Masterarbeit

Titel der Masterarbeit

Multimedia Solutions for Digital Asset Management: Fedora Annotation Service

Verfasser

Bernd Pinter, BSc

angestrebter akademischer Grad

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2008

Studienkennzahl lt. Studienblatt:	A 066 935
Studienrichtung lt. Studienblatt:	Masterstudium Medieninformatik
Betreuer:	Univ.Prof. Dr.techn. Dipl.Ing. Wolfgang Klas

Zusammenfassung

Annotationen bzw. Kommentare in webbasierten Anwendungen, wie zum Beispiel in Digital Libraries, können dabei helfen bestehende Medieninhalte semantisch anzureichern. Allerdings sind in vielen aktuellen Systemen die Annotationsfeatures direkt in den Plattformen integriert, weshalb Änderungen oder die Hinzunahme neuer Medientypen sehr oft zu weitreichenden Änderungen des Gesamtsystems führen. Der Beitrag dieser Arbeit ist ein unabhängiges Annotations-Framework. Eine Serverkomponente verwaltet alle Annotationsdaten einheitlich, egal für welchen Medientypen sie bestimmt sind. Er kann Annotationsdaten in andere Formate transformieren, wodurch das Framework auch bestehenden Interoperabililitätskriterien gerecht wird. Jeder Medientyp benötigt einen eigenen Clienten, der ausschließlich mit dem Annotationsserver kommuniziert. So zieht die Erweiterung oder Hinzunahme neuer Medientypen keine Änderung des bereits bestehenden Systems nach sich. Auch lässt sich, im Gegensatz zur direkten Integration der Annotationsfeatures, ein solch unabhängiges Annotations-Framework leichter in ein bereits bestehendes System integrieren.

Abstract

Annotations or comments of web based applications, for example in digital libraries, can help to semantically enrich Media contents. However, the annotations features of present systems are often directly integrated into the platforms themselves. This means that changes or the addition of new mediatypes often have a deep impact on the whole system. The contribution of this thesis is an independent annotation-framework. A server component administrates uniformly all annotation data irrespective of the mediatype. The server is able to transform the data into other annotation standards so that the framework comes up to the existing criteria of interoperability. Each type of media needs its own client that communicates solely with the annotation-server. This way the enhancement or the addition of new mediatypes doesn't affect the existing system. In opposition to the direct integration of the annotation features an independent annotation-framework can be integrated more easily into an already established system.

Danksagung

Mein Dank gilt Univ. Prof. Dr.techn. Dipl.Ing Wolfgang Klas, Mag. Bernhard Haslhofer und Mag. Wolfgang Jochum für die freundliche und engagierte Betreuung. Alle drei halfen mir stets mit gutem Rat und Tat während allen Phasen der Entstehung dieser Diplomarbeit.

Besonders bedanken möchte ich mich bei meinem Vorgesetzten Prof. Dr. Rene Mayrhofer, der es mir ermöglichte in weniger angespannten Zeiten diese Arbeit weiter voranzutreiben. In Fragen zur wissenschaftlichen Arbeit beriet er mich kompentent und zugleich mit viel Herzblut.

Auch möchte ich mich herzlich bei allen Assistenten, Professoren und Kollegen des Austria Research Centers bedanken, die sich die Zeit nahmen und an der Abschlusspräsentation der Software Ende August 2008 teilnahmen. Große Teile des sechsten Kapitels resultierten direkt aus dieser stundenlangen Präsentation des Frameworks am Institut für Distributed and Multimedia Systems der Universität Wien. Eine angeregte und fruchtende Diskussion ermöglichte es, fehlende, notwendige oder wünschenswerte Features für ein Annotations-Framework aufzuzeigen.

Nicht zu kurz soll an dieser Stelle meine Freundin Mag[a] Julia Hinterseer kommen. Sie hatte während der gesamten Entstehung dieser Arbeit immer Verständnis und viel Geduld für mich aufgebracht.

Zuletzt danke ich meiner Familie — für alles.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

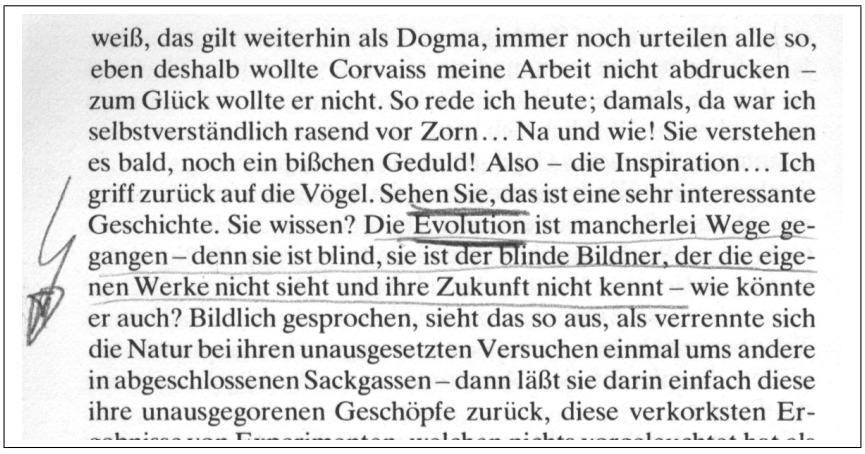
Einführung

Menschen sind es gewohnt, diverse Dokumente (Texte, Bilder) zu annotieren. In Büchern werden am Rand Notizen geschrieben, Passagen unterstrichen, Bilder oder für den einzelnen wichtige Bildbereiche werden umrandet und kommentiert. Auf bedrucktem Papier ist dies seit jeher ein natürlicher Vorgang, den das Medium – indirekt – unterstützt. Diese Notizen sind für den Einzelnen, aber auch für die Allgemeinheit von großem Nutzen.

Annotationen selbst stellen jedoch nicht nur einfache Kommentare der einzelnen Benutzern[1] dar. Sie können, je nach Blickwinkel, verschiedene Aufgaben erfüllen. Als erstes können Annotationen Informationen über bereits existierende Daten, zum Beispiel eines Bildes, sein. In diesem Fall sind die Annotationen selbst wieder Daten bzw. Ressourcen. Annotationen können auch dabei helfen neue Informationen über bereits bestehende Daten zu sammeln. Dadurch können sie die eigentlichen Daten erweitern. Annotationen können aber auch Daten über bestehende Daten beinhalten und sie können in diesem Fall als Metadaten betrachtet werden. Aber auch die Verbindung zwischen mehreren bestehenden Daten kann über eine Annotation erreicht werden.

Abbildung 1.1 zeigt eine Markierung bzw. eine Annotation in einem Buch. Der Leser hat durch einfaches unterstreichen einen bestimmten Textauszug hervorgehoben. Zusätzlich wurde das Wort 'Evolution' umrandet, um so schneller auf den Inhalt des unterstrichenen Bereiches hinzuweisen. Am linken Buchrand wurde noch ein Symbol gezeichnet – offenbar war diese Stelle dem Leser besonders wichtig.

[1] ≫Benutzer≪ steht in dieser Arbeit immer für Benutzerin und Benutzer. Generell wurde nahezu im gesamten Text die männliche Form verwendet, obschon der Text sowohl für Frauen als auch für Männer gelten soll.



weiß, das gilt weiterhin als Dogma, immer noch urteilen alle so, eben deshalb wollte Corvaiss meine Arbeit nicht abdrucken – zum Glück wollte er nicht. So rede ich heute; damals, da war ich selbstverständlich rasend vor Zorn... Na und wie! Sie verstehen es bald, noch ein bißchen Geduld! Also – die Inspiration... Ich griff zurück auf die Vögel. Sehen Sie, das ist eine sehr interessante Geschichte. Sie wissen? Die Evolution ist mancherlei Wege gegangen – denn sie ist blind, sie ist der blinde Bildner, der die eigenen Werke nicht sieht und ihre Zukunft nicht kennt – wie könnte er auch? Bildlich gesprochen, sieht das so aus, als verrennte sich die Natur bei ihren unausgesetzten Versuchen einmal ums andere in abgeschlossenen Sackgassen – dann läßt sie darin einfach diese ihre unausgegorenen Geschöpfe zurück, diese verkorksten Er-

Abbildung 1.1: *Annotationen in einem Buch. Der Text ist dem Kapitel 'Die Lympathersche Formel' aus dem Buch 'Nacht und Schimmel' von Stanisalw Lem [23] entnommen. Der Leser hat durch das Unterstreichen und durch das selbst gezeichnete Symbol am linken Buchrand eine, für ihn wichtige, Stelle im Text gekennzeichnet.*

Diese Arbeit beschäftigt sich mit der Frage, wie solche Annotationen[2] für unterschiedliche Medientypen in einer webbasierten Umgebung realisiert werden können.

1.1 Problemstellung

Immer mehr multimediale Inhalte finden Einzug ins Web. Waren es am Beginn des Internets nur Texte, so sind es nun immer öfter Bilder oder Videos. Unzählige Webseiten und Digital Libraries (z.B. die Bildplattform Flickr, die Videoplattform YouTube oder die Digital Library Encyclopedia of Chicago – vgl. Abschnitt 2.2 und Kapitel 5) sammeln und veröffentlichen ihre Inhalte im Web. Die Benutzer können diese multimedialen Inhalte meist nur betrachten. Ihnen sollte aber die Möglichkeit geboten werden, sich über diese multimedialen Inhalte zu *unterhalten*, sie zu *beschreiben* oder zu *kommentieren*.

Annotationen werden dazu verwendet das verteilte Wissen der Benutzer zu sammeln, indem sie den Benutzern die Möglichkeit geben ihre Gedanken und Kommentare zu einem bestimmtem multimedialen Objekt niederzuschreiben und diese im Web zu publizieren. Zum einen können auf dieses Art und Weise zusätzliche Informationen zu einem Multimedia-Objekt (z.B. einer historischen Aufnahme) gesammelt werden, zum anderen verstärkt diese

[2]Das Wort ≫Annotation≪ bedeutet 'Anmerkung', 'Beifügung', 'Hinzufügung', aber auch 'Aufzeichnung' (veraltet) oder 'Vermerk'. [43] [10]

Interaktion die Bindung der einzelnen User an die Webseite. Dies wiederum verstärkt die Bildung einer Online-Community.

Heutzutage werden meist die Annotations-Features direkt in die Webseiten integriert. Dies ermöglicht zwar eine exakte Abstimmung der Annotation bzw. deren Daten zu einem bestimmten Thema (vgl. Abschnitt 2.2). Werden allerdings neue Medientypen (z.B. Videos) in solch ein starres System eingeführt, zieht dies unweigerlich große Änderungen am bestehenden System (Servercode, Daten, Clientcode) nach sich.

Was also fehlt ist ein *allgemeines* Annotations-Framework, das in der Lage ist mit den verschiedensten Medientypen umzugehen. Ein Framework das sowohl das Abfragen bzw. Speichern als auch das Anzeigen der Annotationsdaten übernehmen kann. Wobei gerade die Hinzunahme eines neuen Medientypen nicht zur Änderung des Gesamtsystems führen darf. Ein solches allgemeines Annotations-Framework muss auch bestehende Interoperabilitätskriterien erfüllen, damit es seine Daten mit anderen Frameworks austauschen kann.

1.2 Beitrag

Der Beitrag dieser Diplomarbeit liegt in der Konzeption und Umsetzung eines eigenständigen Annotations-Frameworks, welches unterschiedliche Medientypen unterstützt [2]. Dieses Annotation-Framework muss unabhängig von jenem System arbeiten, in das es eingebettet wird. Das System, in dem das Annotations-Framework eingebettet ist, wird im weiteren Text *Ursprungssystem* genannt.

Das Framework muss in der Lage sein mit den verschiedensten Medientypen (Text, Bilder, Videos, ...) umzugehen. Die Änderung oder die Hinzunahme eines neuen Medientyps darf zu keiner Änderung des bereits bestehenden Systems führen. Dies gilt sowohl für das Ursprungssystem als auch für das Framework selbst. Das System muss interoperabel sein und seine Daten mit anderen Annotations-Frameworks (z.B. Annotea) austauschen können.

Um diesen Anforderungen gerecht zu werden, besteht das *Fedora Annotation Service*[3] aus zwei unabhängigen Komponenten:

- Annotation-Server
- Annotation-Clients für Bild- und Video-Annotationen

[3] *Fedora Annotation Service* ist der Name des, im Zuge dieser Diplomarbeit implementierten, Annotations-Frameworks.

Annotations-Metadaten stellen alle benötigten Felder für eine Annotation bereit. Die Felder der Metadaten sind vom jeweiligen Medientyp abhängig. Der Server übernimmt die Aufgabe, die Annotationen zu speichern und zu verwalten. Außerdem kann er eine Liste aller Annotationen zu einem Medien-Element liefern. Der Client liest die Metadaten von dem Server und stellt sie dar. Dabei ist der Client vor allem vom verwendeten Medientyp abhängig. Sein Verhalten und sein Aussehen werden großteils von dem annotierten Objekt bestimmt: ein Bild zu annotieren funktioniert grundsätzlich anders als einen Text zu annotieren. Aus diesem Grund gibt es für jeden einzelnen Medientyp einen eigenen Clienten.

1.3 Abgrenzung

Ziel der Arbeit ist die Entwicklung eines Annotations–Frameworks. Es sollen nicht alle denkbaren Clienten für die verschiedenen Medientypen implementiert werden. Die Anwendbarkeit des Frameworks wird jedoch anhand von zwei Clienten gezeigt: ein Client für Bilder und ein Client für Videos.

Aber auch in diesen beiden Fällen verfügen die Clienten über eingeschränkte Funktionalität. Der Bild-Client kann derzeit nur Bilder im Format jpg, png und gif verarbeiten. Der Video-Client kann im Moment nur das Flash-Videoformat[4] darstellen und annotieren. Diese Einschränkungen resultieren direkt aus der Verwendung des Flash9 Webbrowser Plugins für die beiden Clienten.

1.4 Überblick

Im folgenden wird ein kurzer Überblick über die Struktur und den Inhalt dieser schriftlichen Arbeit gegeben:

Kapitel 1 – Einführung – gibt einen kurzen Überblick über Annotationen. Die Problemstellung, der Beitrag und die Abgrenzung dieser Arbeit werden behandelt.

Kapitel 2 – Hintergrund – ist in drei Unterkapitel geteilt. Teil eins beschäftigt sich eingehender mit der Frage, was Annotationen sind, welche Formen sie annehmen können und wie sie in Relation zu Digital Libraries stehen. Teil zwei ist eine state-of-the-art Analyse und gibt einen Überblick über einige gängige Web Annotationstools bzw. -Frameworks, die auch für Bilder

[4]Das Flash-Video-Format kann vom Flash-Plugin ab der Version 6 verarbeitet werden, wobei die folgenden Video-Codecs unterstützt werden: Sorenson, VP6 und MPEG–4 (nach dem H.264 Standard). [45] [4]

und/oder Video geeignet sind. Die Ansätze der einzelnen Tools werden aufgezeigt, miteinander verglichen und beurteilt. Eine Matrix stellt nochmals alle Tools und ihre Möglichkeiten übersichtlich dar, damit sie untereinander vergleichbarer werden. Teil drei beschreibt das verwendete Backend – das Fedora-Repository.

Kapitel 3 – Das Fedora-Annotation-Service – beschreibt die prinzipielle Herangehensweise an das Problem. Die Konzeption der Gesamtarchitektur und Designentscheidungen werden in diesem Kapitel behandelt.

Kapitel 4 – Implementierung – beschreibt die einzelnen Teile des programmierten Frameworks im Detail. Dieses Kapitel besteht aus drei Teilen: Jeder Teil des Frameworks (Server, Client) wird separat behandelt. Den Abschluss des Kapitels bildet ein nochmaliger tabellarischer Vergleich zwischen den in Kapitel 2 besprochenen Annotationstools. Zusätzlich zu diesen Tools wird hier auch das implementierte Annotations-Framework mit den beiden Clienten (für Bild bzw. Video) in den Vergleich mit aufgenommen. Die Praxistauglichkeit des implementierten Frameworks und der beiden Clienten kann so auf einem Blick überprüft werden.

Kapitel 5 – Fallstudie – beschreibt ein typisches Szenario in dem das implementierte Annotations-Framework zur Anwendung kommen könnte. Es wird gezeigt wie sich das Framework in ein bereits bestehendes System integrieren lässt.

Kapitel 6 – Mögliche Erweiterungen – zeigt neue bzw. fehlende Features auf, die in der prototypischen Entwicklung des Frameworks fehlen oder bewusst ausgelassen wurden.

Kapitel 7 – Zusammenfassung und Schlussfolgerungen – gibt eine Zusammenfassung und zieht eine Schlussfolgerung über die gesamte Arbeit.

Kapitel 2

Hintergrund

Dieses Kapitel beschäftigt sich mit dem Hintergrund von Annotationen. Als erstes werden die Begriffe Digital Libraries (digitale Bibliotheken) und Annotationen genauer erklärt. Im zweiten Teil werden zehn aktuelle Annotations-Tools vorgestellt, wobei die Tools nach Ihrem Verwendungszweck gruppiert sind. Es werden dabei jeweils Web-Annotationen, Bild-Annotationen und Video-Annotationen direkt miteinander verglichen. Der dritte und letzte Teil des Kapitels stellt das verwendete Backend, das Fedora-Repository, vor. Die Funktionsweise und die, für das Annotations-Framework wichtigsten Features werden besprochen.

2.1 Digital Libraries und Annotationen

Annotationen werden oft in Verbindung mit Digital Libraries verwendet. Eine Digital Library ist aber nicht einfach nur eine Onlineversion einer herkömmlichen Bibliothek – ihre Aufgaben sind vielfältiger: sie müssen die Inhalte (Content) speichern, aufbereiten und auch präsentieren. Ebenso wird oft eine ausgefeilte Suche angeboten bzw. erwartet. Und nicht zu vergessen sei der Umstand, dass Datenformate sich im Laufe einer, oftmals sehr kurzen, Zeitspanne ändern. Digital Libraries müssen auch diesem Umstand Rechnung tragen und dafür sorgen, dass die Daten auch zu einem späteren Zeitpunkt noch les- und vor allem interpretierbar sind.

Eine eindeutige Formale Definition für Digital Libraries kann leider nicht so einfach gegeben werden, da eine solche Definition stark vom jeweiligen Standpunkt abhängt. Agosti et al. bieten in [1] zwei brauchbare Definitionen. Die erste Definition, aus der Sicht der Informatik, lautet:

> Digital Libraries are concerned with the creation and management of information resources, the movement of information across global networks and the effective use of this information by a wide range of users.

Bibliothekare haben naturgemäß eine andere Sicht auf diese Dinge. Dieser Umstand wird in der zweiten Definition sichtbar:

> Digital Libraries are organisations that provide the resources, including the specialised stuff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use by a defined community or set of communities.

So unterschiedlich diese beide Definitionen auch sein mögen, haben sie doch eine Gemeinsamkeit: Informationen sollen für die Benutzer abrufbar sein.

Annotationen können in verschiedenen Anwendungen, je nach Blickpunkt, unterschiedliche Bedeutungen haben [1] [27] [42]. Deshalb werden im folgenden drei unterschiedliche Betrachtungen bzw. Bedeutungen von Annotationen behandelt.

Annotationen als Erweiterung der Medieninhalte
Wird eine Annotation selbst als ein Medieninhalt betrachtet, so entspricht dies einer *Informationsspezifischen Betrachtung*. Die Annotationen können so die eigentlichen Medieninhalte, die sie annotieren, anreichern und erweitern. Sie erlauben auch das Anlegen von neuen semantischen Relationen zwischen mehreren unabhängigen Medieninhalten, indem zum Beispiel eine Annotation von einem Bild auf ein anderes Bild verweist.

Annotation als Metadaten über Medieninhalte
Wird eine Annotation so aufgefasst, dass sie zusätzliche Daten für einen Medieninhalt bereit stellt, dann entspricht dies einer *Datenspezifischen Betrachtung*. Eine Hauptcharakteristik von Metadaten ist die Verbindung zwischen ihnen und dem Objekt über das sie etwas aussagen sollen. Diese Verbindung gibt es auch bei einer Annotation, die ein bestimmtes Medienobjekt annotiert. In dem Sinne kann bei Annotationen auch von Metadaten über einen Medieninhalt gesprochen werden.

Annotationen als Dialog
Die *Kommunikationsspezifische Betrachtung* rückt die Frage nach der Intention des Benutzers in den Mittelpunkt. Es soll hier jedoch nicht um Sinn oder Unsinn einer einzelnen Annotation gehen, sondern vielmehr um den Akt des

annotierens selbst. Jede Annotation, sofern sie nicht privat ist, stellt implizit eine Kommunikation mit der Allgemeinheit dar – also einen Dialog.

Durch Annotationen haben die Benutzer die Möglichkeit aktiv zu werden. Sie können zu Medieninhalten Kommentare abgeben, sind also nicht nur auf das passive Betrachten der Inhalte beschränkt. Dieses "aktiv werden" bringt die Digital Libraries und die Benutzer enger zusammen, wovon beide profitieren können.

Wie hoch der Profit durch die Annotationen für die Allgemeinheit werden kann zeigt Marshall in einer von ihr durchgeführten Feldstudie sehr gut auf [24]. In dieser Studie wurden StudentInnen beim Kauf von Lehrbüchern beobachtet. Die StudentInnen hatten dabei die Möglichkeit ein neues Buch zu kaufen oder auf ein gebrauchtes Exemplar des letzten Jahres zurückzugreifen. Fast alle gebrauchten Bücher wurden dabei bereits von einem der Vorbesitzer in irgendeiner Weise annotiert: Sätze waren ganz oder teilweise unterstrichen oder am Buchrand wurden Notizen bzw. Symbole angebracht. Diese Annotationen haben auf den ersten Blick nur für diejenige Person einen Sinn, die auch die Notiz verfasst hat. Aber es zeigte sich in der Feldstudie das eine Mehrzahl der StudentInnen gerade auf diese alten Bücher zurückgegriffen hat. Der Grund dafür war aber nicht der etwas geringere Preis gegenüber der neuen Ausgabe. Vielmehr waren es die Notizen, also die Annotationen, die diese alten Bücher so attraktiv machten. Die StudentInnen konnten so, ohne alles genau lesen zu müssen, sich auf die wesentlichen Kapitel oder gar nur Sätze konzentrieren. Die bestehenden Annotationen ergaben bereits eine gute Zusammenfassung bzw. den richtigen Fokus auf die wirklich wichtigen Teile des Buches. Daran ist zu erkennen, dass anfangs private Annotationen in weiterer Folge für die Allgemeinheit ein wertvolles Gut werden können. Diese Erkenntnis deckt sich auch mit den Ergebnissen anderer Feldstudien [3] [25].

2.2 Annotationstools: Eine state-of-the-art Analyse

Dieses Unterkapitel gibt einen Überblick über einige, derzeit am Markt befindliche, Annotationstools. Das Hauptaugenmerk liegt dabei auf Tools die in der Lage sind Bilder und/oder Videos zu annotieren. Dennoch werden auch Frameworks vorgestellt, die auf das Annotieren von Webseiten bzw. Texte spezialisiert sind. Die einzelnen Tools bzw. Frameworks werden dabei nach Ihren Möglichkeiten bezüglich der zu annotierenden Medienelemente (Text, Bilder, Videos) gruppiert. Den Abschluss bildet eine Matrix in der nochmals alle hier vorgestellten Tools miteinander verglichen werden. Dies soll einen einfachen Überblick über die Features geben.

2.2.1 Website Annotationstools

Diese Klasse von Annotationstools sind darauf spezialisiert ganze oder einzelne Bereiche einer Webseite zu annotieren.

