+1);
436
437     // Checken ob gefundener Namespace valide ist
438     for(tmp in this.namespaces){
439      // trace(correctNamespace +'=='+ namespace);
440      if(tmp == namespace) {
441       return this.namespaces[tmp] + rest;
442      }
443     }
444     // Namespace nicht gefunden, sollte ja *eigentlich* nicht vorkommen
445     return false;
446    } // end of getElementURI
447
448    /*
449    Überprüfen, ob ein Attribut eine Resource bezeichnet
450    */
451    private function isResourceIndicator(val){
452     // true zurückgeben wenn bestimmter Name gefunden wurde
453     if(val.toLowerCase() == 'rdf:resource') return true;
454     if(val.toLowerCase() == 'rdf:id') return true;
455     if(val.toLowerCase() == 'rdf:about') return true;
456
457
458     return false;
459    } // end of isResourceIndicator
460
461    /*
462    Überprüfen, ob übergebener Wert in Resourceliste des Objektes gespeichert
463    */
464    private function isResource(element){
465     var tmp;
466     for(tmp in this.resourcenliste){
467      if(resourcenliste[tmp] == element) return true;
468     }
469     return false;
470    } // end of isResource
471
472
473
474    /*
475
476    Checken, olb ein Knoten wirklich keine Kinder hat...
```

```
477
478   */
479   private function nodeIsChildless(knoten){
480    var kinder, counter, i;
481    counter = 0;
482
483    kinder = knoten.childNodes;
484    for(i=0;i<kinder.length;i++){
485     if(kinder[i].nodeName != null)
486      return false;
487    }
488    return true;
489   } // end of nodeIsChildless
490
491   /*
492   Ressource hinzufügen
493   vorher checken, ob sie schon in der Liste ist
494   */
495   private function addResource(rname){
496    var vorhanden;
497
498    for(vorhanden in this.resourcenliste){
499     if(resourcenliste[vorhanden] == rname) return true;
500    }
501    this.resourcenliste.push(rname);
502    return true;
503   } // end of addResource
504
505   /*
506   Überprüfen, ob der Namespace eine Knoten oder Attributes valide ist
507   */
508   private function hasValidNameSpace(theNodeName){
509    var correctNamespace, namespace;
510    var rest;
511    var colpos;
512
513    // Position des : finden
514    colpos = theNodeName.indexOf(':');
515    namespace = theNodeName.substr(0,colpos);
516    rest = theNodeName.substr(colpos+1);
517
518    // Checken ob gefundener Namespace valide ist
519    for(correctNamespace in this.namespaces){
520     // trace(correctNamespace +'=='+ namespace);
```

```
521     if(correctNamespace == namespace) return true;
522    }
523    return false;
524
525   } // end of hasValidNameSpace
526
527   // XML - related private Functions ----------------------------------------E-
528
529   /**
530   Einige Testausgaben
531   */
532   public function outputTests(){
533    var ret;
534    var kinder, atts, name;
535    var i, j, k;
536
537    ret = 'Ab hier Ausgabe';
538    kinder = this.childNodes;
539
540
541
542    // ret += 'Status : ' + this.status;
543    ret += this.toString();
544    ret += 'Ende der Ausgabe.';
545    return ret;
546   } // end of outputTests
547
548
549 }; // end of class RDFP
```

7.2 Anhang II - Verwendetes Beispiel-Vokabular

```
1
2 <?xml version="1.0"?>
3 <rdf:RDF xml:lang="en"
4     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
6
7   <rdfs:Class rdf:about="http://imaginers.de/rdf/elements/1.0/Resource">
8     <rdfs:isDefinedBy rdf:resource="http://imaginers.de/rdf/elements/1.0/"/>
9     <rdfs:label xml:lang="en">Web Resource</rdfs:label>
10    <rdfs:comment xml:lang="en">Web resource at imaginers.de</rdfs:comment>
11    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource" />
12  </rdfs:Class>
```