Annotea / Amaya

Annotea[1] ist ein vom W3C-Konsortium standardisiertes Annotations-Framework [12] [14] [15]. Annotea besteht aus einem Server, der die Daten speichert, einem Clienten, der es den Benutzern erlaubt einzelne Bereiche von einer Webseite zu annotieren und einem Netzwerkprotokoll, das den Transport der Daten regelt. Annotea selbst ist darauf spezialisiert einzelne Textbereiche von Webseiten zu annotieren, ist aber nicht darauf beschränkt. Da Annotea RDF als Datenbasis und XPointer für die Auswahl eines Bereichs nutzt, ist es auch möglich andere Daten als Text zu markieren (z.B. Teile eines Bildes). Amaya[2] ist ein Client für Annotea und auf das Annotieren von HTML- bzw. XHTML-Dokumenten spezialisiert. Abbildung 2.1 zeigt den Amaya-Clienten von Annotea.

WebAnn

Bei WebAnn handelt es sich um ein Plugin für den Internet-Explorer von der Firma Microsoft. WebAnn ermöglicht es seinen Benutzern Bereiche eines Textes (einer Webseite) mit der Maus zu markieren und diesen Bereich mit einem Kommentar zu versehen. Die Annotationen können privat sein oder mit einer Community geteilt werden. Sämtliche Annotationen zu der gerade besuchten Webseite befinden sich in einer eigenen Leiste im Internet-Explorer, die auch über eine Suche verfügt. Zusätzlich können die Benutzer über WebAnn miteinander diskutieren. Marshall und Brush verwendeten in einer Feldstudie über Annotationen dieses Tool [25]. Die Entwicklung von WebAnn wurde allerdings eingestellt. Abbildung 2.2 zeigt das WebAnn-Plugin mit seinen Möglichkeiten.

Sticki-Notes

Sticki-Notes[3] bietet in etwa die selbe Funktionalität wie WebAnn. Es können jedoch nur Kommentare zu einer ganzen Webseiten abgegeben werden, aber

[1] http://annotea.org
[2] http://www.w3.org/Amaya
[3] http://www.stickis.com

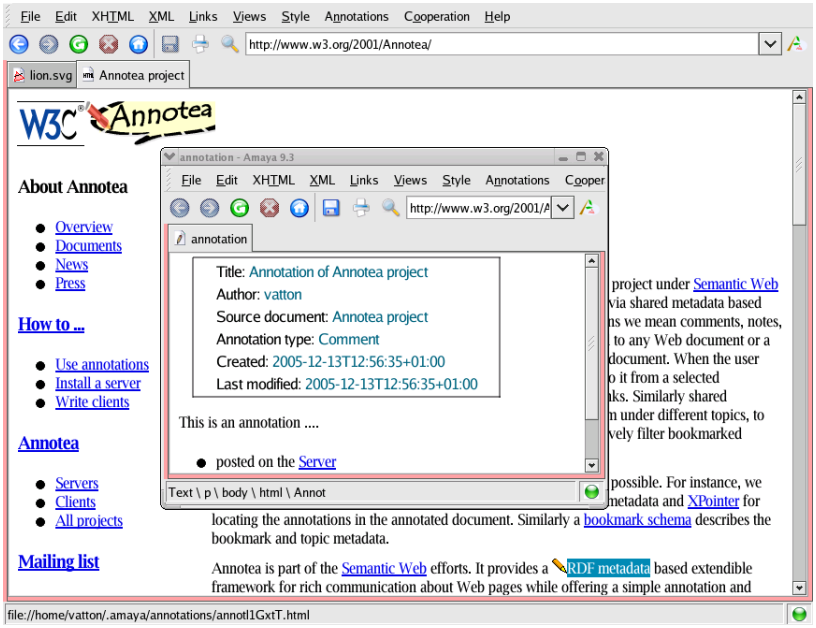


Abbildung 2.1: *Annotationen bei Annotea. Der Annotationsclient Amaya ist in der Lage ganze Webseiten oder nur einzelne Bereiche daraus zu annotieren. Annotea nutzt ein RDF-Schema für die Daten und XPointer um den markierten Bereich zu lokalisieren. Prinzipiell ist es mit Annotea auch möglich andere Medieninhalte als HTML- oder XHTML-Text zu annotieren. Allerdings unterstützt dies Amaya nicht.*

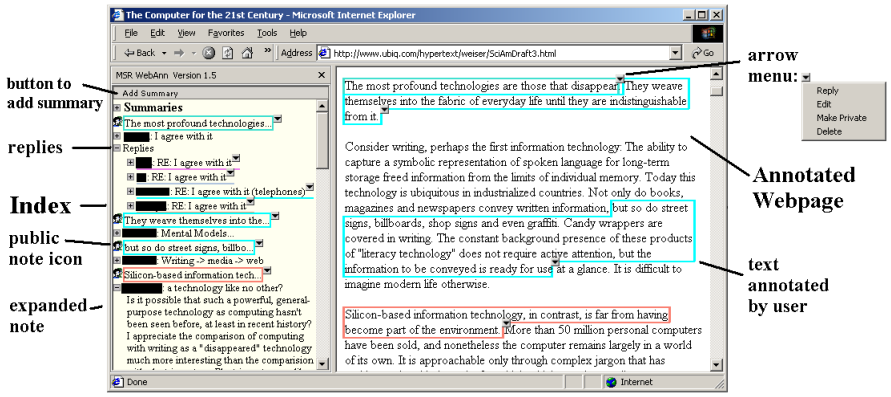


Abbildung 2.2: *Annotationen bei WebAnn. Die Benutzer können Teile einer Webseite annotieren und darüber diskutieren.*

Abbildung 2.3: *Annotationen bei Sticki-Notes. Die Benutzer können nur ganze Webseiten annotieren. Die Kommentare werden im Webbrowser als kleine Fenster (Stickis) über den Text gelegt und sind für die gesamte Sticki-Community sichtbar.*

dafür können die Benutzer über die einzelnen Kommentare eine Diskussion führen. Die Daten werden auf einem zentralen Server gespeichert und stehen allen registrierten Sticki-Benutzern zur Verfügung. Die einzelnen Kommentare bzw. Diskussionen werden in kleinen Fenstern (den sogenannten Stickis) gesammelt. Diese werden über der Webseite als kleine Fenster eingeblendet. Abbildung 2.3 zeigt eine Webseite, die über zwei Stickis verfügt.

Wikalong

Wikalong[4] ist eine Reimplementierung von WebAnn für den Mozilla Firefox Webbrowser. Auch Wikalong bietet, so wie WebAnn, seinen Benutzern die Möglichkeit einzelne Textpassagen einer Webseite zu annotieren. Eine Annotation kann dabei nur für den Autor oder aber für alle Benutzer sichtbar sein. Auf der linken Seite des Browsers wird eine Leiste eingeblendet, die alle (für den einzelnen Benutzer sichtbaren) Annotationen zu der gerade besuchten Webseite anzeigt. Auch hier gibt es eine Suchfunktion, um eine bestimm-

[4] http://www.wikalong.org

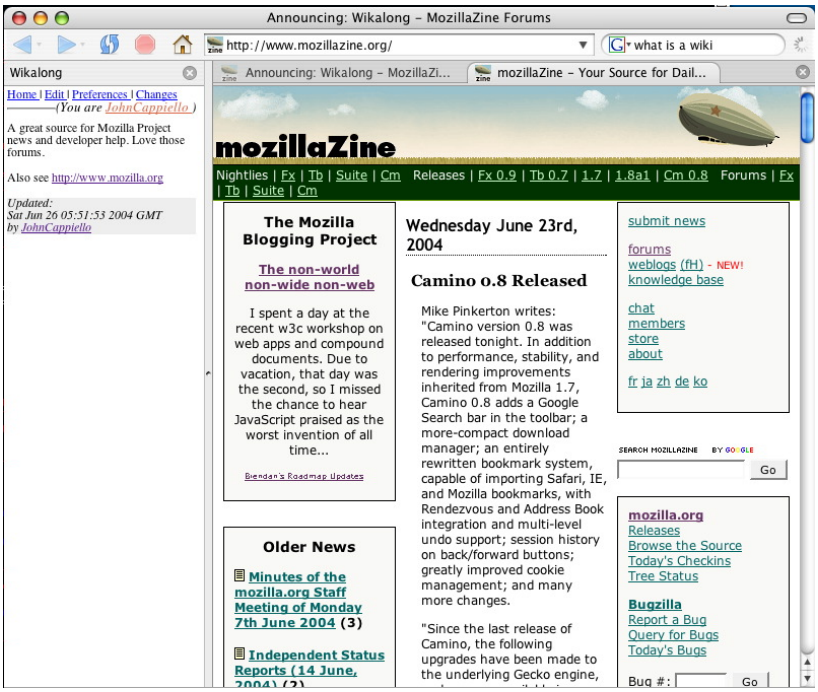


Abbildung 2.4: *Annotationen bei Wikalong. Die Benutzer können ganze Webseiten oder nur einzelne Passagen annotieren. Auf der linken Seite des Firefox-Webbrowser befindet sich eine Leiste, die alle Annotationen dieser Webseite anzeigt.*

te Annotation wiederzufinden. Abbildung 2.4 zeigt den Firefox-Webbrowser mit der Wikalong-Leiste.

2.2.2 Bild Annotationstools

Diese Klasse von Annotationstools haben sich auf das Annotieren von ganzen Bildern oder einzelnen Bildbereichen spezialisiert.

Flickr

Flickr[5] ist derzeit eine sehr beliebte Plattform um seine eigenen Fotos und Bilder online zu stellen und sie so mit der Allgemeinheit zu teilen. Die Benutzer haben die Möglichkeit Kommentare zu den Bildern abzugeben. Seit geraumer Zeit können auch die Bilder direkt mit Notizen versehen werden.

[5] http://www.flickr.com

Diese Notizen erlauben es einzelne Bereiche (Fragmente) des Bildes zu markieren, sind allerdings auf die Form eines Rechtecks beschränkt. Abbildung 2.5(a) zeigt ein Foto mit seinen Notizen, wobei nur der Text jener Notiz sichtbar ist, über dem auch der Mauszeiger steht. In Abbildung 2.5(b) wird das Anlegen bzw. Ändern einer Notiz mit einer Bereichsauswahl gezeigt.

Fotonotes

Fotonotes[6] ist ein OpenSource Framework zum Annotieren von Bildern. Das Paket besteht aus einen Serverteil, der in PHP4/5 geschrieben wurde und einem Clienten der auf Javascript aufbaut. Das Tool arbeitet ausschließlich mit JPEG-Bildern zusammen, da es die einzelnen Annotationen direkt im JPEG-Header des Bildes als XML ablegt. Der Funktionsumfang ist nahezu identisch mit den Notizen von Flickr – die Entwickler selbst sprechen von der besten Ableitung der Flickr-Notes. Dementsprechend kann Fotonotes, ebenso wie Flickr, einzelne rechteckige Bildbereiche annotieren. In Abbildung 2.6(a) sieht man einzelne annotierte Bildbereiche und in Abbildung 2.6(b) wird das Anlegen bzw. Ändern einer Annotation gezeigt.

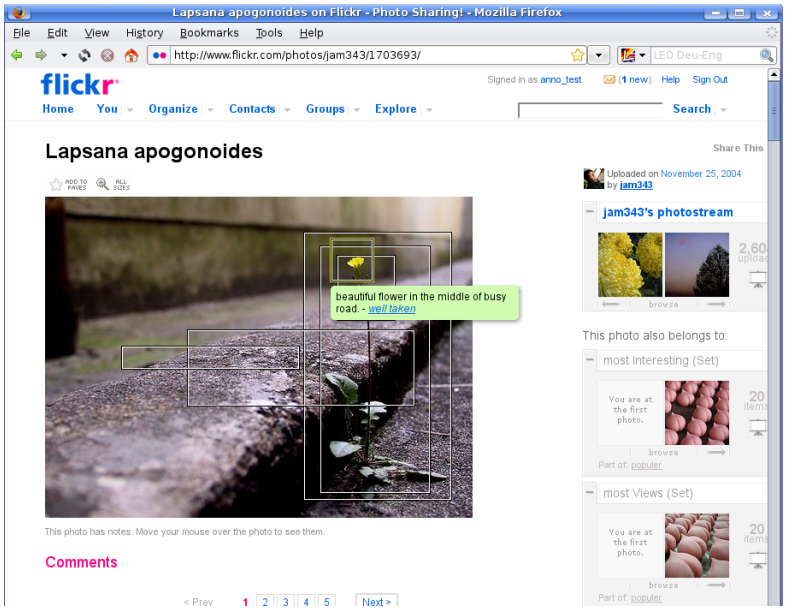
TelPlus Projekt: Image-Annotation

Das ARC (Austrian Research Center) hat im Zuge des TelPlus Projekts[7] ein Tool zur Bildannotation[8] implementiert. Dieses Tool ist in der Lage verschiedene Bildbereiche zu markieren. Diese Bereiche sind jedoch nicht nur auf Rechtecke beschränkt, sondern können auch Ellipsen, ein Fadenkreuz, ein Polygonzug oder eine Freihandzeichnung sein. Des weiteren kann der Benutzer eine Farbe und die Liniendicke für seine Annotation bestimmen. Auch eine Suche über alle Annotationen ist vorhanden. Abbildung 2.7(a) zeigt das Annotationstool, in dem verschiedene Annotationen und zu sehen sind. Abbildung 2.7(b) zeigt wieder das Ändern einer Annotation. In der Abbildung 2.8 wird die Ergebnisliste einer Suche gezeigt.

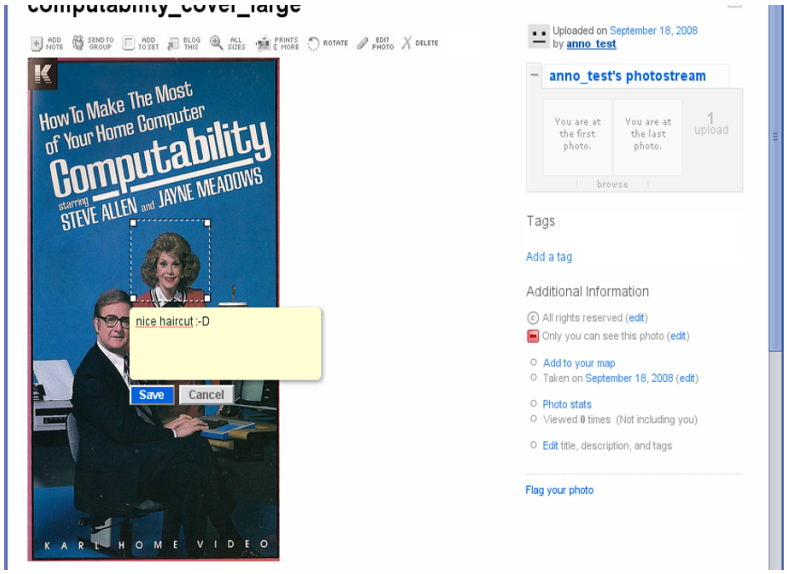
[6] http://www.fotonotes.net

[7] Das TelPlus Projekt ist ein Baustein des Europeana Projekts. Dieses wurde von der EU-Kommission ins Leben gerufen. Siehe auch http://www.theeuropeanlibrary.org/telplus

[8] Der TelPlus Annotationsclient befindet sich derzeit noch in der Implementierungs- und Testphase. Siehe http://dme.arcs.ac.at/image-annotation-frontend/annotate.jsp?↩
user=csa1980&objectURL=http://wienwoche.files.wordpress.com↩
/2007/10/tower_wien_schwechat.jpg&db=tel

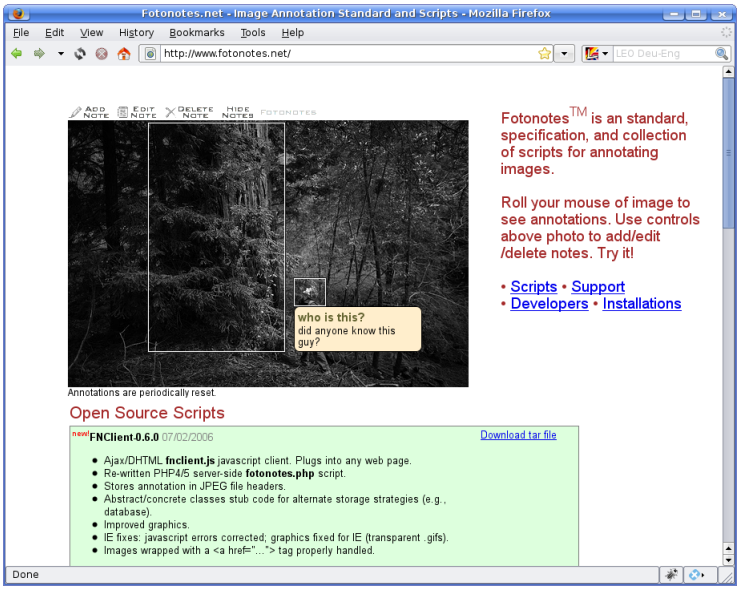


(a) Ein Foto mit mehreren Notizen

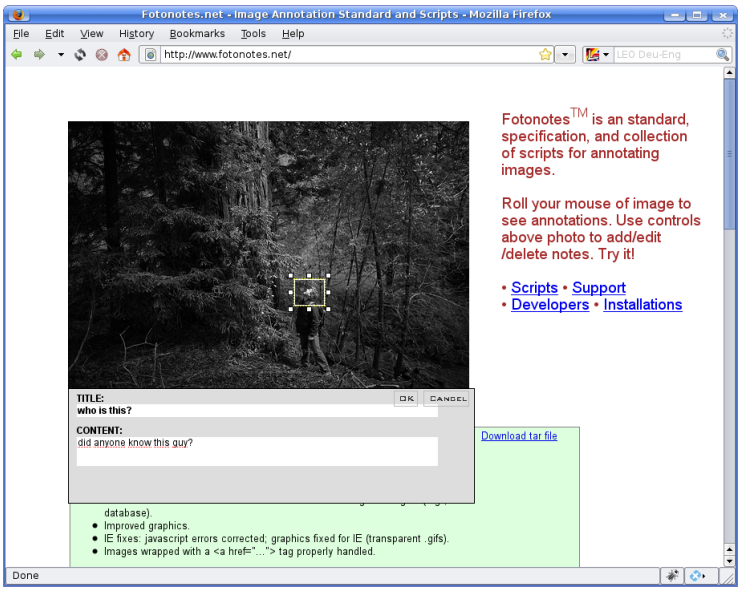


(b) Das Anlegen bzw. Ändern einer neuen Notiz

Abbildung 2.5: *Annotationen bei Flickr. Normale Kommentare gelten für das gesamte Bild und erlauben eine Diskussion darüber zu führen. Notizen, so wie sie hier zu sehen sind, erlauben es den Benutzern sogar einzelne Bildbereiche zu annotieren.*

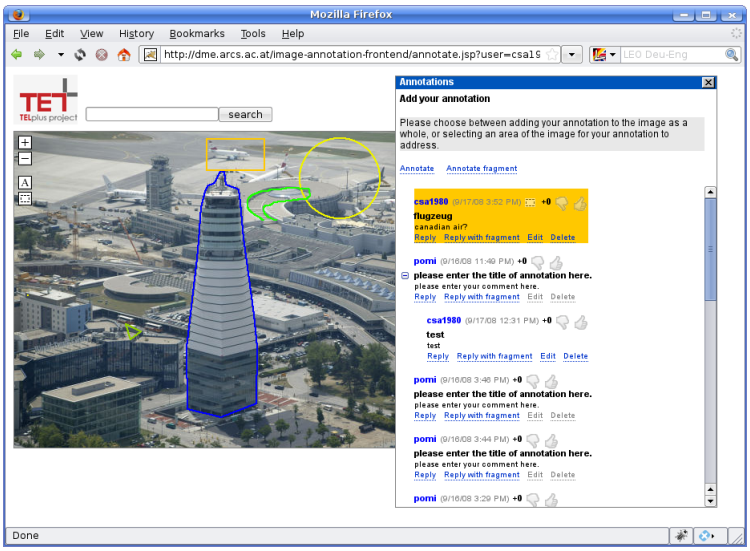


(a) Ein Foto mit mehreren annotierten Bildbereichen

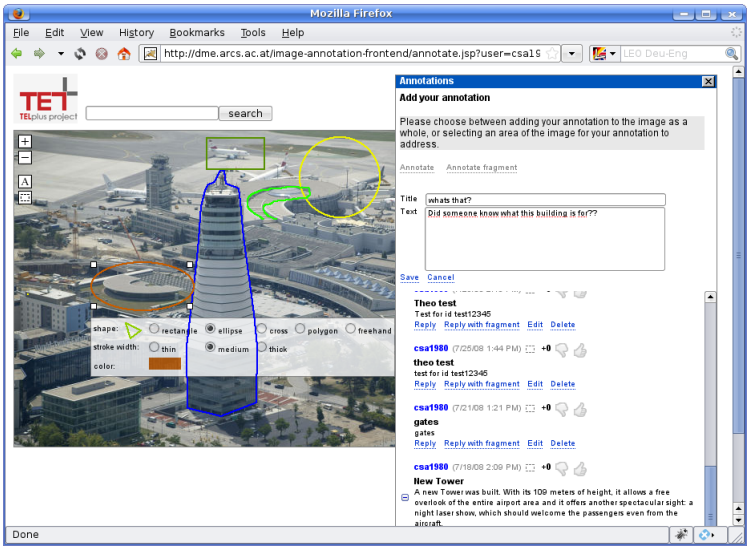


(b) Das Anlegen bzw. Ändern einer Annotation

Abbildung 2.6: *Annotationen bei Fotonotes. Das Framework besteht aus einem Server und einem Javascript-Clienten. Die Annotationsdaten werden im JPEG-Header als XML-Daten gespeichert.*



(a) Ein Foto mit mehreren annotierten Bildbereichen



(b) Das Anlegen bzw. Ändern einer Annotation

Abbildung 2.7: *Annotationen bei TelPlus. Die Abbildung (a) zeigt den Clienten, bei dem die Annotationen verschiedene Formen (Rechteck, Ellipse, Fadenkreuz, Polygonzug, Freihand) und Farben haben können. Abbildung (b) zeigt das Ändern einer Annotation.*

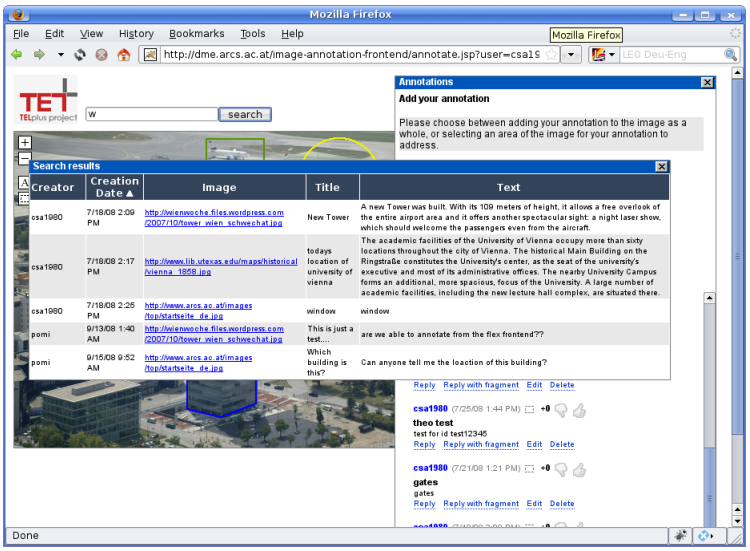


Abbildung 2.8: *Suche nach Annotationen bei TelPlus.*

2.2.3 Video Annotationstools

Diese Klasse von Annotationstools haben sich auf das Annotieren von Videos spezialisiert. Der Funktionsumfang der einzelnen Tools geht hier allerdings weit auseinander, denn Videos bestehen aus drei Dimensionen: zwei Dimensionen für das Bild und eine Dimension für die Zeitachse. Keines der hier vorgestellten Tools ist in Lage alle drei Dimensionen gleichberechtigt zu annotieren.

YouTube

YouTube[9] ist eine Plattform auf der die Benutzer ihre eigenen Kurzvideos online stellen können. YouTube ist für Video das, was Flickr für Fotos ist. Auch hier waren zu Beginn die Kommentare nur für das ganze Video gedacht. Mittlerweile können aber die Benutzer einzelne Teile eines Video mit Marken versehen. Diese Marken können im Bild auf einen bestimmten Punkt zeigen (mit Hilfe einer Sprechblase, wie sie auch aus Comics bekannt sind) jedoch keinen Bildbereich markieren. Oder sie kann für das ganze Bild gelten. In diesem Fall wird einfach nur ein kleines Fenster am unteren Rand des Bildes angezeigt. Auf der Zeitachse hat eine Marke eine Startzeit und

[9] http://www.youtube.com

Abbildung 2.9: *Annotationen bei YouTube. Die Benutzer können Marken setzen, die eine Start- und eine Endzeit haben. Leider sind diese zwei Punkte nicht am Playhead ersichtlich, weshalb es beim Betrachten eines Video für die Benutzer so wirken kann als ob die Sprechblasen willkürlich erscheinen und wieder verschwinden. Des weiteren kann bei YouTube kein Bereich im Bild ausgewählt werden. Es kann nur auf einem Punkt im Bild mit dem Pfeil der Sprechblasen hingewiesen werden.*

eine Endzeit. Allerdings sind diese zwei Zeitpunkte nicht am Playhead[10] ersichtlich. Insofern wirkt es beim Betrachten eines Videos so, als ob sich die Marken willkürlich ein- bzw. ausblenden. Abbildung 2.9 zeigt ein YouTube-Video mit Annotationen, wobei die Sprechblase auf einen bestimmten Punkt im Bild zeigt. Am unteren Rand des Videos ist ein eingeblendetes Fenster zu sehen, das für das gesamte Bild steht.

Viddler

Bei Viddler[11] handelt es sich, wie bei YouTube, um eine Community-Plattform auf der die Benutzer ihre eigenen Videos mit der Allgemeinheit teilen

[10]Playhead: jener Bereich des Video-Players, auf dem eine Markierung anzeigt, wo sich das Video auf der Zeitachse befindet. Üblicherweise befindet sich der Playhead direkt unterhalb des Bildes.

[11]http://www.viddler.com

können. Viddler kann keine Bildbereiche markieren, dafür kann aber ein Zeitpunkt auf am Playhead markiert werden. Diese markierten Zeitpunkte werden, im Gegensatz zu YouTube, auch auf der Zeitachse angezeigt. So sehen die Benutzer sofort, wo sich auf der Zeitachse die einzelnen Annotationen befinden. Sehr oft werden von den Benutzern diese Markierungen dazu genutzt, um längere Videos (Viddler gibt kein Limit für die Länge eines Videos vor) in einzelne Kapitel zu unterteilen. Der Inhalt der Annotationen ist bei Viddler nicht nur auf einen einfachen Text beschränkt, sondern kann auch ganze Diskussionsstränge beinhalten. Auch verschiedene Medienelemente können in einer Annotation gespeichert werden: dazu zählen derzeit Hyperlinks, Bilder und Videos. Abbildung 2.10(a) zeigt eine Annotation mit einer Diskussion darin und in Abbildung 2.10(b) ist eine Annotation mit einem Videoanhang zu sehen.

BubblePLY

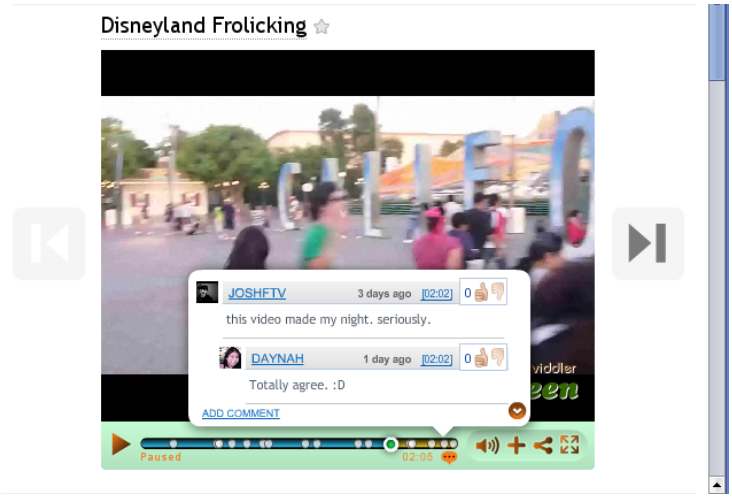
BubblePLY[12] ist ein kommerzielles Produkt der Firma PLYMedia. BubblePLY erlaubt es seinen Benutzern Bereiche der Zeitachse zu markieren und zu annotieren, allerdings können auch hier keine Bildbereiche markiert werden. Statt dessen kann mit einer Sprechblase, wie bei YouTube, auf einen Punkt im Bild hingewiesen werden. Als Zusätzliches Feature kann bei BubblePLY einfach nur ein Text oder ein Bild über das Video gelegt werden. BubblePLY ist das einzige hier vorgestellte Tool, das in der Lage ist, Bereiche auf der Zeitachse zu annotieren und diese Bereiche im Playhead auch anzuzeigen. Abbildung 2.11 zeigt einen Screenshot von BubblePLY mit einer Annotation.

2.2.4 Direkter Vergleich der einzelnen Tools

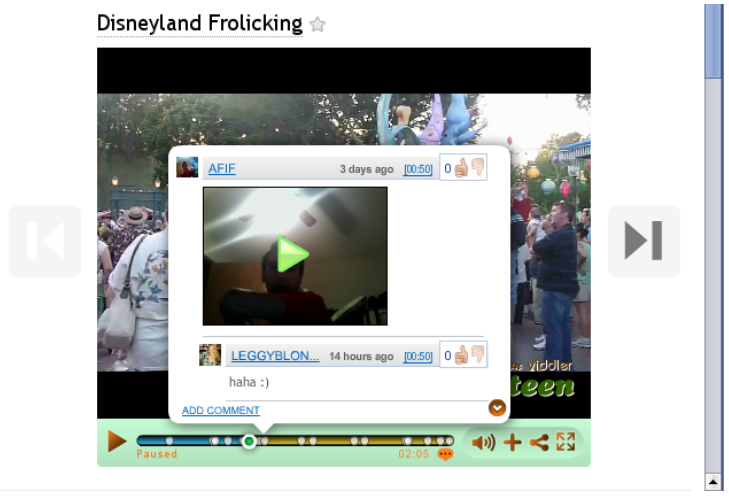
In den Tabellen 2.1 bis 2.3 werden die einzelnen, in diesem Kapitel vorgestellten, Tools miteinander verglichen. Die wichtigsten Annotations-Features werden übersichtlich in einer Matrix zusammengefasst, um so eine besseren Vergleich zu erhalten.

Legende zu den Tabellen 2.1 bis 2.3
⊕ Feature wird voll unterstützt
⊘ Feature wird teilweise unterstützt
⊖ Feature wird nicht unterstützt

[12] http://www.plymedia.com/products/bubbleply/bubbleply.asp



(a) Eine Annotation mit einer Diskussion



(b) Eine Annotation mit einem Video

Abbildung 2.10: *Annotationen bei Viddler. Bildbereiche können zwar bei dieser Plattform nicht markiert werden, statt dessen erlaubt Viddler die Markierung der Zeitachse. Annotationen können neben Text auch Diskussionen, Hyperlinks, Bilder und sogar Videos beinhalten.*

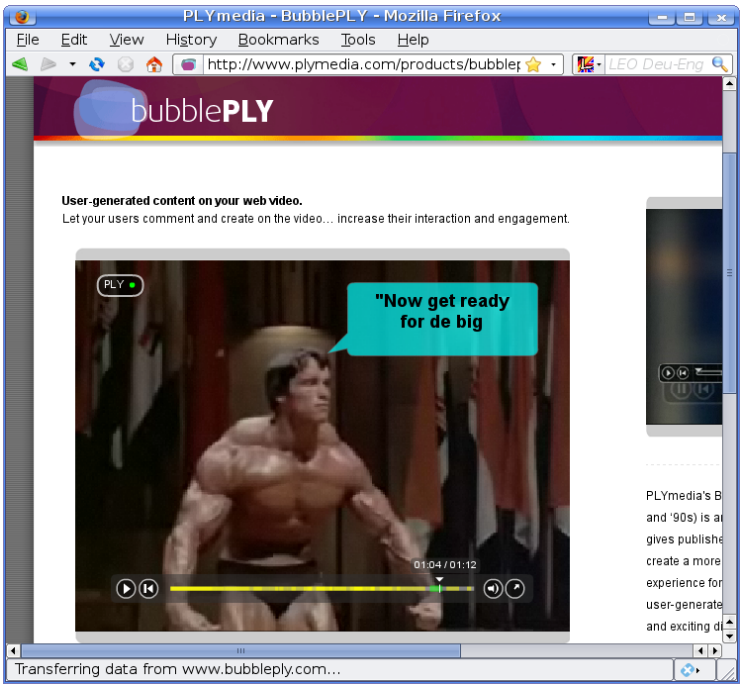


Abbildung 2.11: *Annotationen bei BubblePLY. Dieses Tool ist als einziges in der Lage einen Bereich auf der Zeitachse zu annotieren und diesen Bereich im Playhead anzuzeigen. BubblePLY ist jedoch nicht in der Lage einen Bereich im Bild des Videos zu markieren. Der Text einer Annotation findet sich nur in der Sprechblase wieder, was die Verwendung von längeren Texten nahezu unmöglich macht.*

Web Annotationstools

	Annotea	WebAnn	Sticki-Notes	Wikalong
Suche	⊕	⊕	⊘	⊕
Zugriffskontrolle	⊕	⊕	⊖	⊖
Diskusionen in Annotationen	⊘	⊕	⊕	⊕

Tabelle 2.1: *Vergleich der Website Annotationstools*

Bild Annotationstools

	Flickr	Fotonotes	TelPlus
Suche	⊖	⊖	⊕
Zugriffskontrolle	⊕	⊖	⊖
Diskusionen in Annotationen	⊖	⊖	⊖
Farbwahl einer Annotationen	⊖	⊖	⊕
Bildbereich markieren	⊕	⊕	⊕

Tabelle 2.2: *Vergleich der Bild Annotationstools*

Video Annotationstools

	YouTube	Viddler	BubblePLY
Suche	⊖	⊖	⊖
Zugriffskontrolle	⊘	⊖	⊖
Diskusionen in Annotationen	⊖	⊕	⊖
Farbwahl einer Annotationen	⊖	⊖	⊕
Bildbereich markieren	⊘	⊖	⊘
Zeitbereich markieren	⊘	⊘	⊕

Tabelle 2.3: *Vergleich der Video Annotationstools*

2.3 Fedora Repository

Fedora ist ein Repository (Behälter, Speicher) für beliebige digitale Datenobjekte und wurde primär als Backend für Digital Libraries entworfen. Jedes darin enthaltene Datenobjekt wird mit seiner eindeutigen PID (Persistent Identifier) angesprochen. Ein digitales Objekt besteht dabei aus:

- Metadaten im Dublin-Core Format, die das Objekt selbst beschreiben. Dazu zählen unter anderem ein Titel, eine Kurzbeschreibung des Objekts und das Datum der Erstellung sowie der Name des Autors.
- Einen oder mehreren Datenströme (Datastreams). Diese stellen den oder die eigentlichen Dateninhalte dar.
- Semantische Relationen (RELS-EXT) zu anderen Objekten im Repository.
- Versionierungsdaten (AUDIT), um Änderungen der Daten zurückverfolgen zu können.
- Zugriffsdaten (XACML), um das Objekt und seine Daten zu schützen, bzw. nur bestimmten Personen oder Personengruppen Lese- und/oder Schreibrechte zu erteilen.

Abbildung 2.12 gibt einen groben Überblick über das Fedora-Repository. Fedora speichert und verwaltet digitale Objekte, die Clienten greifen nur über wohldefinierte APIs[13] auf diese Objekte zu. Die digitalen Objekte werden über einen eindeutigen Identifier (PID) angesprochen und können Daten beinhalten. Diese Daten müssen aber nicht unbedingt innerhalb des Objekts gespeichert sein: Fedora ist in der Lage, die Daten von externen Quellen zu beziehen oder durch ein Web-Service berechnen zu lassen. Digitale Objekte können auch in einer Relation zueinander stehen. Ein Objekt kann als ein Teil von einem anderen Objekt betrachtet werden (isMemberOf), oder ein Objekt stellt eine Annotation (isAnnotationOf) für ein anderes dar. Abschnitt 2.3.2 stellt den Aufbau und Möglichkeiten der digitalen Objekte genauer vor, Abschnitt 2.3.3 beschäftigt sich mit den Möglichkeiten der Relationen zwischen den Objekten.

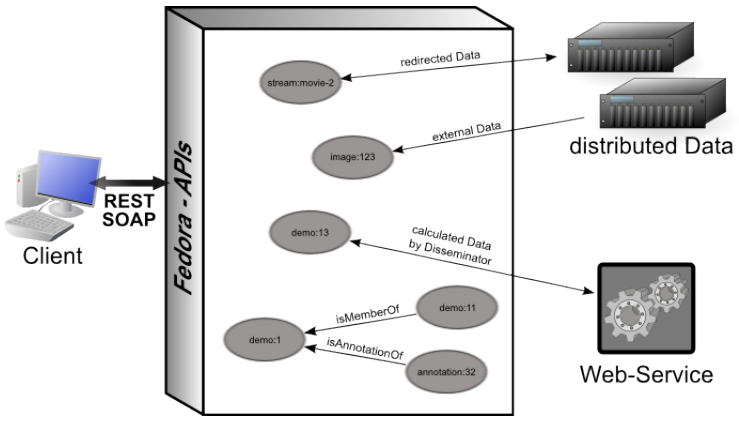


Abbildung 2.12: *Das Fedora Repository verwaltet Objekte, wobei die einzelnen Objekte untereinander in Beziehung stehen können. Die Daten der Objekte müssen nicht unbedingt in Fedora gespeichert sein: sie können auf einem anderen Rechner liegen oder sie können durch ein Web-Service berechnet worden sein. Für den Clienten spielt dies keine Rolle, denn Fedora bietet für alle Objekte im Repository eine einheitliche REST- und SOAP-Schnittstelle an.*

Sämtliche Zugriffe auf das Repository und seinen Funktionen werden über Web-Services realisiert. Dadurch ist eine grundlegende, technische Interoperabilität mit anderen System gewährleistet. Bei der Implementierung eines Clienten, der das Fedora-Repository nutzt, entstehen so keine Abhängigkeiten zu den Programm-Bibliotheken von Fedora. Selbst Änderungen innerhalb des Systems bleiben vor dem Client verborgen. Clienten können primär über zwei unterschiedliche Web-Services mit Fedora kommunizieren:

- Die API-A (Access) bietet einfachen Zugriff auf die Datenobjekte und

[13] Application Programming Interface, englisch für Programmierschnittstelle in der Informatik

deren Datenströme. Diese Schnittstelle ist mittels SOAP[14] bzw. REST[15] Technologie realisiert.

- Die API-M (Management) stellt alle Funktionen für das Anlegen, Warten, Löschen und Ändern der Objekte zur Verfügung. Diese Schnittstelle ist aufgrund ihrer höheren Komplexität als SOAP Interface implementiert. Nur einzelne Funktionalitäten sind auch als REST Interface ausgeführt.

Abbildung 2.13 zeigt die einzelnen Module des Fedora-Repository. Clienten greifen ausschließlich über die wohldefinierten Web-Services auf die einzelnen Funktionalitäten des Repositories zu. Die beiden wichtigsten Schnittstellen sind dabei die *API-A (Access)*, um die Daten auszulesen und die *API-M (Manage)*, um Daten anzulegen, zu ändern oder zu löschen. *Basic Search* bietet den Clienten die Möglichkeit nach Objekten anhand ihres Titels zu suchen, *RDF Search* sucht nach Relationen zwischen den gespeicherten Objekten im Tripelstore (vgl. Abschnitt 2.3.3). Der OAI-Provider bietet eine standardisierte Schnittstelle (OAI-PMH 2.0) um Metadaten zwischen den verschiedenen Anbietern von Repositories bzw. Digital Libraries auszutauschen. Der OAI Standard wird von zahlreichen Firmen und Institutionen getragen und ständig weiterentwickelt, um eine möglichst gute Interoperabilität zwischen den verschiedenen Herstellern zu gewährleisten. [9]

2.3.1 Namespaces

Fedoras digitale Objekte werden in Namespaces aufgeteilt. Dies soll es den Benutzern erleichtern, dass Repository zu ordnen. Es können zum Beispiel Bilder einem eigenen Namespace zugeordnet sein. Dokumente und Texte teilen sich ebenfalls einem eigenen Namespace und sämtliche Disseminatoren (Siehe Abschnitt 2.3.4) sind ebenso in einen eigenen Namespace zusammengefasst. So kann eine logische Strukturierung der digitalen Objekte vorgenommen werden. Diese Strukturierung macht sich bei einem Volumen von mehren tausend Bildern und Texten bezahlt.

Auch das Fedora-Annotation-Service sammelt seine Annotationen in einem eigenen Namespace. Dieser Namespace muss `annotation` heißen. Da dieser

[14] SOAP – Simple Object Access Protocol – dient dazu Daten auszutauschen bzw. entfernte Services aufzurufen. Die SOAP-Technologie stützt sich dabei im wesentlichen auf HTTP als Netzwerkprotokoll und auf XML für den Datenaustausch. [49] [18]

[15] REST – Representational State Transfer – dient, ebenso wie SOAP, zum Datenaustausch bzw. zum Aufrufen von entfernten Services. Im Gegensatz zu SOAP ist REST nicht genau spezifiziert, sondern stellt einen Softwarearchitekturstil für verteilte Systeme dar, der darauf beruht, dass jede Ressource unter einer eindeutigen URI ansprechbar ist. REST nutzt, wie SOAP, das HTTP-Netzwerkprotokoll zum Transport der Daten. [5] [40] [47]

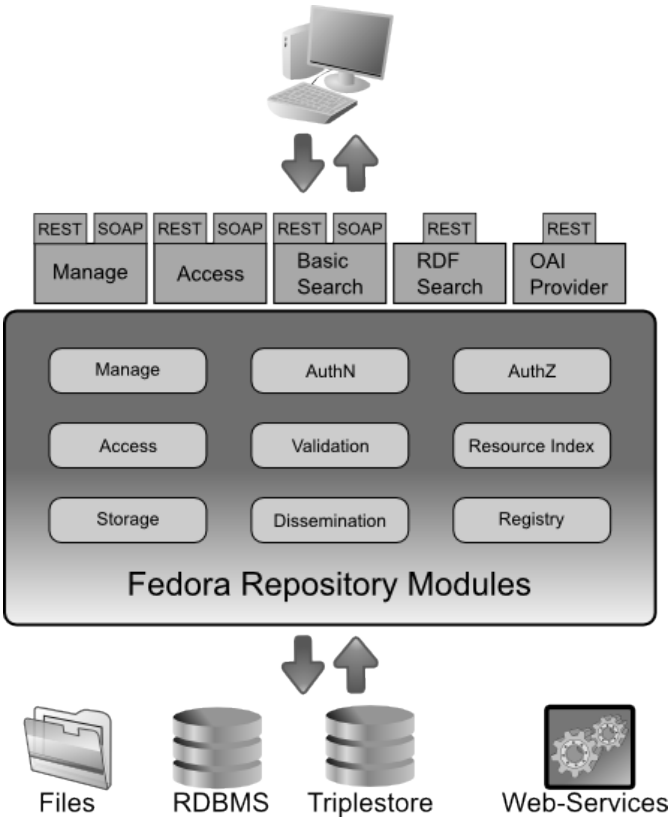


Abbildung 2.13: *Die Fedora Web-Services: der Zugriff auf das Repository erfolgt primär über zwei Schnittstellen. API-A (Access), um auf die Daten zuzugreifen. API-M (Management), um die Daten zu verwalten.*

nach der Installation von Fedora noch nicht zur Verfügung steht, muss er händisch nachgetragen werden. Die Konfiguration der Namespaces wird im Anhang A.1.4 beschrieben.

2.3.2 Fedora Digital Object Model

Fedora definiert einen einzigen Datentyp: das *Fedora Digital Object*. Dieser Datentyp ist so allgemein gehalten, dass er prinzipiell jeden beliebigen Medientyp aufnehmen kann. Dazu zählen zum Beispiel Bilder, Videos, Texte im PDF Format oder eBooks. Selbst ein Container, der wiederum verschiedene Mediendaten in sich vereint, lässt sich mit dem Digital Object Model darstellen. Ein Objekt wird in Fedora immer mit seiner PID, bestehend aus einem Namespace plus eindeutigem Identifier, angesprochen.

Die Abbildung 2.14 zeigt den schematischen Aufbau des Digital Object Mo-

dels. Zu sehen ist, dass ein Objekt in Fedora aus einem eindeutigen Identifier, aus reservierten Datenströmen, aus mehreren Datenströmen, welche die eigentlichen Daten beinhalten und aus Disseminatoren besteht bzw. bestehen kann.

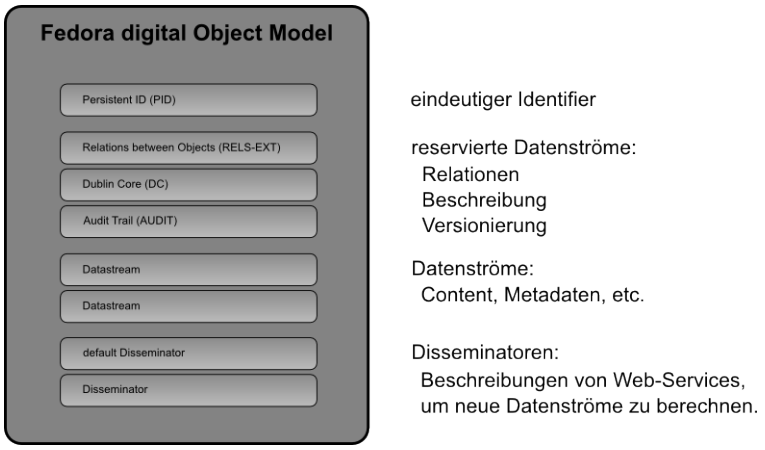


Abbildung 2.14: *Aufbau des Fedora Digital Object Model: ein Objekt besteht aus einer eindeutigen `PID`, aus reservierten Datenströmen (`DC`, `RELS-EXT`, `AUDIT`), aus beliebig vielen weiteren Datenströmen, welche die eigentlichen Daten beinhalten und aus Disseminatoren, die berechnete Daten erzeugen. Der eindeutige `PID` und der Dublin Core Datenstrom `DC` sind immer vorhanden.*

Persistent Identifier – PID

Jedes Objekt in Fedora hat einen eindeutigen Schlüssel. Ein Objekt wird stets mit diesem Schlüssel angesprochen. Dieser Schlüssel wird in Fedora *PID* – **P**ersistent **Id**entifier – genannt. Ein PID besteht dabei aus

1. dem Namespace, in dem das Objekt liegt und
2. einem eindeutigen, alphanumerischen Wert innerhalb des Namespaces.

Die beiden Teile sind durch einen Doppelpunkt (`:`) voneinander getrennt. Beide Teile, Namespace und Object-Identifier, sind case-senstive. Groß- und Kleinschreibung ist also zu beachten. Die Länge eines PIDs ist mit 64 Zeichen beschränkt.