```
13
14    <rdf:Property rdf:about="http://imaginers.de/rdf/elements/1.0/Title">
15      <rdfs:isDefinedBy rdf:resource="http://imaginers.de/rdf/elements/1.0/"/>
16      <rdfs:label xml:lang="en">Title</rdfs:label>
17      <rdfs:comment>Title of the Resource</rdfs:comment>
18      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
19    </rdf:Property>
20
21    <rdf:Property rdf:about="http://imaginers.de/rdf/elements/1.0/Author">
22      <rdfs:isDefinedBy rdf:resource="http://imaginers.de/rdf/elements/1.0/"/>
23      <rdfs:label xml:lang="en">Author</rdfs:label>
24      <rdfs:comment>Author of the Resource</rdfs:comment>
25      <rdfs:range rdf:resource="http://imaginers.de/rdf/elements/1.0/Resource"/>
26      <rdfs:domain rdf:resource="http://imaginers.de/rdf/elements/1.0/Resource"/>
27    </rdf:Property>
28
29    <rdf:Property rdf:about="http://imaginers.de/rdf/elements/1.0/related">
30      <rdfs:isDefinedBy rdf:resource="http://imaginers.de/rdf/elements/1.0/"/>
31      <rdfs:label xml:lang="en">Related Resource</rdfs:label>
32      <rdfs:comment xml:lang="en">Resources within imaginers.de website</rdfs:comm
33      <rdfs:range rdf:resource="http://imaginers.de/rdf/elements/1.0/Resource"/>
34    </rdf:Property>
35
36    <rdf:Property rdf:about="http://imaginers.de/rdf/elements/1.0/type">
37      <rdfs:isDefinedBy rdf:resource="http://imaginers.de/rdf/elements/1.0/"/>
38      <rdfs:label xml:lang="en">Resource Type</rdfs:label>
39      <rdfs:comment>Type of Required Resource</rdfs:comment>
40      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
41    </rdf:Property>
42
43  </rdf:RDF>
```

Abbildungsverzeichnis

Literaturverzeichnis

[1] Semantic Web Official Homepage http://www.w3.org/2001/sw/

[2] Sir Tim Berners Lee, von vielen als der Erfinder des WWW bezeichnet. Alles wichtige über den Nobelpreisträger http://www.w3.org/People/Berners-Lee/

[3] CoolMax Materialinformation von Globetrotter http://www.globetrotter.de/de/beratung/mat_info_deta

[4] How much Information 2003 http://www.sims.berkeley.edu/research/projects/how-much-info-2003/execsum.htm

[5] Resource Description Framework Official Homepage http://www.w3.org/RDF/

[6] RDF Vocabulary Description Language 1.0: RDF Schema http://www.w3.org/TR/rdf-schema/

[7] http://www.dspace.org/ - DSpace ist eine digitale Bibliothek speziell für das akademische Umfeld und soll helfen, das geistiges Eigentum zu Speichern, Verwalten, Erhalten, Indizieren und wiederverwendbar zu machen.

[8] DAML+OIL http://www.daml.org/2001/03/daml+oil-index

[9] Web Ontology Language (OWL) http://w3.org/2004/OWL/

[10] World Wide Web Consortium http://www.w3.org/

[11] W3C RDF Validator http://www.w3.org/RDF/Validator/

[12] Shelley Powers „Practical RDF“, O'Reilly Verlag

[13] The global structure of an HTML document (25.05.2005) http://www.w3.org/TR/REC-html40/struct/global.html#h-7.4.4

[14] http://www.bitloeffel.de/DOC/2003/N3-Primer-20030415-de.html

[15] Interview mit Peter Norvig „Semantic Web Ontologies: What Works and What Doesn't“ Extract des Interviews http://www.alwaysson-network.com/comments.php?id=P7480_0_3_0_C; Reaktion von Tim Finin, Mitgleid einer Forschergruppe der University of Maryland Blog http://ebiquity.umbc.edu/v2.1/blogger/index.php?p=120

[16] Google Summer of Code 2005 http://code.google.com/summerofcode.html und Semedia Open Source Projects proposed for the Summer of Code http://semanticweb.deit.univpm.it/tiki-index.php?page=ProjectProposalPage