Die formale Definition im BNF-Stil sieht wie folgt aus:

```
object-pid   = namespace-id ":" object-id
namespace-id = 1*( ALPHA | DIGIT | "-" | "." )
object-id    = 1*( ALPHA | DIGIT | "-" | "." | "~" | "_" | hex )
hex          = "%" HEXDIG HEXDIG
```

Beispiele: content:1
demo:MyFedoraDigitalObject
content:Image-1
image:3
doc:201

Datenströme

Ein Objekt ist nicht nur auf einen Medientypen, wie zum Beispiel ein Bild, beschränkt. Vielmehr kann jedes Objekt beliebig viele verschiedene Mediendaten besitzen. Dabei müssen die einzelnen Daten nicht vom selben Medien-Typen (MIME-TYPE) sein. Die einzelnen Mediendaten eines Objektes werden *Datastream* (zu deutsch: *Datenstrom*) genannt. Auf diese Weise ist es zum Beispiel sehr einfach möglich, dass ein Objekt das selbe Bild öfter abgespeichert hat – jedoch in unterschiedlichen Auflösungen. Einmal wird das Bild als kleine Vorschau (Thumbnail) gespeichert. Einmal in mittlerer Auflösung und einmal in voller Auflösung. Der Client, der auf das Repository zugreift, kann seinerseits diese unterschiedlichen Auflösungen von ein und dem selben Bild nutzen, indem er immer nur jenes Bild, bzw. Datenstrom, vom Server lädt, dass er für seine tatsächliche Aufgabe benötigt. Wird nur eine Vorschau in einer Liste angezeigt, so wird wahrscheinlich das Thumbnail reichen. Nur wenn der Benutzer genauere Details des Bildes betrachten möchte, muss der Client die volle Auflösung des Bildes laden. Dadurch ist es möglich Bandbreite und Ladezeiten zu sparen bzw. zu optimieren. In Abbildung 2.15 wird dieses Szenario dargestellt. Der linke Screenshot zeigt ein Haus in seiner vollen Auflösung, der rechte Screenshot zeigt das dazugehörige Thumbnail. Beide Datenströme sind in einem einzigen digitalen Objekt gespeichert.

Jeder Datenstrom eines Objekts kann auf eine der folgenden vier unterschiedlichen Arten gespeichert und in Fedora dementsprechend behandelt werden:

- Internal XML (X)
  Fedora speichert die XML-Daten des Datenstroms innerhalb des Objekts. Optional können diese XML-Daten einem eigenen XML-Namespace angehören. Dies wird zwar von Fedora nicht überprüft, hat aber den Vorteil, dass die verwendeten XML-Tags nicht mit Fedoras eigenen Tags kollidieren.

- Managed (M)
  Fedora speichert einen beliebigen Datenstrom innerhalb des Objekts

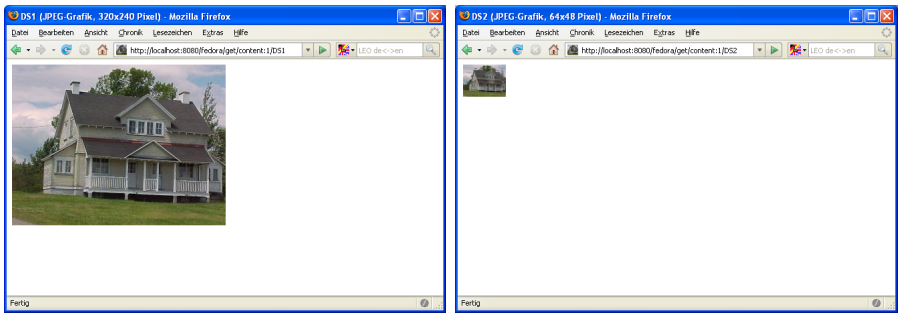


Abbildung 2.15: *Zwei Datenströme eines digitalen Objekts. Links der Datenstrom DS1: er enthält ein Bild von einem Haus in voller Auflösung (320 x 240 Pixel). Rechts der Datenstrom DS2: er enthält das gleiche Bild von dem Haus. Aber in einer weit geringeren Thumbnail-Auflösung (64 x 48 Pixel). Ein Client, der das Bild von dem Haus anzeigen muss, kann einfach denjenigen Datenstrom nutzen, der für seine Aufgabe geeigneter ist: DS2 für eine Vorschau; DS1 für eine Detailanzeige.*

als Bytestrom (BASE-64[16] codiert) ab.

- External (E)
  Der Datenstrom wird nicht von Fedora gewartet. Die Daten selbst liegen auf einem anderen Rechner (Server). Bei jedem Zugriff auf diesen Datenstrom lädt Fedora die Daten von dem Rechner auf dem die Daten liegen und leitet sie anschließend an den Clienten weiter.

- Redirect (R)
  Fedora sendet den Aufrufer des Datenstroms ein HTTP-Redirect zu dem Rechner, auf dem die Daten liegen. Auch bei dieser Variante wird der Datenstrom nicht von Fedora gewartet. Redirect ist u.a. bei Streaming oder bei einem Verweis auf eine Homepage, in der mit relativen Links gearbeitet wurde, der Variante External vorzuziehen.

Die letzten beiden Varianten (E und R) bieten sich an, wenn zwar Objekte und Relationen zwischen einzelnen Objekten verwaltet werden sollen, die Daten sich allerdings auf einem anderen Rechner befinden (müssen).

Abbildung 2.16 zeigt ein digitales Objekt mit den unterschiedlichen Typen von Datenströmen. Der Datenstrom `DC` (Dublin-Core) wird intern als XML gespeichert (X), `DS0` ist ein Bild, das ebenfalls intern gespeichert ist (M) – Fedora kann diese Daten direkt an den Clienten ausliefern. Der Datenstrom `DS1` besteht aus einem externen Bild (E) – Fedora lädt dieses Bild von dem

[16] Base64 ist ein Verfahren zur Kodierung von Binärdaten, bei dem jeweils 3 Byte (= 24 Bit) in vier 6 Bit Blöcke aufgeteilt werden. Jeder Block bildet eine Zahl zwischen 0 und 63, die durch ein druckbares ASCII Zeichen (die Buchstaben a-z, A-Z und die Zahlen 0-9) dargestellt wird. Dadurch ist es möglich, beliebige Binärdaten als Text zu kodieren. [44]

externen Rechner und gibt dann diese beiden Daten an den Clienten weiter. `DS2` ist ein Redirect (R) auf eine Homepage – Fedora leitet daher den Clienten mit einem HTTP-Redirect an diese Homepage weiter. Die Pfeile in der Abbildung stellen die jeweiligen Datenflüsse zwischen dem Clienten, Fedora und den externen Ressourcen dar.

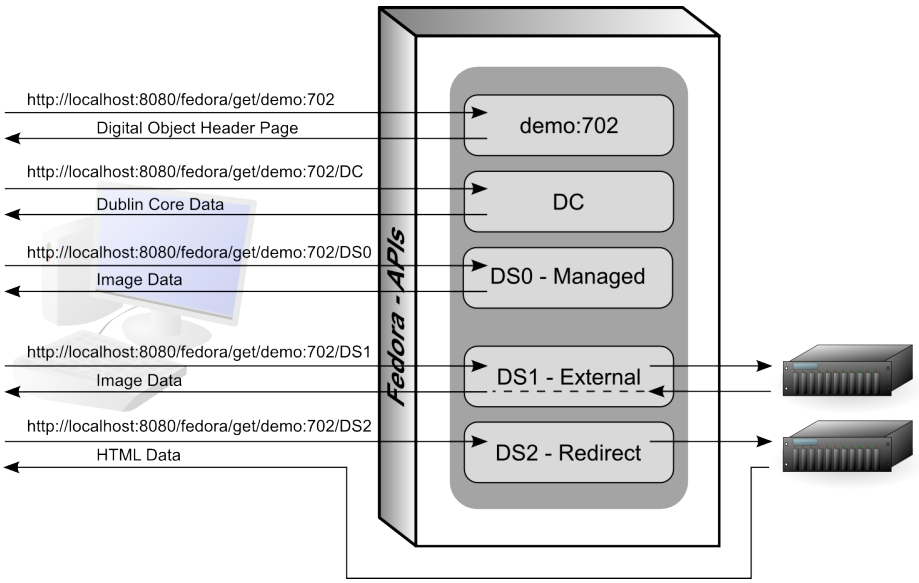


Abbildung 2.16: *Ein digitales Objekt und seine verschiedenen Typen von Daten (X, M, E, R). Die Pfeile zeigen den jeweiligen Datenfluss zwischen Client, Fedora und den externen Daten an.*

Dublin-Core Metadaten: DC Datenstrom

Zusätzlich zu den eigentlichen Daten beinhaltet das Digital Object Model noch weitere Informationen. Diese sollen über den Inhalt, den Autor usw. Aufschluss geben. Diese Metadaten werden im weit verbreiteden *Dublin-Core* Format [8][7] innerhalb des Objektes gespeichert. Intern sind auch diese Daten als Datenstrom abgelegt. Der dazugehörige Datenstrom muss den Namen *DC* tragen. Mit dieser Herangehensweise bleibt die Definition des Digital Object Models einfach und konsistent. Zum Unterschied zu den eigentlichen Daten, wie sie Abschnitt 2.3.2 beschrieben worden sind, ist dieser Datenstrom immer vorhanden. Selbst wenn er nicht explizit angelegt wurde, werden diese Daten von Fedora gepflegt und nötigenfalls erzeugt. In diesem Fall wird ein Dublin Core Datenstrom mit den Datenfeldern `dc:title` für den Titel und `dc:identifier` mit der PID des neuen Objekts angelegt.

Versionierung: AUDIT Datenstrom

Natürlich ändern sich Daten im Laufe der Zeit. Fedora trägt auch diesem Umstand Rechnung. Ein Objekt kennt alle Änderungen die an ihm vorgenommen wurden. Fedora führt Buch über sämtliche Datenänderungen eines Objektes und speichert diese Versionsdaten intern als einen eigenen Datenstrom. Dieser spezielle Datenstrom trägt den Namen *AUDIT*. Der Benutzer des Repositories hat keinen direkten Einfluss auf diese Daten – sie werden ausschließlich von Fedora verwaltet. Diese Daten dienen nicht nur dem Aufzeichnen von Änderungen an den Daten des Objekts. Der Administrator kann mit der Hilfe dieser AUDIT-Daten jederzeit eine frühere Versionen des Objektes wiederherstellen. Der `fedora-admin` Client bietet hierfür pro Datenstrom einen eigenen Schieberegler an. Mit dem ist es möglich zu jeder vorangegangenen Version zurückzukehren. Die Clienten können jede Version eines Objekts abrufen, indem sie den Zeitstempel der gewünschten Version mit angeben.

Die API-A bietet eine Schnittstelle an, um die Versions-Geschichte eines Objekts abzufragen. Das Ergebnis eines REST-Aufrufs

`http://hostname:port/fedora/getObjectHistory/`*`pid`*`[?xml=BOOLEAN]`

ist eine Liste von allen Änderungen des Objekts mit der angegeben *pid*. Wahlweise kann die Liste als HTML- oder XML-Dokument an den Aufrufer gesendet werden.

Beispiel:

`http://localhost:8080/fedora/getObjectHistory/content:1?xml=true`

liefert als Ergebnis ein XML-Dokument, dass für für jede Änderung des Objekts den jeweiligen Zeitstempel angibt

```
<fedoraObjectHistory pid="content:1" xmlns:xsd="http://www.w3.org... >
        <objectChangeDate>2008-04-14T14:07:43.843Z</objectChangeDate>
        <objectChangeDate>2008-04-14T14:09:13.593Z</objectChangeDate>
        <objectChangeDate>2008-04-14T14:09:55.265Z</objectChangeDate>
        <objectChangeDate>2008-04-14T14:10:43.812Z</objectChangeDate>
</fedoraObjectHistory>
```

Um die Daten einer bestimmte Version eines Objekts abzurufen, muss der Client lediglich den entsprechenden Zeitstempel mit angeben.

`http://localhost:8080/fedora/get/content:1/DC`

liefert die gerade aktuelle Version (14. April 2008, 14:10:43) des Dublin-Core Datenstroms vom Objekt mit der PID `content:1`.

`http://localhost:8080/fedora/get/content:1/DC/2008-04-14T14:09:55.265Z`

hingegen liefert den Dublin-Core Datenstrom in der Version vom 14. April 2008, 14:09:55.

Weitere Datenströme

Nicht immer müssen alle möglichen Daten eines Objekts in den eigenen Datenströmen abgespeichert werden. In vielen Fällen ist es durchaus möglich, ja sogar einfacher, einen Datenstrom mithilfe eines bereits existierenden Datenstroms zu berechnen. Zum leichteren Verständnis für diese Idee hier ein Beispiel:

Ein Objekt beinhaltet ein Bild. Im Gegensatz zu dem Beispiel aus Abbildung 2.15 wird dieses Bild allerdings nur einmal im Objekt gespeichert und zwar in seiner vollen Auflösung. Die anderen Auflösungen des Bildes werden diesmal bei jedem Aufruf dynamisch berechnet. Das abgespeicherte, hochauflösende Bild wird dabei einem Web-Service übergeben, der das Bild in die gewünschte Größe skaliert. Aus Sicht des Clienten, der das kleinere Bild angefordert hat, sieht es so aus als ob das Bild sehr wohl im Repository gespeichert gewesen wäre. Dies ist jedoch nicht der Fall – es wurde nur mithilfe des hochauflösenden Bildes und einem Web-Service berechnet.

Damit Fedora dies bewerkstelligen kann, sind folgende Schritte nötig: zum einen müssen spezielle Objekte im Repository angelegt werden, die Schnittstelle eines Web-Services, welche von Fedora benutzt werden soll, beschreiben. Diese speziellen Objekte sind selbst wiederum digitale Objekte, werden *Disseminatoren* genannt und sind intern etwas anders aufgebaut. Zum anderen müssen den digitalen Objekte die nötigen Disseminatoren zugewiesen werden. Abschnitt 2.3.4 beschäftigt sich näher mit den Disseminatoren und den Möglichkeiten die sie bieten.

Ein anderer Datenstrom bildet Relationen zwischen einzelnen Objekten ab. Dieser Datenstrom trägt den Namen *RELS-EXT* und beinhaltet XML-Daten. Diese Daten sind die XML-Serialisierung einer Ontologie im RDF-Format. Fedora selbst stellt bereits ein einfaches Vokabular in RDFS [35] zur Verfügung. Abschnitt 2.3.3 wird sich noch genauer mit den Relationen, ihrer Speicherung und den daraus resultierenden Möglichkeiten beschäftigen.

2.3.3 Beziehungen zwischen Objekten: Ressource Index

Digitale Objekte können in Fedora in unterschiedlicher Weise miteinander in Beziehung stehen. Ein Objekt kann als Container dienen, wobei andere Objekte Teil dieses Containers sind. Oder ein Objekt kann ein anderes Objekt annotieren, d.h. es bietet für dieses Objekt eine Beschreibung an.

Mit semantischen Relationen ist es möglich eine Ordnungssystem in die Datenbank zu bringen, denn ein einzelnes Objekt steht nur in den wenigsten

Fällen für sich alleine. Viel mehr steht jedes Objekt mit einem oder mehreren anderen Objekten in einer bestimmten Beziehung. Diese Beziehungen werden mit dem *Ressource Index* innerhalb des Fedora Repositories abgebildet. Das System bietet eine einfache Schnittstelle an, um nach Objekten und ihren Relationen zu suchen. Der Abschnitt 'Abfragen des Ressource Indexes' in diesem Kapitel beschreibt diese Schnittstelle genauer.

Fedora bietet seinen Benutzern eine vordefinierte Ontologie an, die einige, wenn auch sicherlich nicht alle möglichen, Relationsarten für Objekte definiert. Dem Benutzer steht es frei, diese Grund-Ontologie [35] zu verwenden um Objekte miteinander in Beziehung zu stellen. Der Benutzer kann dieses Schema nach seinen eigenen Wünschen erweitern oder ein gänzlich anderes Ontologieschema nach seinen Bedürfnissen erstellen.

Auch das *Fedora-Annotation-Service* verwendet Relationen, um eine Annotation auszudrücken. Eine komplette Annotation besteht dabei immer aus zwei Objekten:

- Dem Objekt, das annotiert werden soll.
- Einem Objekt, welches die Annotation selbst darstellt. Dieses *Annotations-Objekt* zeigt dabei auf das zu annotierende Objekt, d.h. es steht in einer Relation zu diesem Objekt. Diese Relation, bzw. der Typ dieser Relation, muss daher immer wohldefiniert sein. Die Bezeichnung kommt aus dem Grund-Ontologieschema von Fedora und lautet `isAnnotationOf`.

Durch diese Herangehensweise wird das zu annotierende Medienobjekt selbst nicht direkt von einer Annotation verändert. Ein und das selbe Medienobjekt kann so auch beliebig viele Annotationen haben, ohne dass es erweitert oder in irgend einer Art speziell behandelt werden muss. Des weiteren werden sämtliche Clienten, die mit diesem Medienobjekt arbeiten, durch die Annotationen nicht beeinflusst. Solange sie nicht an den Annotationen interessiert sind, brauchen sie sich auch nicht weiter um diese zu kümmern.

Der Datenstrom RELS-EXT

Jedes digitale Objekt in Fedora kann (maximal) einen Datenstrom enthalten, der die Relationen zu anderen Objekten darstellt. Dieser Datenstrom muss den Namen *RELS-EXT* tragen und beinhaltet XML-Daten. Diese Daten sind die Beschreibungen aller Relationen von diesem Objekt aus betrachtet und wird in RDF/XML ausgedrückt. Dabei kann der Benutzer das RDF-Schema der Grund-Ontologie [35] von Fedora nutzen. Er kann dieses selbst erweitern oder aber ein komplett neues Ontologieschema definieren.

Abbildung 2.17 zeigt den RELS-EXT Datenstrom eines digitalen Objekts. Das Fedora-Annotation-Service legt seine Annotations-Objekte nach genau diesem Muster an. Dieser RELS-EXT Datenstrom erzeugt eine Relation vom Typ `isAnnotationOf` zwischen dem Annotations-Objekt und dem zu annotierenden Objekt.

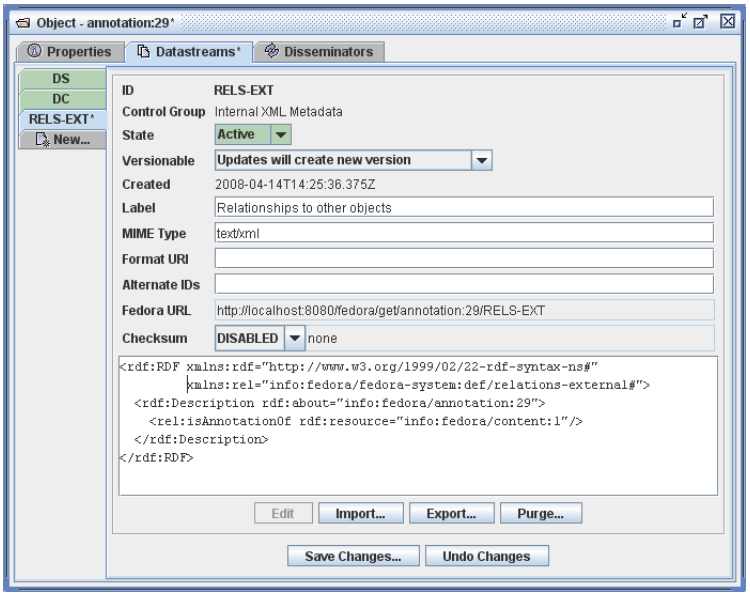


Abbildung 2.17: *Der RELS-EXT Datenstrom eines digitalen Objekts. Er stellt eine Relation vom Typ `isAnnotationOf` zwischen diesem Annotations-Objekt mit der PID `annotation:29` und dem zu annotierendem Objekt mit der PID `content:1` her. Als Format für den RELS-EXT Datenstrom kommt in Fedora RDF/XML zum Einsatz. Der Screenshot ist dem `fedora-admin` Clienten entnommen.*

Daten im Ressource Index

Welche Daten werden nun im Triplestore von Fedora abgelegt? Zum einen all jene Relationen, die in den RELS-EXT Datenstrom der einzelnen digitalen Objekte gespeichert wurden. Zum anderen werden von Fedora automatisch noch weitere Tripel gespeichert, die aus den Daten der Objekte erzeugt werden. Dazu zählen administrative Daten über das Objekt, wie zum Beispiel das Datum, an dem das Objekt erzeugt wurde, oder die Daten aus der Dublin Core Beschreibung. Aber auch Metadaten zu den einzelnen Datenströmen werden im Triplestore abgelegt.

Die Tabellen 2.4 bis 2.7 geben Aufschluss darüber, welche Daten von Fedora im Triplestore abspeichert werden und wie die Subjekte, die Prädikate und die Objekte eines RDF-Tripel danach aussehen. Dieses Wissen ist

notwendig, wenn diese Daten abgefragt bzw. gesucht werden sollen. Aus Gründen der Übersichtlichkeit sind nur die wichtigsten Tripel aufgelistet und auf die Angaben der Kardinalitäten wurde gänzlich verzichtet. Eine komplette Übersicht über alle möglichen Tripel und deren Kardinalitäten liefert [36].

Base Triples

Subjekt	Prädikat	Objekt
info:fedora/$PID	rdf:type	fedora-model:FedoraObject or BMechObject or BDefObject
info:fedora/$PID	fedora-model:createdDate	(date created in UTC)
info:fedora/$PID	fedora-view:lastModifiedDate	(date modified in UTC)
info:fedora/$PID	fedora-model:state	fedora-model:Active or Inactive or Deleted
info:fedora/$PID	fedora-model:owner	(not used)
info:fedora/$PID	fedora-model:label	(any string)
info:fedora/$PID	fedora-model:cModel	(any string)

Tabelle 2.4: *Resource Index - Base Triples*

Dublin Core Triples

Subjekt	Prädikat	Objekt
info:fedora/$PID	dc:title	(any string)
info:fedora/$PID	dc:identifier	(any string)
info:fedora/$PID	(any other dc predicate)	(any string)

Tabelle 2.5: *Resource Index - Dublin Core Triples*

RELS-EXT Triples

Subjekt	Prädikat	Objekt
info:fedora/$PID	(any non-reserved predicate)	(any uri or literal)

Tabelle 2.6: *Resource Index - RELS-EXT Triples*

Datastream Triples

Subjekt	Prädikat	Objekt
info:fedora/$PID	fedora-model:hasDatastream	info:fedora/$PID/$DSID
info:fedora/$PID	fedora-view:disseminates	info:fedora/$PID/$DSID
info:fedora/$PID/$DSID	fedora-view:disseminationType	info:fedora/*/$DSID
info:fedora/$PID/$DSID	fedora-view:mimeType	(any mime type string)
info:fedora/$PID/$DSID	fedora-view:lastModifiedDate	(date modified in UTC)
info:fedora/$PID/$DSID	fedora-model:state	fedora-model:Active or Inactive or Deleted
info:fedora/$PID/$DSID	fedora-view:isVolatile	(true if R or E, false if M or X)

Tabelle 2.7: *Resource Index - Datastream Triples*

Abbildung 2.18 zeigt das digitale Objekt mit der PID `demo:700` und seine Einträge im Ressource Index. Als Subjekt kommt hier immer das digitale Objekt `info:fedora/demo:700` zur Anwendung. Die Pfeile sind die einzelnen Prädikate und die dazugehörigen Daten stellen das jeweilige Objekt im Tripelstore dar. Dieses digitale Objekt mit der PID `demo:700` hat zum Beispiel einen Eintrag im Dublin-Core Datenstrom für den Titel. Dieser Eintrag lautet `Demo Object with managed Data`. Daraus folgt, dass es im Ressource Index einen Eintrag in der Form Subjekt – Prädikat – Objekt gibt der lautet:

```
info:fedora/demo:700 - dc:title - Demo Object with managed Data
```

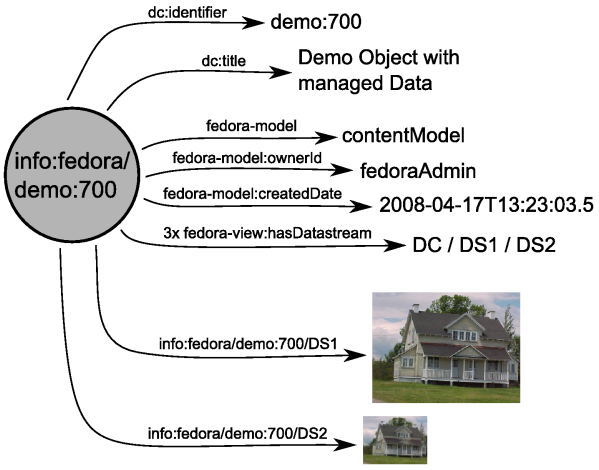


Abbildung 2.18: *Das digitale Objekt* `demo:700` *und seine Daten. Die Pfeile stellen die Prädikate im Triplestore des Ressource Indexes dar, so wie sie von Fedora gespeichert wurden.*

Abfragen des Ressource Indexes

Fedora speichert die Tripel vor allem deshalb ab, damit die Benutzer später wieder danach suchen können. Somit ist es leicht möglich, Objekte nach den verschiedensten Kriterien wiederzufinden. Zum einem kann der Benutzer nach Einträgen im Dublin Core Datenstrom suchen. Zum anderen kann er sich all jene Objekte aus dem Repository filtern, die in einer ganz bestimmten Beziehung zueinander stehen.

Mit Hilfe der Tripel im Ressource Index können also Objekte beliebig selektiert werden. Fedora bietet den Clienten für die Suche im Repository ein REST-Interface an. Möchte der Benutzer eine Abfrage von Hand erstellen, steht das Ressource Index Search Service, auch *RISearch* genannt, unter `http://localhost:8080/fedora/risearch` zur Verfügung.

Das User Interface besteht aus drei Tabs: Find Tuples, Find Triples und Show Aliases. *Find Tuples* sucht nach Einträgen im Ressource Index. Das Ergebnis ist eine Liste aller betroffenen Objekte. Das Ergebnis von *Find Triples* ist eine Liste von RDF-Statements. *Show Aliases* zeigt alle gesetzten Aliases für die Suche. Diese haben den Vorteil, dass der Sourcecode einer Abfrage nicht ausartet und damit unleserlich bzw. unwartbar wird. So ist es beispielsweise möglich, anstatt des korrekten und vollständigen Prädikats
`info:fedora/fedora-system:def-relations-external#isAnnotationOf`
einfach
`fedora-rels-ext:isAnnotationOf`
zu verwenden. Abbildung 2.19 zeigt das User Interface des RISearch-Servic-

es. Der erste Screenshot zeigt das Such-Interface, wobei iTQL als Abfragesprache gewählt wurde. Das Ergebnis soll im SPARQL-Format geliefert werden und alle digitalen Objekte zurückliefern, die das Objekt mit der PID `content:1` annotieren. Im Ergebnis sollen sowohl die PID als auch der Titel aller betroffenen Objekte aufscheinen. Die Ergebnisliste soll anhand der PID der Annotations-Objekte aufsteigend sortiert werden.

Eine Suche im Ressource Index verwendet als Abfragesprache primär iTQL [22]. Diese Sprache ist konzeptionell stark an SQL angelehnt und wird vom Kowari-Triplestore direkt unterstützt. Eine Abfrage besteht, wie in SQL, zumindest aus einem SELECT und einem FROM Block. Ein optionaler WHERE Block kann die Ergebnisliste einschränken. Ein ebenfalls optionaler ORDER BY Block kann sie auf- oder absteigend sortieren. Der Benutzer kann außerdem das Format der Ergebnisliste wählen. Es stehen dabei vier verschiedene Formate zur Auswahl:

- CSV - Comma Separated Values
  Pro Zeile ein Datensatz. Die Spalten werden mit einem Komma (,) voneinander getrennt.
- Simple
  Pro Zeile steht eine Spalte der Ergebnisliste. Die Zeilen der Ergebnisliste sind durch eine Leerzeile gekennzeichnet. Dieses Ausgabeformat ist besonders für Menschen gut leserlich.
- SPARQL
  Ein XML Format, das vom W3C standardisiert wurde [19]. Es ist speziell für Abfragen in Ontologien entwickelt worden.
- TSV - Tab Separated Values
  Pro Zeile ein Datensatz. Die Spalten werden mit einem Tabulator (\t) voneinander getrennt.

Eine Abfrage kann vom Benutzer limitiert werden, d.h. dass nicht alle betroffenen Tupel zurückgegeben werden sollen, sondern nur die ersten 10, 100 oder 1000. Dies eignet sich besonders zum Testen einer neuen Abfrage in einem sehr großen Repository, wo eine Abfrage mehrere zehntausend Tupel zurückliefern und daher eine Menge Ressourcen (u.a. RAM, CPU-Auslastung) und Zeit benötigen würde. Die Limitierung der Ergebnisliste soll eine unnötige Belastung des Rechners verhindern helfen.

Des weiteren stehen drei Flags für eine Abfrage zur Verfügung:

- Force Distinct
  Löscht doppelte Objekte aus dem Ergebnis. Dieses Flag macht nur

bei einer RDQL-Abfrage[17] Sinn, da Kowari's iTQL immer doppelte Einträge löscht.

- Fake Media-Types
  Ein Workaround für manche Browser, damit diese das Ergebnis anzeigen und nicht den 'Speichern unter...' Dialog öffnen.

- Stream Immediately
  Wenn aktiviert, beginnt RISearch sofort mit dem senden der Antwort. Wenn inaktiv, so wartet RISearch, bis alle Daten vorhanden sind. Erst dann wird das gesamte Ergebnis an den Clienten geschickt.

Zur Erläuterung von iTQL, hier ein paar Beipiele:

```
# get all tuples. no matter what they are
select $subject $predicate $object
from   <#ri>
where  $subject $predicate $object


# get all tuples. but subject and object must be a Fedora-Digital-
  Object
select $subject $predicate $object
from   <#ri>
where  $subject $predicate $object
and    $subject <rdf:type> <fedora-model:FedoraObject>
and    $object  <rdf:type> <fedora-model:FedoraObject>


# get all tuples from the subject with PID (= DublinCore:Identifier) '
  content:1'
select $subject $predicate $object
from   <#ri>
where  $subject $predicate $object
and    $subject <dc:identifier> 'content:1'


# get the PIDs and dublin-core titles of all digital-objects
select $obj $title
from   <#ri>
where  $obj <rdf:type> <fedora-model:FedoraObject>
and    $obj <dc:title> $title
```

In Abbildung 2.19 wird das RISearch-Interface verwendet, um alle Annotations-Objekte zu dem digitalen Objekt mit der PID `content:1` zu finden. Zusätzlich werden die Annotations-Objekte nach dem Identifier aufsteigend sortiert. Diese iTQL-Abfrage wird auch vom Fedora Annotations Framework verwendet[18], um alle Annotations-Objekte zu einem bestimmten digitalen Objekt im Repository zu finden:

[17] RDQL ist eine vom W3C-Konsortium standardisierte Abfragesprache für RDF Daten. RDQL ist, wie iTQL, an die Abfragesprache SQL angelehnt. [20].

[18] Das Annotation-Framework kürzt das Ergebnis der iTQL-Abfrage allerdings etwas ab, denn es ist nur an den einzelnen PIDs der Annotations-Objekte interessiert.

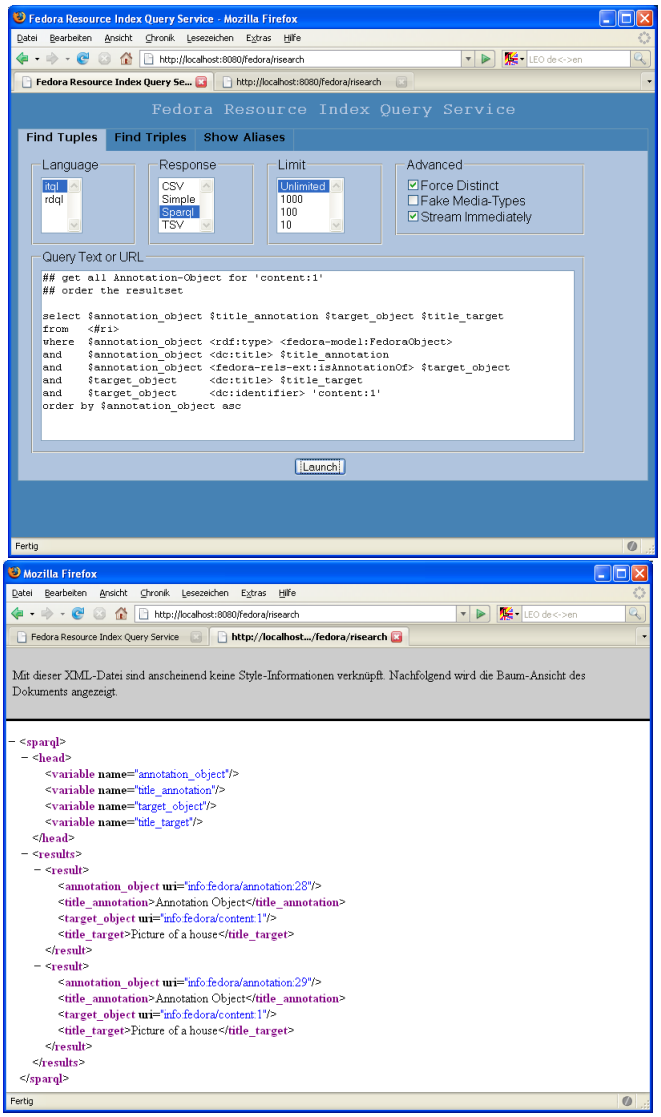


Abbildung 2.19: *Das interaktive RISearch Interface mit einer iTQL-Abfrage. Oben das HTML-Frontend für eine Tupel-Suche. Unten das Ergebnis im SPARQL-Format. Die Abfrage sucht nach allen Objekten, die in einer Relation vom Typ* `isAnnotationOf` *zueinander stehen. Das Zielobjekt hat die PID* `content:1` *und das Subjekt* `$annotation_object` *muss ein digitales Objekt sein. Als Ergebnis der Suche sollen sowohl die PID als auch der Titel der beiden Objekte, die in Beziehung zueinander stehen, ausgegeben werden. Die Ergebnisliste soll zusätzlich aufsteigend nach der PID von* `$annotation_object` *sortiert werden.*

```
select $annotation_object $title_annotation $target_object
  $title_target
from   <#ri>
where  $annotation_object <rdf:type> <fedora-model:FedoraObject>
and    $annotation_object <dc:title> $title_annotation
and    $annotation_object <fedora-rels-ext:isAnnotationOf>
  $target_object
and    $target_object     <dc:title> $title_target
and    $target_object     <dc:identifier> 'content:1'
order by $annotation_object asc
```

Hier das dazugehörige Ergebnis im SPARQL-Format. Gefunden wurden in diesem Beispiel zwei Objekte, die `content:1` annotieren: die Annotations-Objekte `annotation:28` und `annotation:29`.

```
<?xml version="1.0" encoding="UTF-8"?>
<sparql xmlns="http://www.w3.org/2001/sw/DataAccess/rf1/result">
  <head>
    <variable name="annotation_object"/>
    <variable name="title_annotation"/>
    <variable name="target_object"/>
    <variable name="title_target"/>
  </head>
  <results>
    <result>
      <annotation_object uri="info:fedora/annotation:28"/>
      <title_annotation>Annotation Object</title_annotation>
      <target_object uri="info:fedora/content:1"/>
      <title_target>Picture of a house</title_target>
    </result>
    <result>
      <annotation_object uri="info:fedora/annotation:29"/>
      <title_annotation>Annotation Object</title_annotation>
      <target_object uri="info:fedora/content:1"/>
      <title_target>Picture of a house</title_target>
    </result>
  </results>
</sparql>
```

2.3.4 Dynamisch generierte Daten: Disseminatoren

Häufig können aus bereits existierenden Daten bzw. Datenströmen weitere Daten generiert werden. So können zum Beispiel aus einem hochauflösenden Farbbild mehrere Ansichten generiert werden. Zum einen das Bild in geringerer Auflösung. Zum anderen das Bild in schwarzweiß. Disseminatoren stellen in Fedora die Schnittstelle zwischen gespeicherten Objektdaten und dynamisch generierten Daten dar. Die Disseminatoren greifen dabei stets auf REST oder SOAP Web-Services zurück, um aus einem Datenstrom bzw. aus einer Fedora-URL neue Daten zu gewinnen.

Die Disseminatoren selbst sind dabei nichts anderes als Fedora Digital Objects und werden ebenso wie die Datenobjekte im Repository gespeichert.

Zwei Objekte – *bDef* und *bMech* – in Kombination ergeben einen Disseminator:

- *bDef*: Behavior Definition Objects
  Die bDef-Objekte repräsentieren eine abstrakte Schnittstelle für einen Disseminator. In dem Sinne sind sie mit den Interfaces von Java vergleichbar. Sie definieren eine oder mehrere Methoden, die ein digitales Objekt erweitern, wenn es diesen bDef als Disseminator benutzt. Ein und das selbe bDef-Objekt kann von mehreren digitalen Objekten genutzt werden.

- *bMech*: Behavior Mechanism Objects
  Ein bMech-Objekt ist eine konkrete Implementierung eines bDef-Objekts, wobei ein bDef-Objekt durchaus von mehreren verschiedenen bMech-Objekte implementiert werden kann. Die bMech-Objekte speichern alle Daten die nötig sind, um ein Web-Service aufzurufen. Dazu zählen die URL und die Art des Web-Services (REST oder SOAP) und nötigenfalls die dazugehörige WSDL. Des weiteren werden in den bMech-Objekten das Daten-Profil für den Disseminator gespeichert. Dieses Profil definiert die Anzahl und den Typ der Eingabe-Daten für das Web-Service und gegebenenfalls die nötigen Parameter für den Aufruf.

Die bDef-Objekte stellen somit die abstrakte Schnittstelle für die digitalen Objekte dar und erweitern diese. Die bMech-Objekte sind konkrete Implementierungen bzw. Bindungen eines Web-Services an solch eine abstrakte Schnittstelle. Der Disseminator eines digitalen Objekts ist also nichts weiter als eine Verknüpfung von einem bDef und einem bMech mit dem Objekt. Aus Sicht des Clienten sieht es so aus, als ob das Objekt durch die Methoden, die im bDef definiert sind, erweitert wurde.

Abbildung 2.20 zeigt ein digitales Objekt mit drei internen und zwei berechneten Datenströmen. Das Objekt benutzt nur ein einziges bDef-Objekt mit der PID `BDEF:2`. Dieses bDef-Objekt definiert die zwei Methoden: `Zoom` und `Grayscale`. Dadurch sieht es für den Clienten so aus, als ob das Objekt selbst zwei weitere Datenströme besitzt. Im dazugehörigen bMech-Objekt sind die genauen Daten für das benutzte Web-Service hinterlegt. Darunter fallen die URL und die Eingabe-Daten für das Web-Service. In diesem Beispiel wird für jede der zwei bDef-Methoden der gleiche Eingabe-Datenstrom verwendet: das hochauflösende Bild, vom Datenstrom `HIGH`. In der Graphik ist dies mit dem Pfeil zwischen den Eingabe-Datenstrom und dem Disseminator angedeutet.

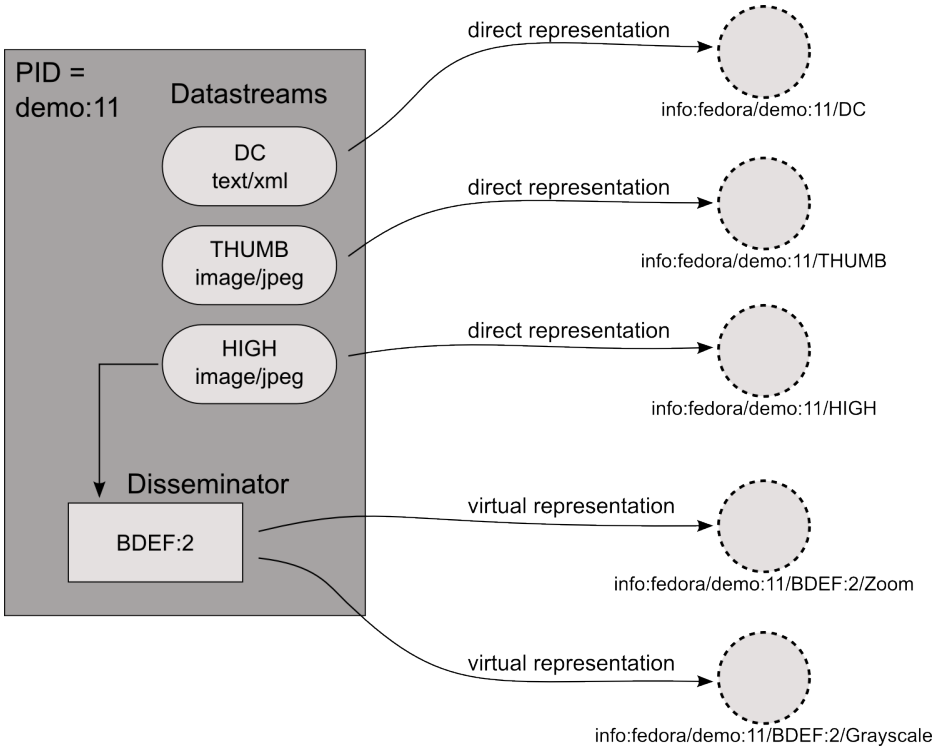


Abbildung 2.20: *Das digitale Objekt* `demo:11` *mit drei internen und zwei berechneten Datenströmen. Im bDef-Objekt* `BDEF:2` *sind zwei Methoden definiert:* `Zoom` *und* `Grayscale`*. Der Disseminator verwendet den Datenstrom* `HIGH` *als Eingabe und erweitert das Objekt um diese zwei Methoden.*

2.3.5 FOXML

Fedora speichert seine digitalen Objekte als XML-Daten. Diese Daten folgen dem FOXML-Schema [31], welches eine direkte Umsetzung des Fedora Digital Object Models ist. Zur Illustration der Daten sei hier ein (stark gekürztes) FOXML gezeigt.

Das Objekt in diesem Beispiel hat die PID `content:1`, was bedeutet, dass das Objekt im Namespace `content` abgelegt ist, und in diesem Namespace den Identifier `1` hat. Der Datenstrom `DC` ist die Beschreibung des Objekts im Dublin-Core Metadaten Format. Das Objekt besitzt einen binären Datenstrom vom MIME-TYPE `image/jpeg`.

```
<?xml version="1.0" encoding="UTF-8"?>
<foxml:digitalObject PID="content:1"
  fedoraxsi:schemaLocation="info:fedora/fedora-system:def/foxml# http://www.
    fedora.info/definitions/1/0/foxml1-0.xsd"
  xmlns:audit="info:fedora/fedora-system:def/audit#"
  xmlns:fedoraxsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:foxml="info:fedora/fedora-system:def/foxml#">
```

```
<!-- Dublin Core Datastream -->
<foxml:datastream CONTROL_GROUP="X" ID="DC" STATE="A" VERSIONABLE="true">
  <foxml:datastreamVersion CREATED="2008-04-14T14:10:43.812Z" ID="DC.1" LABEL="
     Dublin Core Metadata"
     MIMETYPE="text/xml" SIZE="226">
     <foxml:contentDigest DIGEST="none" TYPE="DISABLED"/>
     <foxml:xmlContent>
       <oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http
          ://www.openarchives.org/OAI/2.0/oai_dc/">
         <dc:title>Picture of a house</dc:title>
         <dc:identifier>content:1</dc:identifier>
       </oai_dc:dc>
     </foxml:xmlContent>
  </foxml:datastreamVersion>
</foxml:datastream>

<!-- Managed Datastream (Base64 format) -->
<foxml:datastream CONTROL_GROUP="M" ID="DS1" STATE="A" VERSIONABLE="true">
  <foxml:datastreamVersion CREATED="2008-04-14T14:09:13.593Z" ID="DS1.0" LABEL
     ="House in normal Resolution"
     MIMETYPE="image/jpeg" SIZE="0">
     <foxml:contentDigest DIGEST="none" TYPE="DISABLED"/>
     <foxml:binaryContent>
              /9j/4AAQSkZJRgABAQEBLAEsAAD/2
                wBDAAUDBAQEAwUEBAQFBQUGBwwIBwcHBw8LCwkMEQ8SEhEPERET
                     <<< image-data of datastream DS1 removed >>>
              2gZqE2NrgLGZIx97B/hqGR7hCdrA5GV5qZriTG519unb29aOZlc6fQ//2Q==
           </foxml:binaryContent>
  </foxml:datastreamVersion>
</foxml:datastream>

</foxml:digitalObject>
```

Im Abschnitt A.1.6 findet sich ein vollständiges FOXML Dokument, das aus einem Dublin-Core-, zwei binären und einen AUDIT-Datenstrom besteht.

2.3.6 Volltextsuche: Generic Search

Fedora bietet nur eine einfache Suche nach indizierten Feldern an (Basic Search). Dies sind u.a. der Titel und PID eines Objektes. Mit dem Ressource Index können Daten über ihre Relationen gesucht werden. Sehr oft reichen diese beiden Varianten vollkommen aus. Was aber wenn nicht? Stellen wir uns vor, die Objekte enthalten anstatt Bilder (wie in Abbildung 2.15) Texte. Beispielweise im PDF-Format, oder als Plain-Text oder als Latex-Source[19]. Wäre es dann nicht sinnvoll auch innerhalb dieser Texte, also *in* den Datenströmen selbst, suchen zu können?

Für diese Aufgabenstellung wurde das Generic Search Service – kurz: *GSearch* – entwickelt. GSearch läuft, so wie Fedora, als Web-Service und bietet sowohl ein REST als auch ein SOAP Interface. GSearch indiziert dabei alle textuellen Daten des Fedora Repository. Das sind jene Datenströme, die als `internal XML` oder `Managed Contet` abgespeichert sind. Diese beiden Typen werden direkt in den FOXML-Daten und somit direkt im Repository hinterlegt.

[19] Der Latex-Source kann mit der Hilfe eines Web-Services in PostScript oder PDF umgewandelt werden. Fedora kann dies dynamisch bewerkstelligen: mit den Disseminatoren. Siehe Abschnitt 2.3.4

Die eigentliche Indizierung, Speicherung und Suche nach Daten übernimmt dabei ein Plugin, dass für eine Volltext-Suche geeignet ist. GSearch implementiert die Volltextsuche nicht selbst, sondern tritt als Vermittler zwischen dem Fedora-Repository und der eigentlichen Volltext Such-Engine auf. Derzeit unterstützt GSearch die folgende Plugins bzw. Volltext Such-Engines:

- Apache Lucene
- Solr
- Zebra

Abbildung 2.21 zeigt die grundlegende Architektur von GSearch. Auf der Graphik ist zu erkennen, dass die Volltextsuche nicht in Fedora integriert ist, sondern parallel dazu existiert. Die Indizes und Daten der Volltextsuche werden weder von Fedora berechnet noch in dessen Repository abgelegt. Auch der Zugriff auf die Volltextsuche läuft nicht über die beiden Fedora Schnittstellen API-A und API-M, sondern über die Schnittstellen von GSearch.

2.3.7 Zusätzliche Web-Service in Fedora

Fedora bietet neben den eigenen Web-Services für das Repository (API-A und API-M) noch drei weitere Web-Service auf REST-Basis an. Diese drei Web-Services haben selbst nichts mit dem Fedora-Repository zu tun, können aber in Verbindung mit Disseminatoren sehr nützlich sein.

FOP

Dieses Service bietet einen Formatting Objects Processor (FOP[20]) an. Als Eingabe erwartet das Service eine URL auf ein XML-Dokument im FOP-Syntax. Die Ausgabe ist ein Dokument in Adobe's PDF-Format.
Beispiel:
`http://localhost:8080/fop/FOPServlet↩`
`?source=/fedora/get/demo:4711/DS_FOP`

[20]FOP ist eine Programm, das aus einem XML Dokument mit Hilfe eines XSL-FO Stylesheets (Extensible Stylesheet Language – Formatting Objects) druckbare Seiten generiert. Das XML Dokument beinhaltet dabei Anweisungen um Texte, Linien und andere graphische Elemente auf einer Seite anzuordnen. [29].

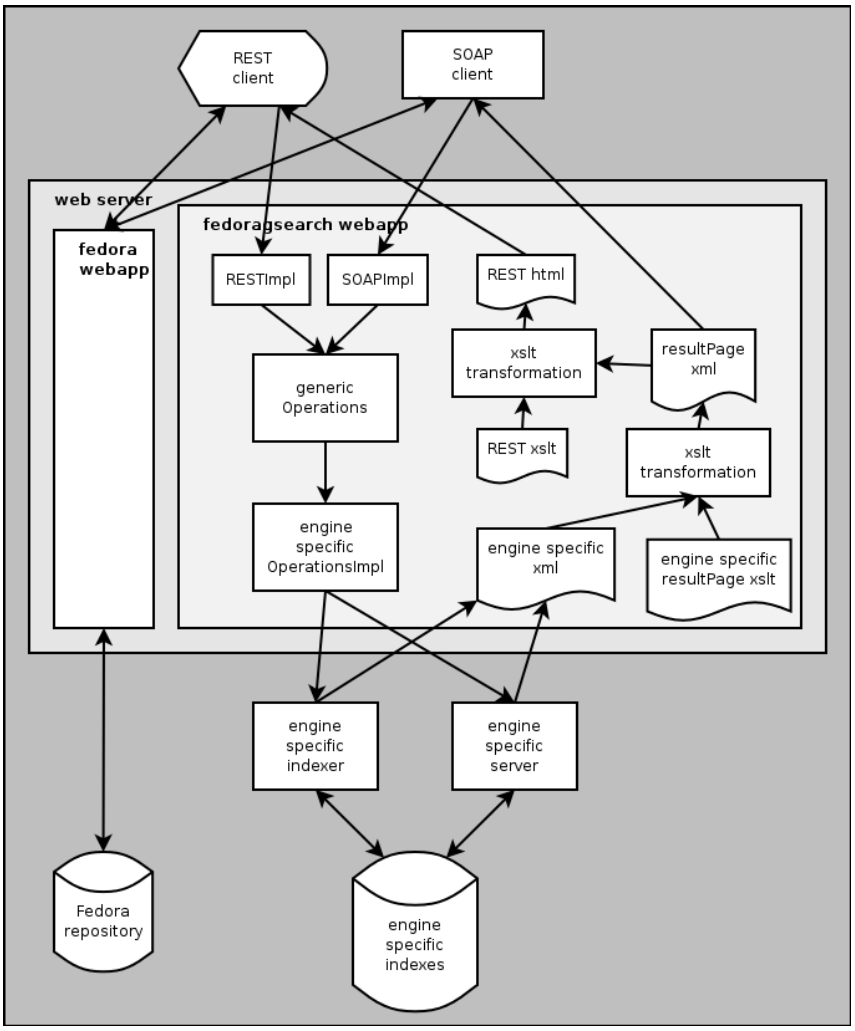


Abbildung 2.21: *Generic Search Service: Architektur-Übersicht. GSearch bietet eine Volltextsuche für Fedora an. Das Service existiert parallel zum Fedora Repository und ist nicht Teil der Standard Fedora Distribution.*

ImageManipulation

Dieses Service bietet eine Reihe von Funktionen an, um Bilder zu manipulieren. Die Funktionen reichen von konvertieren über skalieren bis hin zu Pixelveränderungen (umwandeln in Graustufen, hinzufügen eines Wasserzeichens). Als Eingabe erwartet das Service eine URL zu den Bilddaten, die bearbeitet werden sollen und die gewünschte Operation.
Beispiel:
`/imagemanip/ImageManipulation?url=/fedora/get/demo:700/DS1↩`

`&op=grayscale`

Saxon

Dieses Web-Service dient als Schnittstelle zu der Saxon-API[21]. Damit lassen sich XML-Dokumente mit Hilfe von XSLT-Stylesheets in eine beliebige andere Form transformieren. Als Eingabe erwartet das Service sowohl eine URL auf ein XML-Dokument als auch auf ein XSLT-Stylesheet. Auch der Server des Fedora-Annotation-Services nutzt dieses Web-Service, um seine Ausgabe gegebenfalls zu transformieren (z.B. in das Annotea-Format).
Beispiel:
`/saxon/SaxonServlet?source=/fedora/get/demo:4711/DS_FOP↩`
`&style=/fedora/get/demo:4711/DS_STYLE`

2.4 Zusammenfassung

In diesem Kapitel wurden die Begriffe Digital Libraries und Annotationen genauer erläutert. Es wurden aktuelle Website-, Bild- und Video-Annotationstools vorgestellt. Die jeweiligen Features wurden diskutiert und die Tools wurden schlussendlich miteinander in einer übersichtlichen, tabellarischen Form verglichen. Der dritte Teil des Kapitels behandelte das verwendete Backend – das Fedora-Repository. Das Hauptaugenmerk fiel dabei vor allem auf jene Teile des Repositories, die auch für diese Arbeit von besonderen Interesse sind. Dazu zählen unter anderem das Fedora Digital Object Model, der Zugriff auf das Repository (API-A und API-M), sowie die Relationen zwischen einzelnen Objekten (RELS-EXT).

[21] Saxon ist ein XSLT und XQuery Prozessor. [13].

Kapitel 3

Das Fedora-Annotation-Service

Die Annotations-Features sind oft direkt in die Webseiten oder Plattformen integriert. Die Annotationen können auf diese Art zwar direkt auf die vorhandenen Medientypen abgestimmt werden, jedoch führt die Hinzunahme neuer Medientypen unweigerlich zu großen Änderung des Gesamtsystems.

Ein allgemeines Annotations-Framework sollte daher unabhängig vom Ursprungssystem arbeiten. Das hat den Vorteil, dass sich ein solches Framework leicht in ein bestehendes System integrieren lässt (vgl. Kapitel 5), die Hinzunahme eines neuen Medientypen muss dadurch auch nicht zu einer Änderung des bereits bestehenden Systems führen.

Dieses Kapitel beschreibt einen möglichen Lösungsweg für diese Problemstellung. Den Anfang macht ein grober Überblick über die Architektur des Fedora-Annotation-Services (Abschnitt 3.1). Danach wird geklärt, warum das Fedora-Repository als Backend für das Framework eingesetzt wird (Abschnitt 3.2) und abschließend werden die einzelnen Komponenten des Fedora-Annotation-Services genauer vorgestellt (Abschnitt 3.3 bis 3.5). Dabei wird aufgezeigt wie der Server, die Annotations-Metadaten und die Clienten prinzipiell aufgebaut sind.

3.1 Architektur

Ein allgemeines Annotations-Framework muss unabhängig vom Ursprungssystem, in dem es eingebettet ist, arbeiten. Daher besteht das Fedora-Annotation-Service aus mehreren Komponenten, die das Ursprungssystem um

die Annotations-Features erweitert. Eine Serverkomponente (vgl. Abschnitt 3.4) übernimmt dabei die Aufgabe die Annotationen zu verwalten. Die Annotationen werden in einem Backend (vgl. Abschnitt 3.2) gespeichert, der Server kann sie selektieren, anlegen, ändern oder löschen. Dazu stellt er seinen Clienten ein einfaches Web-Service zur Verfügung. Die Annotations-Metadaten definieren die Struktur und Bedeutung (Semantik) der Annotationsdaten, wobei jeder Medientyp eigene, spezielle Daten, die nur für ihn relevant sind, besitzt. Jeder Medientyp (z.B. Bilder, Videos oder Texte) benötigt auch einen eigenen, auf den Medientypen spezialisierten, Annotationsclienten (vgl. Abschnitt 3.5).

Abbildung 3.1 zeigt die Architektur des Fedora-Annotation-Services im Überblick. Im Fedora-Repository sind drei verschiedene Medienobjekte gespeichert. Die drei Annotationsobjekte mit der PID `annotation:8, 9 und 12` verweisen auf dabei das annotierte Medienobjekt mit der PID `content:1`; `annotation:2, 5 und 25` verweisen auf das Medienobjekt `content:2`. Der Client greift ausschließlich über den Annotations-Server auf die Annotationen zu. Für ihn soll auch der Umstand, dass das Fedora-Repository als Backend zum Einsatz kommt, unerheblich sein.

3.2 Das Fedora-Repository als Backend

Das Fedora-Annotation-Service nutzt aus den folgenden Gründen das Fedora-Repository für sein Backend[1]:

- Fedora ist ein Repository, das in der Lage ist, beliebigen Content (Daten) abzuspeichern. Dieser Content kann alles mögliche sein: es können Texte, Bilder, Videos oder sonstige multimediale Inhalte sein.
- Die Daten unterliegen einer Versionierung und es ist möglich die einzelnen Digitalen Objekte zueinander in eine semantische Relation zu stellen (vgl. Abschnitt 2.3.3).
- Das Fedora-Repository wird sehr oft als Backend in diversen Digital Libraries eingesetzt, wie zum Beispiel in der Encyclopedia of Chicago[2] oder dem Projekt Phaidra der Universität Wien[3].

Die Zugriffe auf das Repository sind dabei klar über Web-Services (REST und SOAP) geregelt, wodurch eine prinzipielle technische Interoperabilität

[1] Unter den Begriff *Backend* soll hier jener Teil eines Softwarepakets verstanden werden, der für das Abfragen, Speichern, Ändern und Löschen von Daten zuständig ist.

[2] http://www.encyclopedia.chicagohistory.org

[3] http://phaidraservice.univie.ac.at

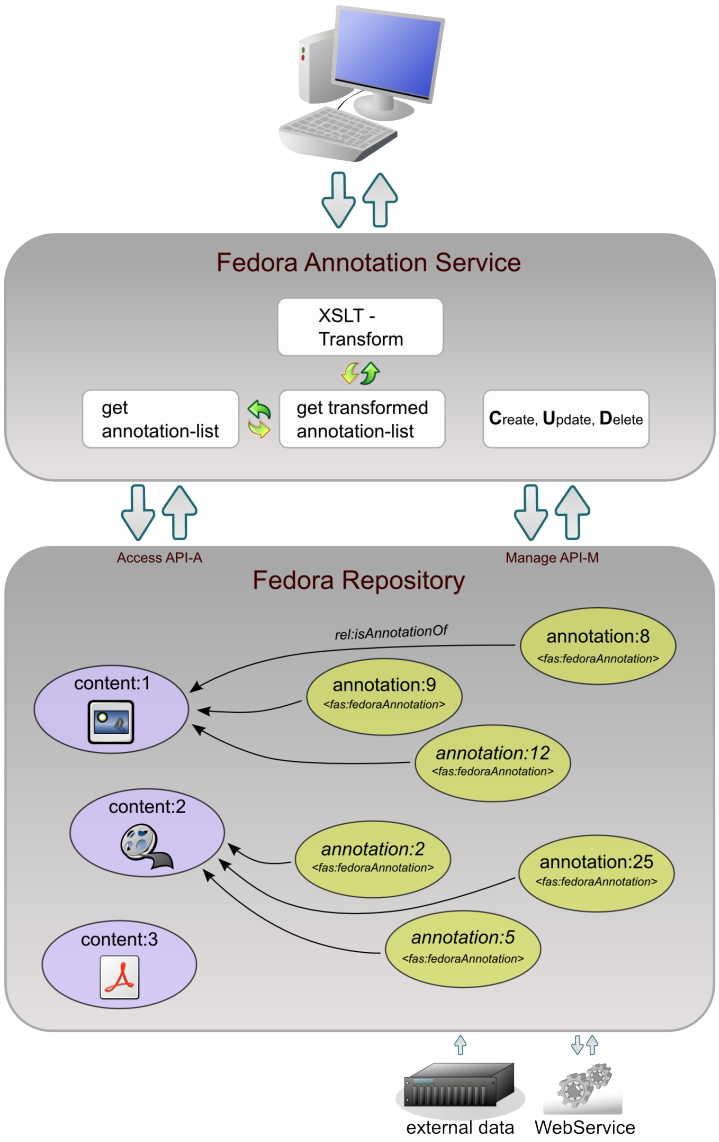


Abbildung 3.1: *Architektur des Fedora-Annotation-Service. Clienten rufen ausschließlich die Funktionen des Server auf. Dieser wiederum benutzt die Web-Services von Fedora, um die Annotationsobjekte zu selektieren oder zu manipulieren. Durch die konfigurierbaren Transformationen ist der Server in der Lage eine generierte Liste von Annotationen in eine beliebig andere Form zu konvertieren.*

gewährleistet ist. Da nun viele Digital Libraries Fedora verwenden, liegt es nahe, dieses System ebenso als Backend für das Annotations-Framework zu nutzen. Dadurch ergeben sich einige Synergien und Vorteile, die vor allem die Integration des Fedora-Annotation-Services in eine bestehende Digital Library betreffen.

Nutzt eine Digital Library Fedora als Backend, so speichert sie ihre Daten bzw. Medieninhalte in Fedora ab. Das Fedora-Annotation-Service seinerseits kann auf bereits bestehende Medienobjekte verweisen, indem es ein neues Objekt für die Annotation anlegt und dieses Annotationsobjekt mit dem zu annotierenden Medienobjekt semantisch mit der nötigen Relation (`isAnnotationOf`) verknüpft. Alle Annotationsobjekte, die ein bestimmtes Medienonjekt annotieren, können dabei einfach durch eine Suche im Ressource-Index von Fedora (vgl. Abschnitt 2.3.3) abgefragt werden. Das heißt, dass Fedora in der Lage ist, alle Annotationen eines bestimmten Medienobjekts aus seinem Datenbestand zu selektieren. Das Fedora-Annotation-Service muss somit diese Selektion nicht selbst implementieren, sondern kann sich auf die bereits vorhandenen Features des Repositories stützen.

Allerdings verwenden nicht alle Digital Libraries das Fedora-Repository als Backend. In diesem Fall liegt der Vorteil der Entscheidung, Fedora als Backend für das Fedora-Annotation-Service zu nutzen, nicht unmittelbar auf der Hand. Aber auch hier kann Fedora die Integration des Frameworks erleichtern. Fedora muss nicht unbedingt die Daten seiner Medienobjekte in der eigenen Datenbank speichern. Es ist auch möglich, dass Fedora die Daten eines Objekts von einem fremden Rechner bzw. Server bezieht (siehe auch Abschnitt 2.3.2). Für einen Clienten, der solche externen Daten anfordert, macht es keinen Unterschied wo die Daten schlussendlich gespeichert sind. Fedora nimmt die Anfrage entgegen, ruft die externen Daten von dem fremden Rechner ab und schickt sie anschließend an den Clienten weiter. Für den Clienten sieht es dabei so aus als ob die Daten sehr wohl innerhalb von Fedora gespeichert gewesen wären.

Das Fedora-Annotation-Service benötigt zum Erfüllen seiner Aufgabe stets das zu annotierende Medienobjekt in der Datenbank von Fedora, damit es die notwendige semantische Relation zwischen diesem Medienobjekt und dem Annotationsobjekt anlegen kann. Sind die Mediendaten nun nicht in Fedora gespeichert, weil die Digital Library nicht das Fedora-Repository verwendet, dann liegt auch das Medienobjekt selbst nicht in der Datenbank von Fedora. Somit kann auch keine semantische Relation geschaffen werden. Aber Fedora ist, wie im vorigen Absatz erläutert, in der Lage auch Objekte zu verwalten deren Daten auf einem externen Rechner liegen. Sollte also das Framework in einem System eingebunden werden, dass nicht Fedora verwendet, muss für jedes (annotierte) Medienobjekt der Library ein

Vertreter- bzw. Proxy-Objekt im Fedora-Repository angelegt werden dessen Daten von einer externen Quelle – der Library – kommen. Die Erstellung der Proxy-Objekte im Backend des Fedora-Annotation-Services kann dabei einmalig von einem Administrator vorgenommen werden. Oder aber der Server des Frameworks kümmert sich darum das fehlende Proxy-Objekte bei Bedarf automatisch angelegt werden[4]. In Abschnitt A.2.2 wird gezeigt, wie Objekte mit externen Daten in Fedora angelegt werden können.

3.3 Annotations-Metadaten

Die Metadaten stellen die eigentlichen Daten einer Annotation dar. Eine Annotation legt dabei die Daten als einen internen XML-Datenstrom innerhalb seines digitalen Objekts ab, das Aussehen der XML-Daten wird dabei von einem eigenen XML-Schema bestimmt. Abbildung 3.2 gibt einen Überblick über den Aufbau des Annotations-XML-Schemas.

Für unterschiedliche Medientypen (z.B. Bilder oder Videos) müssen unterschiedliche Annotationsdaten gespeichert werden. Dies gilt insbesonders für die Adressierung von Medienfragmenten. Ein Fragment dient dazu einen ausgewählten Bereich eines Medieninhalts zu annotieren. Bei einem Bild kann dies etwa ein Bildausschnitt sein; bei einem Video ein Bereich der Zeitachse und ein Bereich des Bildes. Aus diesem Grund definiert das Annotations-XML-Schema für jeden Medientypen ein eigens Annotations-Format. Jeder Annotations-Typ baut dabei auf einer einfachen Annotation (simpleType), das keine Fragmentierung kennt, auf und erweitert diese. Eine Annotation muss nicht notwendigerweise ein Fragment besitzen. Ein Benutzer kann durchaus eine Annotation anlegen wollen, ohne sich auf einen bestimmten Bereich beziehen zu wollen. Tabelle 3.1 zeigt alle derzeit definierten Annotationstypen mit den dazugehörigen Fragmenten.

Annotations-Typ	**Definiertes Fragment**
simple	Besitzt kein Fragment
image	Ein Rechteck im Bild
video	Ein Rechteck im Bild und ein Bereich mit Start- und Stoppzeit auf der Zeitachse
xml	Ein Bereich der mit zwei XPointer definiert wird

Tabelle 3.1: *Die Annotationstypen mit den dazugehörigen Fragmenten, so wie sie in den Metadaten definiert sind.*

Soll ein neuer Medientyp ermöglicht werden, so muss dieser dem Schema hinzugefügt werden. Der Aufbau des Schemas wurde daher so gestaltet, dass

[4]Das dynamische Anlegen von Proxy-Objekten wurde im Rahmen dieser Diplomarbeit nicht implementiert. Es soll hier nur die prinzipielle Machbarkeit aufgezeigt werden.

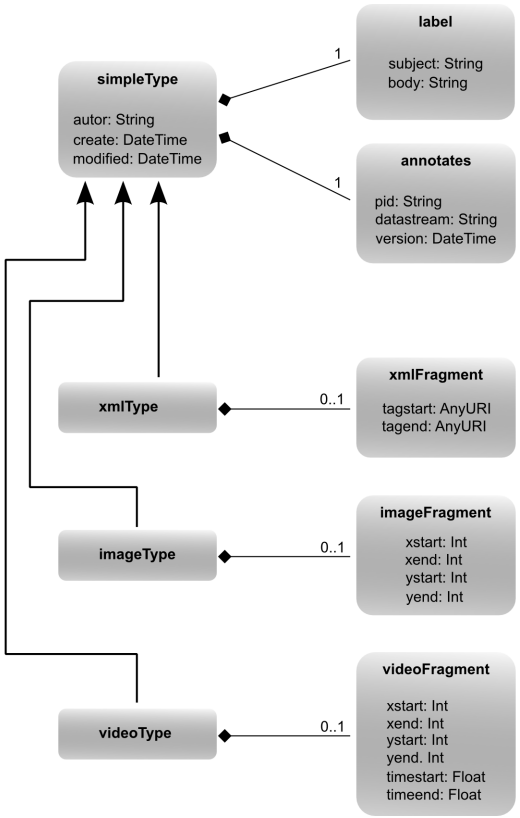


Abbildung 3.2: *Überblick über die Annotations-Metadaten. Jeder Medientyp besitzt seine eigenen Metadaten mit einer eigenen Fragmentdefinition.*

eine nachträgliche Erweiterung leicht durchzuführen ist. Bereits vorhandenen Medientypen des Schemas werden dadurch nicht verändert.

Ein Annotations-XML-Dokument besteht entweder aus genau einer Annotation oder aus einer Liste einzelner Annotationen. Eine einzelne Annotation wird ausschließlich innerhalb eines Annotationsobjekts gespeichert. Hingegen liefert der Server des Frameworks immer eine Liste von Annotationen, wenn ein Client die Annotationen eines Medienobjekts anfordert. Damit der Client aus einer Annotationsliste jede einzelne Annotationen eindeutig zuordnen kann (zum Beispiel wenn eine Annotation von einem Benutzer geändert wurde und diese Änderung dann gespeichert werden soll), hinterlegt der Server zusätzlich in den einzelnen Annotation der Liste die jeweilige, eindeutige PID der Annotation.

Listing 3.1 zeigt ein XML-Dokument eines Annotationsobjekts, wobei die Daten für ein Bild (`fas:image`) bestimmt sind. Der Text der Annotation (`fas:subject`) lautet 'Ampel' und die Annotation besitzt ein Fragment (`fas:fragment`). Das bedeutet, dass diese Annotation nicht für das ganze Bild sondern nur für einen rechteckigen Teilbereich des Bildes bestimmt ist.

Listing 3.1: Daten einer Bildannotation

```
<?xml version="1.0" encoding="UTF-8"?>
<fas:fedoraAnnotation xmlns:fas="urn:fedora:annotation" xmlns:xsi="
  http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:
  fedora:annotation http://localhost:8080/fedoraAnnotations/Definition
  /FedoraAnnotation.xsd">
  <fas:annotation>
    <fas:image>
      <fas:author>fedoraAdmin</fas:author>
      <fas:create>2008-08-16T10:55:15</fas:create>
      <fas:label>
        <fas:subject>Ampel</fas:subject>
      </fas:label>
      <fas:annotates>
        <fas:pid>content:100</fas:pid>
        <fas:datastream>DS1</fas:datastream>
      </fas:annotates>
      <fas:fragment>
        <fas:xstart>243</fas:xstart>
        <fas:xend>261</fas:xend>
        <fas:ystart>141</fas:ystart>
        <fas:yend>181</fas:yend>
      </fas:fragment>
    </fas:image>
  </fas:annotation>
</fas:fedoraAnnotation>
```

Zum Vergleich findet sich ein XML-Dokument mit einer Liste von Annotationen in Listing 4.1.

3.4 Server

Die Serverkomponente des Fedora-Annotation-Services übernimmt die Aufgabe die Annotationen im Fedora-Repository zu verwalten. Alle Anfragen der Clienten bezüglich Annotationen müssen über diesen Server laufen. Dazu stellt er seinen Clienten ein einfaches REST Interface zur Verfügung [40] [5].

Wird eine neue Annotation angelegt, so erzeugt der Server im Fedora-Repository ein neues digitales Objekt. Dieses Objekt hat einen Datenstrom mit dem Namen `DS`, in dem die eigentlichen Annotationsdaten gespeichert werden. Zusätzlich hat dieses neue Objekt einen RELS-EXT Datenstrom, in dem die semantische Relation `isAnnotationOf` mit dem zu annotierenden Medienobjekt gespeichert wird.

Wenn ein Client die Liste aller Annotationen zu einem Medienobjekt anfordert, selektiert der Server in einem ersten Schritt die betroffenen Annotationsobjekte mit Hilfe des Ressource-Indizes. Im zweiten Schritt ruft der Server für die selektierten Annotationsobjekte die Annotationsdaten ab. Die Gesamtheit der einzelnen Daten ergibt dann die geforderte Liste, welche entweder direkt an den aufrufenden Clienten zurückgegeben oder noch zusätzlich transformiert werden kann. Diese Transformationen sind nicht fix im Server codiert, sondern können in einer Konfigurationsdatei definiert werden. Eine Transformation besteht dabei immer aus drei Elementen: einem eindeutigen Namen, unter der diese Transformation angesprochen werden kann, einem XML-Stylesheet, das die eigentliche Transformation der zuvor generierten Liste übernimmt und aus einer URL, die auf das Fedora Saxon Web-Service zeigt (vgl. Abschnitt 2.3.7). Dieses Saxon Web-Service wird dann vom Server aufgerufen, damit es die zuvor generierte XML-Liste mithilfe des angegebenen XML-Stylesheets in das gewünschte Format transformiert. So ist es prinzipiell möglich die Liste aller Annotationen eines Medienobjekts in jede beliebige Form an den Clienten zu senden.

3.5 Client

Der Client stellt die Verbindung zwischen dem Framework und den Benutzern her. Seine Aufgabe ist es das Medienobjekt und die dazugehörigen Annotationen darzustellen. Ein Benutzer muss in der Lage sein, neue Annotationen anzulegen, bestehende zu ändern oder zu löschen.

Jeder Medientyp benötigt einen eigenen Clienten, der auf genau dieses Medium spezialisiert ist. Ein Bild zu annotieren funktioniert anders als einen

Text oder ein Video zu annotieren. Ein Client muss diesen Umstand Rechnung tragen. Er muss in der Lage sein das Medium, auf das er spezialisiert ist, darzustellen. Ein Client für Bilder muss Bilddaten laden und darstellen können. Analog dazu muss ein Client, der auf Videos spezialisiert ist, in der Lage sein Videos zu laden und abzuspielen. Daher wird für jeden Medientypen, den das Framework unterstützen soll, ein eigener Client benötigt. Soll ein neuer Medientyp unterstützt werden, so wird auch ein neuer Client benötigt, der auf eben diesem neuen Medientypen spezialisiert ist. Dadurch ist es möglich das Framework zu erweitern, ohne dass dies zu einer Änderung des bereits bestehenden Systems führt.

Abbildung 3.3 zeigt den Clienten für Bilder, den Annotations-Server und das Fedora-Repository. Die Pfeile geben den jeweiligen Datenfluss zwischen Server und Clienten an. Der Client bezieht seine Daten aus zwei unterschiedlichen Kanälen: vom Fedora-Repository lädt er den Medieninhalt – in diesem Beispiel das Bild. Vom Annotations-Server wird die Liste aller Annotationen, die zu diesem Medieninhalt gehören, geladen.

3.6 Zusammenfassung

Das Fedora-Annotation-Service nutzt das Fedora-Repository als Backend um seine Daten zu speichern. Der Grund dafür ist, dass zum einem viele Digital Libraries selbst Fedora als Backend verwenden. Zum anderen ist der Zugriff auf die gespeicherten Daten klar über Web-Services geregelt. Dadurch kann ein Objekt über eine eindeutige URI (durch Fedoras API-A) abgerufen werden. Der Annotations-Server verwaltet die Annotationsdaten und speichert sie im Fedora-Repository. Eine Annotation besteht dabei immer aus einem digitalen Objekt, das die eigentlichen Annotations-Metadaten und eine semantische Relation auf das zu annotierende Objekt beinhaltet. Der Server ist in der Lage alle Annotationen zu einem Medienobjekt als Liste an einen Clienten zu liefern. Diese Liste kann entweder direkt im internen XML-Format oder aber in einer beliebigen anderen Form zurückgegeben werden, wobei diese Formen durch XSLT Transformationen erreicht werden.

Das Framework soll mit den verschiedensten Medientypen umgehen können. Unterschiedliche Medientypen brauchen aber eine unterschiedliche Behandlung bzw. ein unterschiedliches User Interface. Daher wird für jeden, vom Framework unterstützten, Medientyp ein eigener Client benötigt. Dieser Client muss auf seinen eigenen Medientypen spezialisiert sein.

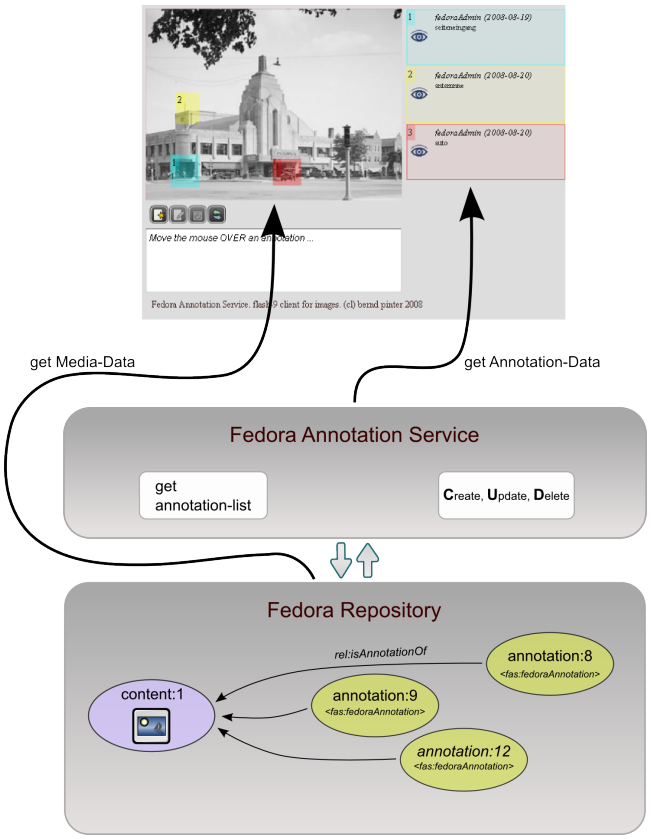


Abbildung 3.3: *Datenfluss des Annotations-Clienten. Der Client bezieht seine Daten von zwei Stellen. Die Mediendaten (in diesem Beispiel ein Bild) werden direkt vom Fedora-Repository geladen. Die Annotationsdaten werden vom Annotations-Server im internen XML-Format abgerufen.*

Kapitel 4

Implementierung

Dieses Kapitel widmet sich den Details der Implementierung des zuvor beschriebenen Annotations-Frameworks. Der erste Teil beschäftigt sich mit dem Annotations-Server und seinen Details. Welche Möglichkeiten er bietet und wie er von seinen Clienten aufgerufen werden kann. Der zweite Teil geht auf die beiden implementierten Clienten (Bild und Video) ein. Speziell wird dabei auf die verwendete Plattform – dem Flash9–Browserplugin – eingegangen und es soll geklärt werden, warum die Entscheidung zugunsten dieser Technologie gefallen ist. Den Abschluss bildet ein Vergleich der bereits in Abschnitt 2.2.4 vorgestellten Annotationstools mit dem Fedora-Annotation-Service. Dadurch soll die Praxistauglichkeit des Gesamtsystems auf einen Blick gezeigt werden.

4.1 Fedora-Annotation-Service – Server

Der Server wurde, so wie auch das Fedora-Repository, vollständig in Java implementiert. Alle benötigten Java-Bibliotheken sind bereits im Source Code inkludiert, wodurch keine zusätzlichen Abhängigkeiten während des Compile-Vorgangs entstehen. Das Ergebnis des Übersetzungsvorgangs ist ein Java `war`-Datei. Diese Datei kann direkt von einem Java Servlet Container verarbeitet werden.

Der Server hat die Aufgabe die Annotationen zu speichern, zu verwalten und abzufragen. Dazu stellt er seinen Clients ein einfaches REST-Interface zur Verfügung, das nur HTTP-GET und HTTP-POST verwendet. HTTP-GET sucht nach Annotationen und retourniert eine Annotationsliste (wie im Schema definiert). HTTP-POST dient dem Anlegen, Ändern und Löschen

einzelner Annotationen. Der Server speichert alle Daten in einem Fedora-Repository ab, wobei für jede neue Annotation ein neues Digitales Objekt im Repository erzeugt wird. Die Verbindung zwischen dem Annotationsobjekt und dem zu annotierenden Medienobjekt wird mit einer semantischen Relation (`isAnnotationOf`) hergestellt. Daraus folgt auch, dass das zu annotierende Medienobjekt bereits im Fedora-Repository als Digitales Objekt vorhanden sein muss. Dies bedeutet im weiteren, dass sich das Annotations-Framework nahtlos in ein System integrieren lässt, welches bereits das Fedora-Repository als Backend verwendet (wie zum Beispiel die Encyclopedia of Chicago[1] oder das Projekt Phaidra der Universität Wien[2]).

Seinen Clienten stellt der Server die folgenden fünf Operationen zur Verfügung:

- Abrufen der Annotationen im internen Format
  Eine der Hauptaufgaben des Servers ist es, eine Liste aller Annotationen zu einem Medienobjekt zu liefern. Diese Operation liefert die Annotationsdaten im internen XML-Format und das Aussehen dieser Liste ist im XML-Schema der Annotations-Metadaten definiert.

- Abrufen der Annotationen in einem transformierten Format
  Der Server ist in der Lage die Liste der Annotationen in ein anderes Format zu transformieren bevor er sie an den Clienten schickt. Dies erhöht die Interoperabilität zu anderen Systemen. Die Transformationen können dabei in der Konfigurationsdatei des Server definiert werden. Im Zuge der Implementierung wurden bereits zwei verschiedene Transformationen erzeugt. Die Transformation mit dem Namen `html` konvertiert die Liste nach XHTML. Jene mit den Namen `annotea` konvertiert die Liste in das Annotea RDF/XML Format.

- Anlegen einer neuen Annotation
  Diese Operation erzeugt eine neue Annotation. Dabei wird im Backend des Servers (im Fedora-Repository) ein neues Digitales Objekt im Namespace `annotation` erzeugt. Dieses neue Digitale Objekt besteht aus den eigentlichen Annotationsdaten und einer semantischen Relation zu dem annotierten Medienobjekt. Die semantische Relation ist dabei vom Typ `isAnnotationOf`. Dieser Typ ist bereits in der Grundontologie von Fedora definiert.

- Ändern einer bestehenden Annotation
  Die Daten einer Annotation können sich im Laufe der Zeit ändern, wenn ein Benutzer zum Beispiel den Text der Annotation ändert. Diese Operation ändert die Annotationsdaten, in dem sie die gesamten

[1] http://www.encyclopedia.chicagohistory.org
[2] http://phaidraservice.univie.ac.at/

Annotationsdaten des betroffenen Annotationsobjekts neu schreibt. Selbst wenn nur einzelne Teile der Daten geändert werden, wird trotzdem der gesamte Datenstrom, der die Annotationsdaten beinhaltet, neu geschrieben.

- Löschen einer bestehenden Annotation
  Eine bestehende Annotation kann mit dieser Operation gelöscht werden. Dabei wird stets das gesamte Annotationsobjekt aus dem Fedora-Repository gelöscht. Dadurch werden die Annotationsdaten und die semantische Relation des Annotationsobjekts gelöscht.

In den folgenden Unterkapiteln werden die einzelnen Operationen genauer beschrieben. Es soll geklärt werden wie die Operationen von den Clienten aufgerufen werden können. Jede Operation sendet ein Ergebnis an den aufrufenden Clienten, damit dieser über den Erfolg bzw. Misserfolg der angestoßenen Operation informiert ist. Auch diese Ergebnisse des Servers werden hier diskutiert.

4.1.1 Abrufen der Annotationen im internen Format

Diese Operation wird von den Clienten verwendet, um alle Annotationen eines Medienobjekts zu erhalten. Das Ergebnis ist dabei stets eine Liste aller Annotationen, die allerdings auch leer sein kann. In diesem Fall besitzt das Medienobjekt keine Annotationen.

Angestoßen wird diese Operation über eine HTTP-GET Anfrage an den Server [16] [17]. Das URL-Schema des Aufrufs sieht dabei wie folgt aus:

`/fedoraAnnotations/pid/`*`pid`*

bzw.

`/fedoraAnnotations/pid/`*`pid`*`/type/xml`

Wobei *`pid`* der einzige Parameter ist und die Fedora-PID des annotierten Medienobjekts angibt. Wird der Parameter weggelassen, so retourniert der Server eine Liste von allen gespeicherten Annotationen. Dies ist zugleich das Standardverhalten des Server, wenn eine ungültige Anfrage gestellt wurde.

Beispiele:

`/fedoraAnnotations/pid/content:100`
Liefert die Annotationen zu dem Medienobjekt mit der PID `content:100`.

`/fedoraAnnotations/pid/content:100/type/xml`
Liefert das selbe Ergebnis wie im vorigen Beispiel.

`/fedoraAnnotations`
Liefert alle im Repository gespeicherten Annotationen.

Listing 4.1 zeigt ein XML-Dokument, das vom Server zurückgegeben wurde. Dieses Dokument besteht aus einer Liste aller Annotationen, die das Bild mit der PID `content:100` annotieren. Das erste Annotationsobjekt (`annotation:80`) besitzt ein Bild-Fragment. Diese Annotation soll somit nur für einen bestimmten Bereich des Bildes gelten. Das zweite Annotationsobjekt (`annotation:81`) besitzt kein Fragment und soll daher für das gesamte Bild und nicht nur für einen bestimmten Bereich gelten.

Listing 4.1: Eine Liste von Bildannotationen

```
<?xml version="1.0" encoding="UTF-8"?>
<fas:fedoraAnnotation xmlns:fas="urn:fedora:annotation" xmlns:xsi="http://www.w3.
  org/2001/XMLSchema-instance" xsi:schemaLocation="urn:fedora:annotation http://
  localhost:8080/fedoraAnnotations/Definition/FedoraAnnotation.xsd">
  <fas:annotations>
    <!-- Annotation-List for ObjectModel info:fedora/content:100 -->

    <!-- #0 Annotation Data of info:fedora/annotation:80 -->
    <fas:annotation pid="annotation:80">
      <fas:image>
        <fas:author>fedoraAdmin</fas:author>
        <fas:create>2008-08-16T10:55:15</fas:create>
        <fas:label>
          <fas:subject>Ampel</fas:subject>
        </fas:label>
        <fas:annotates>
          <fas:pid>content:100</fas:pid>
          <fas:datastream>DS1</fas:datastream>
        </fas:annotates>
        <fas:fragment>
          <fas:xstart>243</fas:xstart>
          <fas:xend>261</fas:xend>
          <fas:ystart>141</fas:ystart>
          <fas:yend>181</fas:yend>
        </fas:fragment>
      </fas:image>
    </fas:annotation>

    <!-- #1 Annotation Data of info:fedora/annotation:81 -->
    <fas:annotation pid="annotation:81">
      <fas:image>
        <fas:author>fedoraAdmin</fas:author>
        <fas:create>2008-08-16T10:56:32</fas:create>
        <fas:label>
          <fas:subject>Auto?</fas:subject>
        </fas:label>
        <fas:annotates>
          <fas:pid>content:100</fas:pid>
          <fas:datastream>DS1</fas:datastream>
        </fas:annotates>
      </fas:image>
    </fas:annotation>

  </fas:annotations>
</fas:fedoraAnnotation>
```

4.1.2 Abrufen der Annotationen in einem transformierten Format

Der Server kann die Annotationliste auch in einem beliebigen Format retournieren. Um dies zu bewerkstelligen, generiert er als erstes die angeforderte Liste im internen XML-Format. Diese Liste wird in einem zweiten

Schritt vom Fedora Saxon-Service (vgl. Abschnitt 2.3.7) mit einem geeignetem XSLT-Stylesheet in die gewünschte Form transformiert.

Die möglichen Transformationen werden in der Konfigurationsdatei des Servers definiert, wobei die folgenden Parameter angegeben werden müssen:

- Ein eindeutiger Name, unter der die Transformation für den Clienten abrufbar ist. Der Name der Transformation darf allerdings nicht `xml` lauten, denn diese "Dummy-Transformation" ist für die Liste im internen XML-Format reserviert.
- Eine URL, die den REST Aufruf des Fedora Saxon-Services wiederspiegelt. Diese URL beinhaltet zugleich die URI, unter der das XLST-Stylesheet erreichbar ist.
- Den MIME Type des Ergebnisses der Transformation. Der Server sendet den aufrufenden Clienten das Ergebnis, wobei genau dieser MIME-Type im HTTP-Response Header gesetzt wird. So ist es möglich, dass der Client (z.B ein Webbrowser) das Ergebnis richtig interpretiert.

Im Zuge der Implementierung wurden bereits zwei Transformation definiert: `html` konvertiert die Liste in ein XHTML-Dokument und `annotea` konvertiert die Liste in das Annotea RDF/XML Format. Nachfolgend wird exemplarisch die Konfiguration anhand der Annotea-Transformation gezeigt:

```
...
<transform>
    <name>annotea</name>
    <URLWebService>http://localhost:8080/saxon/SaxonServlet?source
      =http://localhost:8080/fedoraAnnotations/__PID__type/xml&amp
      ;style=http://localhost:8080/fedoraAnnotations/Definition/
      FedoraAnnotation2Annotea.xsl</URLWebService>
    <contentType>text/xml;charset=UTF-8</contentType>
</transform>
...
```

Die gezeigte Transformation trägt den Namen `annotea` und das Ergebnis ist vom MIME-Type `text/xml`, wobei `UTF-8` als Zeichensatz verwendet wird. Der XML-Tag `URLWebService` gibt den REST-Aufruf für das Fedora Saxon-Service an, wobei die Zeichenkette `__PID__` durch jene Fedora-PID ersetzt wird, die der Client aufgerufen hat. Des weiteren ist hier auch die Vorgehensweise des Servers ersichtlich. Er ruft das Saxon-Service auf, wobei die XML-Liste, die transformiert werden soll durch einen nochmaligen Aufruf des Annotation-Servers erzeugt wird. Dies bedeutet, dass ein Aufruf einer transformierten Liste tatsächlich zwei unabhängige Aufrufe des Server bedingt. Der erste Aufruf erfolgt durch den Clienten; der zweite Aufruf erfolgt – implizit – durch den Server selbst.

Auch diese Operation wird über eine HTTP-GET Anfrage an den Server angestoßen [16] [17]. Das URL-Schema des Aufrufs sieht dabei wie folgt aus:

`/fedoraAnnotations/pid/`*`pid`*`/type/`*`name`*

Wobei *pid* die Fedora-PID des annotierten Medienobjekts angibt und *name* der eindeutige Name der gewünschten Transformation ist.

Beispiele:

`/fedoraAnnotations/type/html`
Liefert alle im Repository gespeicherten Annotationen im XHTML Format.

`/fedoraAnnotations/pid/content:100/type/annotea`
Liefert die Annotationen zu dem Medienobjekt mit der PID `content:100` im Annotea RDF/XML Format.

Im Vergleich zu Listing 4.1, in dem die Liste der Annotationen im internen Format zu sehen ist, wird in Listing 4.2 die gleiche Liste im Annotea RDF/XML Format gezeigt.

Listing 4.2: Eine Liste von Bildannotationen im Annotea-Format

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:a="http://
  www.w3.org/2000/10/annotation-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/">

   <rdf:Description about="http://localhost:8080/fedora/get/annotation:80">
   <!-- Annotation Object annotation:80 -->
      <rdf:type resource="http://www.w3.org/2000/10/annotation-ns#Annotation"/>
      <rdf:type resource="http://www.w3.org/2000/10/annotationType#Comment"/>
      <a:annotates rdf:resource="http://localhost:8080/fedora/get/content:100"/>
      <a:context>http://localhost:8080/fedora/get/content:100/DS1</a:context>
      <a:created>2008-08-16T10:55:15</a:created>
      <dc:title>annotation:80</dc:title>
      <dc:creator>fedoraAdmin</dc:creator>
      <dc:date>2008-08-16T10:55:15</dc:date>
      <a:body><![CDATA[Ampel]]></a:body>
   </rdf:Description>

   <rdf:Description about="http://localhost:8080/fedora/get/annotation:81">
   <!-- Annotation Object annotation:81 -->
      <rdf:type resource="http://www.w3.org/2000/10/annotation-ns#Annotation"/>
      <rdf:type resource="http://www.w3.org/2000/10/annotationType#Comment"/>
      <a:annotates rdf:resource="http://localhost:8080/fedora/get/content:100"/>
      <a:context>http://localhost:8080/fedora/get/content:100/DS1</a:context>
      <a:created>2008-08-16T10:56:32</a:created>
      <dc:title>annotation:81</dc:title>
      <dc:creator>fedoraAdmin</dc:creator>
      <dc:date>2008-08-16T10:56:32</dc:date>
      <a:body><![CDATA[Auto?]]></a:body>
   </rdf:Description>

</rdf:RDF>
```

4.1.3 Anlegen einer neuen Annotation

Diese Operation erzeugt ein neues Annotationsobjekt im Fedora-Repository. Dieses neue Digitale Objekt besitzt stets zwei Datenströme:

- Der Datenstrom mit dem Namen `DS` speichert die eigentlichen Annotationsdaten im XML-Format[3]. Daher ist der Typ von diesem Datenstrom `Internal XML` (vgl. Abschnitt 2.3.2).

- Der Datenstrom `RELS-EXT` speichert die semantische Relation zu dem annotierten Medienobjekt, wobei der Typ der Relation aus der Grundontologie von Fedora stammt und `isAnnotationOf` lautet.

Der Aufruf dieser Operation wird mit einer HTTP-POST[4] Anfrage an den Server gestartet. Die Annotationsdaten (ein Annotations XML-Dokument mit genau einer Annotation darin) müssen dabei als POST-DATA an den Server übermittelt werden [16] [17].

Das URL-Schema des Aufrufs sieht wie folgt aus:

`/fedoraAnnotations/create/pid/`*`pid`*

Wobei der Parameter *`pid`* die Fedora-PID des zu annotierenden Medienobjekts angibt.

Beispiele:

`/fedoraAnnotations/create/pid/content:100`
Erzeugt ein neues Annotationsobjekt, wobei die Annotationsdaten direkt aus den POST-DATA des Aufrufs gewonnen werden.

Der aufrufende Client wird vom Server über den Erfolg bzw. Misserfolg der Operation unterrichtet. Dazu sendet er ein *Response XML-Dokument* an den Server zurück. Dieser Response ist in einem eigenen Schema definiert und beinhaltet stets die PID des betroffenen Annotationsobjekts, die ausgeführte Operation und eine Mitteilung, ob die Operation erfolgreich war. Zusätzlich enthält der Response bei einem Misserfolg die Fehlermeldung, die vom Server erzeugt wurde. Wenn ein neues Annotationsobjekt erfolgreich angelegt wurde, enthält die zurückgegebene PID jene, die das neue Annotationsobjekt bekommen hat. Der Client kann sich sofort auf diese neue PID beziehen.

Zur Illustration des Response XML-Dokuments werden hier zwei verschiedene Responses gezeigt. Der erste Response gibt Auskunft darüber, dass ein neues Annotationsobjekt mit der PID `annotation:101` angelegt wurde. Das

[3]Der Name des Datenstroms, in dem die Daten gespeichert werden, kann in der Konfigurationsdatei des Servers geändert werden.

[4]Anders als ursprünglich für REST vorgesehen, verwendet der Server weder HTTP-PUT noch HTTP-DELETE. Der Grund dafür ist, dass nicht alle Implementierungen eines HTTP-Clienten diese Anfragen starten können (z.B. das Flash9-Browserplugin). In Abschnitt 6.1.1 wird noch einmal gesondert auf dieses Problem eingegangen.

zweite Dokument zeigt den Misserfolg der Operation und die dazugehörige Fehlermeldung des Server an:

```
<?xml version="1.0" encoding="UTF-8"?>
<fasr:fedoraAnnotationResponse xmlns:fasr="urn:fedora:annotation:rest_result"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:
   fedora:annotation:rest_result http://localhost:8080/fedoraAnnotations/
   Definition/FedoraAnnotationResponse.xsd">
   <!-- REST interface: get|create|update|delete -->
   <fasr:restOperation>create</fasr:restOperation>
   <!-- Status: ok|error -->
   <fasr:status>ok</fasr:status>
   <fasr:fedoraPid>annotation:101</fasr:fedoraPid>
   <fasr:msg><![CDATA[Create new annotation-object for object pid=content:100]]></
      fasr:msg>
</fasr:fedoraAnnotationResponse>
```

Ein Response XML-Dokument, das den Erfolg der ausgeführten Operation anzeigt. In diesem Fall wurde ein neues Annotationsobjekt mit der PID `annotation:101` *angelegt.*

```
<?xml version="1.0" encoding="UTF-8"?>
<fasr:fedoraAnnotationResponse xmlns:fasr="urn:fedora:annotation:rest_result"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:
   fedora:annotation:rest_result http://localhost:8080/fedoraAnnotations/
   Definition/FedoraAnnotationResponse.xsd">
   <!-- REST interface: get|create|update|delete -->
   <fasr:restOperation>create</fasr:restOperation>
   <!-- Status: ok|error -->
   <fasr:status>error</fasr:status>
   <fasr:fedoraPid>content:100</fasr:fedoraPid>
   <fasr:msg><![CDATA[Cant create new annotation-object for object pid=content
      :100]]></fasr:msg>
   <fasr:error>
      <!-- an error happend! this is the exception message -->
      <fasr:exception><![CDATA[cant ingest new Annotation-Object with fedora's API-
         M WebService! fedora.server.errors.ObjectValidityException:       org.xml.
         sax.SAXParseException: The element type "fas:fedoraAnnotation" must be
         terminated by the matching end-tag "</fas:fedoraAnnotation>".]]></fasr:
         exception>
   </fasr:error>
</fasr:fedoraAnnotationResponse>
```

Ein Response XML-Dokument, das den Misserfolg der ausgeführten Operation anzeigt. In diesem Fall scheiterte das Anlegen eines neuen Annotationsobjekts, da der XML-Tag `fas:fedoraAnnotation` *nicht korrekt geschlossen wurde.*

4.1.4 Ändern einer bestehenden Annotation

Das Ändern einer bestehenden Annotation funktioniert analog wie das anlegen einer neuen Annotation. Die Operation wird mit einem HTTP-POST aufgerufen, die Annotationsdaten werden in den POST-DATA an den Server übermittelt [16] [17]. Auch bei dieser Operation sendet der Server, genauso wie beim Anlegen einer neuen Annotation, dem Aufrufer ein Response XML-Dokument zurück.

Das URL-Schema des Aufrufs sieht wie folgt aus:

`/fedoraAnnotations/update/pid/`*`pid`*

Wobei der Parameter *pid* die Fedora-PID des zu ändernden Annotationsobjekts angibt.

Beispiele:

`/fedoraAnnotations/update/pid/annotation:101`
Ändert die Daten des Annotationsobjekt mit der Fedora-PID `annotation:101`, wobei die Annotationsdaten direkt aus den POST-DATA des Aufrufs gewonnen werden.

4.1.5 Löschen einer bestehenden Annotation

Auch das Löschen einer Annotation funktioniert ähnlich wie das Anlegen – die Operation wird wieder mit einem HTTP-POST aufgerufen [16] [17]. Auch hier sendet der Server dem Aufrufer ein Response XML-Dokument, das Aufschluss über den Erfolg bzw. Misserfolg der Operation gibt, zurück.

Das URL-Schema des Aufrufs sieht wie folgt aus:

`/fedoraAnnotations/delete/pid/`*pid*

Wobei der Parameter *pid* die Fedora-PID des zu löschenden Annotationsobjekts angibt.

Beispiele:

`/fedoraAnnotations/delete/pid/annotation:101`
Löscht das Annotationsobjekt mit der Fedora-PID `annotation:101`.

4.1.6 Die Konfigurationsdatei des Servers

Der Server besitzt eine einzige Konfigurationsdatei, in dem alle nötigen Parameter für den laufenden Betrieb hinterlegt sind. Die Datei wird nur einmal geladen, wenn der Servlet-Container den Annotations-Server startet.

Die Konfigurationsdatei findet sich unter `web/WEB-INF/fasConfig.xml` des Servlet Verzeichnisses vom Annotations-Server. Nachfolgend ein Auszug aus der Konfiguration, wobei aus Gründen der Übersichtlichkeit auf die Definitionen der vorhandenen Transformationen verzichtet wurde.

```
<config>
  <!-- fedora-admin user & password -->
  <fedoraUsr>fedoraAdmin</fedoraUsr>
  <fedoraPwd>fedoraAdmin</fedoraPwd>

  <!-- URLs for the Fedora REST-Interfaces (API-A and RISearch) -->
  <fedoraGetDataURL>http://localhost:8080/fedora/get/</fedoraGetDataURL>
  <fedoraRISearchURL>http://localhost:8080/fedora/risearch</fedoraRISearchURL>

  <!-- fedora repository namespace for annotation-objects.
```

```
        the namespace MUST exist in the repository! -->
    <fedoraAnnotationRepoNamespace>annotation</fedoraAnnotationRepoNamespace>

    <!-- datastream-name used in the annotation-objects to store its data -->
    <fedoraAnnotationDatastream>/DS</fedoraAnnotationDatastream>

    <!-- directory for the templates used by the servlet
        (i.e. to generate iTQL-query and FOXML-data for a new object) -->
    <templateDir>WEB-INF/tmpl/</templateDir>


    <!-- xslt-transformation for the result-set of getAnnotations().
        Hint: do not create a transformer with the name "xml". this name is
          reserved!! -->
    <transform>
        ...
    </transform>
</config>
```

4.1.7 Das Definition Servlet

Sämtliche XML-Schemas und XSLT-Stylesheets sind am Server hinterlegt. Die Verwaltung dieser Dateien obliegt einem eigenen Servlet, das Bestandteil des Annotations-Server Pakets ist. Das Servlet ist unter der URL

`/fedoraAnnotations/Definition/`

erreichbar. Ohne weitere Angabe liefert dieser Aufruf eine einfache HTML Seite, auf der alle verfügbaren Dateien aufgelistet und verlinkt sind. Ähnlich wie bei Fedora kann optional der Parameter `?xml=true` an der URL angehängt werden. In diesem Fall liefert das Servlet die selbe Liste in einem einfachen XML-Format.

Wird in der URL der Name einer bestehenden Datei mit angegeben, so wird dessen Inhalt retourniert. Beispielsweise liefert der Aufruf
`/fedoraAnnotations/Definition/FedoraAnnotation2Annotea.xsl`
den Inhalt jenes Stylesheets, das für die Transformation der Ergebnisliste des internen XML-Formats nach Annotea RDF/XML zuständig ist.

Das Servlet ist so programmiert, dass es nur jene Dateinamen als Parameter akzeptiert, die auch tatsächlich vorhanden sind. Zu diesem Zweck scannt das Servlet, wenn es vom Servlet Container gestartet wird, ein wohldefiniertes Verzeichnis nach allen Dateien ab. Dieses Verzeichnis befindet sich im Unterverzeichnis `web/WEB-INF/definitions/`. Dementsprechend müssen vom Administrator alle zusätzlichen Dateien (wie zum Beispiel neue XSLT-Stylesheets zum transformieren der internen XML Liste) in genau diesem Verzeichnis gespeichert werden. Da das Servlet nur während seines Starts das Verzeichnis untersucht, muss es neu gestartet werden, wenn die neuen Dateien verfügbar sein sollen.

4.2 Fedora-Annotation-Service – Clients

Im Rahmen dieser Arbeit wurden exemplarisch zwei Clienten für Bilder bzw. Videos implementiert, um die Funktionstüchtigkeit des Gesamtsystems zu zeigen. Wie bereits in Abschnitt 3.2 diskutiert, wird für jeden Medientypen ein eigener Client benötigt. Der Grund dafür ist, dass jeder Medientyp anders behandelt werden muss. Einen Text zu annotieren funktioniert grundsätzlich anders als ein Video zu annotieren. Daher müssen vor allem die Benutzeroberflächen der einzelnen Clienten an den jeweiligen Medientypen, auf dem sie spezialisiert sind, angepasst sein. Ein Client für Text muss Text darstellen und Teilbereiche (z.B Absätze) daraus markieren können. Ein Client für Video muss in der Lage sein die (heute gängigen) Videos abzuspielen, sowie Bereiche im Videobild und auf der Zeitachse zu markieren.

Die beiden Clienten, die im Zuge der Implementierung für das Framework entworfen wurden, können mit Bildern bzw. mit Videos umgehen und diese annotieren. Die Clienten wurden dabei für die Verwendung im Web konzipiert. Die Endbenutzer sollen die Möglichkeit haben das Framework ohne zusätzliche Installation von Clientprogrammen nutzen zu können. Diese Forderung impliziert, dass die verwendete Technologie für die Clienten direkt in einem beliebigen Webbrowser funktionieren muss. Dabei fiel die Entscheidung für die verwendete Technologie auf das aktuelle Flash-Plugin[5], das in der Version 9 vorliegt. Dieses Plugin ist für alle gängigen Webbrowser und Betriebssysteme[6] verfügbar.

Eine Alternative zum Flash-Plugin wäre eine Implementierung in Javascript gewesen. Diese Variante wurde aber aus mehreren Gründen verworfen. Um die Entscheidung zu Gunsten von Flash nachvollziehbar zu machen, soll hier eine kurze Auflistung der Vorteile von Flash9 gegeben werden:

- Das Flash-Plugin ist für nahezu jeden Webbrowser und jedes Betriebssystem vorhanden.
- Das Flash-Programm bildet eine eigene Kapsel (genannt Sandbox) auf der Webseite. Es ist daher nicht nötig, in den DOM-Baum des umschließenden HTML-Dokuments einzugreifen, um ein dynamisches User-Interface zu gestalten.
- Das Flash-Plugin funktioniert überall gleich. Auf jedem Browser, auf jedem Betriebssystem. Dies ist bei Javascript nach wie vor nicht der Fall.

[5] http://www.adobe.com/products/flash

[6] Adobe, der Hersteller von Flash, arbeitet seit geraumer Zeit an der Portierung des Flash-Plugins für Handys. In naher Zukunft wird es daher wahrscheinlich auch möglich sein, Flash auf mobilen Endgeräten zu nutzen.

- Flash verfügt über eine eigene Programmiersprache: Actionscript. In der aktuellen Version von Flash (Version 9) liegt Actionscript in der Version 3 vor. Die Sprache wurde in dieser Version von Grund auf neu konzipiert und entspricht dem ECMAScript Standard [11]. War sie früher nur ein kleiner Teil von Flash, so stellt sie in der aktuellen Version ein Grundpfeiler der Gesamttechnologie dar. Actionscript3 ist durchgängig objektorientiert und bietet alle Features, die ein Entwickler von einer modernen Programmiersprache erwartet. Alle Funktionen von Flash lassen sich mit ihr programmieren. Dies geht sogar soweit, dass der Entwickler nicht mehr unbedingt die (kostenintensive) Flash-IDE[7] benötigt, denn selbst Graphiken lassen sich per Actionscript-Programm gestalten.

- Der Actionscript3 Compiler wurde von Adobe unter OpenSource gestellt. Er steht zum kostenlosen Download bereit.

- Das Flash-Plugin kann Videos abspielen, wobei das Video-Format allerdings auf das Flash-Video Format beschränkt ist [45] [4]. Aber fast alle heute gängigen Videos im Web bauen auf genau diesem Format auf (z.B. YouTube, Viddler), da es vor allem für diesen Anwendungszweck konzipiert wurde. Das Flash-Plugin kann die Videos dabei auf zwei unterschiedliche Arten laden. Die einfachere Variante lädt das Video über eine HTTP-GET Anfrage (progressiv-download). Die zweite Variante kann das Video über RMTP-Streaming von einem geeigneten Livestreaming-Server laden [48] [28].

- Innerhalb des Flash Programms ergeben sich mehr Möglichkeiten zum Gestalten des User Interfaces. Drag & Drop, Farben, Alpha-Transparenzen, Scrolling. Das sind alles Effekte, die nur sehr umständlich mit Javascript gelöst werden können, aber das Auge des Benutzers erfreuen. Und das wiederum ist wichtig für den Erfolg des Gesamtsystems.

- Sehr leichtes Einbinden des Plugins in den HTML-Code des Ursprungssystems. Es ist nur ein einziger HTML-Tag dazu notwendig. Und zwar genau dort, wo der Client im Dokument liegen soll. Es ist kein Prolog nötig, um diverse Javascript-Bibliotheken einzubinden. Der HTML-Code des Ursprungssystems muss somit nur minimal geändert werden, um einen Clienten in die Webseite einzubinden.

- Adobe hat neben der Flash-IDE eine zweite, professionelle IDE entwickelt. Diese richtet sich vor allem an professionelle Entwickler, wohingegen die Flash-IDE eher für Webdesigner und Graphiker konzi-

[7] IDE – Integrated Development Environment.

piert ist. Diese neue IDE basiert auf der Eclipse-Plattform[8] und ist für die Forschung und Lehre kostenlos zu beziehen. Die IDE trägt den Namen FlexBuilder[9] und ist aktuell in der Version 3 erhältlich. Flex bietet dem Entwickler noch zusätzliche Features, die in der Flash-IDE nicht vorhanden sind. Darunter fallen fertige User-Interface Komponenten, Datenkomponenten und ein ausgereifter Debugger.

4.2.1 Der Bildclient

Der Client für Bilder bildet alle Funktionalitäten des Servers ab. Er kann ein Bild aus dem Fedora-Repository und die dazugehörigen Annotationen vom Annotations-Server laden, wobei das geladene Bild stets auf eine, für den Clienten, optimale Größe von 320 x 240 Pixel skaliert wird. Auf der rechten Seite wird die Liste aller Annotationen angezeigt. Dabei wird der Autor, das Datum und der (notfalls gekürzte) Text von jeder Annotation dargestellt. Jede Annotation bekommt vom Clienten eine eindeutige Farbe zugeteilt, die auch für das jeweilige Fragment benutzt wird. Die Fragmente werden direkt an der richtigen Stelle und in der richtigen Größe über das Bild gelegt und sind, damit der darunterliegende Bildausschnitt sichtbar bleibt, leicht transparent.

Durch einen Klick mit der Maus auf ein Fragment oder auf den Text einer Annotation wird diese fokussiert. Das Fragment dieser Annotation wird dadurch weniger transparent, alle anderen Annotationen werden aber noch durchsichtiger. Nur eine fokussierte Annotation kann vom Benutzer bearbeitet werden. Der Text oder das Bildfragment der Annotation kann geändert oder die gesamte Annotation kann gelöscht werden.

Die einzelnen Operationen des Clienten (neu, ändern, löschen) stehen dem Benutzer durch eine Toolbar zur Verfügung. Diese befindet sich direkt unterhalb des angezeigten Bildes, wobei nur jene Operationen anwählbar sind die auch erlaubt sind. Ändern und Löschen ist nur möglich, wenn eine Annotation den Fokus besitzt. Eine neue Annotation anlegen ist nur dann erlaubt, wenn keine der bereits vorhandenen Annotation den Fokus besitzt.

Unterhalb der Toolbar befindet sich ein Eingabefeld, das den Text jener Annotation anzeigt über der sich der Mauszeiger befindet bzw. jener Annotation, die gerade den Fokus besitzt. In diesem Feld kann auch der Text

[8] Die Eclipse-Plattform ist ein Grundgerüst für eine IDE, die zum entwickeln von Programmen gedacht ist. Ursprünglich wurde Eclipse für die Sprache Java entwickelt. Im Laufe der Zeit wurde aus der reinen Java-IDE ein Allzweckwerkzeug, das mit Plugins erweitert werden kann.

[9] http://www.adobe.com/products/flex

eingegeben werden, wenn eine Annotation neu angelegt oder geändert werden soll. An seinem untersten Rand zeigt der Client aktuelle Statusinformationen an. Darunter fallen unter anderem der Erfolg bzw. Misserfolg einer Speicher- oder Löschoperation.

Die Abbildung 4.1(a) auf Seite 75 zeigt den Bildclienten, wobei keine Annotation den Fokus besitzt. Allerdings befindet sich der Mauszeiger über der ersten Annotation (mit der Farbe türkis). Deswegen wird der Text dieser Annotation im Eingabefeld angezeigt und das Fragment wird etwas heller als normal dargestellt. Abbildung 4.1(b) zeigt nochmals den Clienten, wobei die dritte Annotation (mit der Farbe rot) fokussiert ist. Diese Annotation kann vom Benutzer geändert oder gelöscht werden. Darum sind die entsprechenden Buttons der Toolbar für den Benutzer auch anwählbar. In der Statuszeile wird zusätzlich angezeigt, das die dritte Annotation den Fokus besitzt ("Annotation 3 selected.").

4.2.2 Der Videoclient

Der Videoclient ist konzeptionell stark an den Bildclienten angelehnt - beide haben aus Sicht des Benutzers das gleiche Verhalten (Fokus, Toolbar-Buttons).

Er kann ein Flash-Video[10] über HTTP-GET vom Fedora-Repository laden (progressiv download), wobei das geladene Video stets auf 320 x 240 Pixel[11] skaliert wird.

Unterhalb des Videobilds befindet sich der Playhead. Ein kleines Dreieck gibt die zeitliche Position des Videos wieder und es kann vom Benutzer verschoben wurden, um das Video vor- bzw. zurückzuspulen. Dieses Verhalten ist den Benutzern von anderen Videoplayern im Web (z.B. der Player von der YouTube) bereits bekannt. Dem Playhead werden die Fragmente, die die Zeit des Videos betreffen, überlagert. Auch hier wird jenes Fragment, dessen Annotation den Fokus hat, für den Benutzer sichtbarer dargestellt.

Die Toolbar des Videoclienten ist um die Funktionalitäten eines Videoplayers erweitert. Zusätzlich zu den bereits vom Bildclienten bekannten Buttons gibt es drei weitere. Ein Button um das Video zu starten bzw. zu stoppen (Play/Pause) und ein Button um den Ton des Videos ein- bzw. auszuschalten (Mute on/off). Ein weiterer Button beeinflusst das Verhalten

[10] Das Flash-Video-Format kann vom Flash-Plugin ab der Version 6 verarbeitet werden, wobei die folgenden Video-Codecs unterstützt werden: Sorenson, VP6 und MPEG–4 (nach dem H.264 Standard). [45] [4]

[11] 320 x 240 Pixel entspricht der PAL Auflösung.

der Bildfragmente des Videos. Standardmäßig werden nur jene Bildfragmente gezeigt, bei denen sich der Playhead innerhalb des Zeitfragments befindet. So wird während des Abspielens des Videos die Aufmerksamkeit des Benutzers vor allem auf die gerade aktuellen Annotationen gelenkt. Dieses Verhalten kann ein- bzw. ausgeschaltet werden.

Die Abbildung 4.2(a) auf Seite 76 zeigt den Videoclienten, wobei keine Annotation fokussiert ist. Es ist nur das Bildfragment der zweiten Annotation (mit der Farbe gelb) zu sehen, da sich der Playhead des Videos nur über dessen Zeitfragment befindet. Die Aufmerksamkeit des Benutzers wird so auf diese zeitlich aktuelle Annotation gelenkt. In Abbildung 4.2(b) hat die dritte Annotation (mit der Farbe rot) den Fokus erhalten. Das Video "springt" dadurch an den zeitlichen Beginn der Annotation und sowohl das Zeitfragment als auch das Bildfragment sind visuell hervorgehoben.

4.2.3 Das Flash9-Plugin (swf-Datei)

Die beiden Clienten (Bild und Video) sind in einer einzigen Flash swf-Datei zusammengefasst. Die Clienten müssen vom Ursprungssystem eingebunden werden, damit sie dessen User-Interfaces um die Annotations-Features erweitern.

Die HTML Ausgabe des Ursprungssystem muss dahin geändert werden, dass es anstelle des Medieninhalts folgenden HTML-Code erzeugt:

```
...
<!-- ** ANNOTATION-CLIENT start ** -->
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
  id="AnnotationClient"
  width="504"
  height="375"
  codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.
    cab">
  <param name="bgcolor" value="#dddddd" />
  <param name="allowScriptAccess" value="sameDomain" />
  <param name="movie" value="AnnotationClient.swf" />
  <param name="flashVars" value="%%FLASH_VARS%%" />

  <embed type="application/x-shockwave-flash"
    name="AnnotationClient"
    width="504"
    height="375"
    pluginspage="http://www.adobe.com/go/getflashplayer"
    bgcolor="#dddddd"
    allowScriptAccess="sameDomain"
    src="AnnotationClient.swf"
    flashVars="%%FLASH_VARS%%" >
  </embed>
</object>
<!-- ** ANNOTATION-CLIENT end ** -->
...
```

Wobei `%%FLASH_VARS%%` die Parameter für das Flash Programm angeben. Folgende Parameter werden von dem Flash9-Plugin ausgewertet:

pid	Fedora PID des annotierten Medienobjekts.
ds	Datenstrom Name des Medieninhalts.
mime	MIME-Type des Medieninhalt. Bestimmt auch den zu ladenden Clienten (Bild oder Video).
fedora	URL zur API-A des Fedora-Repositories.
fas	URL zum Server des Fedora-Annotation-Services.

In dem folgenden Beispiel wird der Bildclient für den Datenstrom `DS1` des Medienobjekts `content:100` geladen.

```
pid=content:100&ds=DS1&mime=image/jpeg&fedora=http://localhost:8080/fedora/get&
  fas=http://localhost:8080/fedoraAnnotations
```

Hier wird der Videoclient für das Medienobjekt `content:432` geladen.

```
pid=content:432&ds=DS1&mime=video/x-flv&fedora=http://localhost:8080/fedora/get&
  fas=http://localhost:8080/fedoraAnnotations
```

4.3 Vergleich des Fedora-Annotation-Services mit aktuellen Annotationstools

An dieser Stelle wird das Fedora Annotation Framework ('FAS' in den Tabellen genannt) mit aktuellen Bild- bzw. Video-Annotationstools verglichen, um so die Praxistauglichkeit des Frameworks in Allgemeinen und der beiden Clienten für Bild und Video im Besonderen besser abschätzen zu können. Die Tools, die zum Vergleich herangezogen wurden, sind jene, die bereits in Abschnitt 2.2 vorgestellt wurden. Auch hier wird, genauso wie in Abschnitt 2.2.4, der Vergleich tabellarisch dargestellt.

Legende zu den Tabellen 4.1 und 4.2
⊕ Feature wird voll unterstützt
⊘ Feature wird teilweise unterstützt
⊖ Feature wird nicht unterstützt

Bild Annotationstools

	Flickr	Fotonotes	TelPlus	FAS
Suche	⊖	⊖	⊕	⊘
Zugriffskontrolle	⊕	⊖	⊖	⊘
Diskusionen in Annotationen	⊖	⊖	⊖	⊖
Farbwahl einer Annotationen	⊖	⊖	⊕	⊘
Bildbereich markieren	⊕	⊕	⊕	⊕

Tabelle 4.1: *Vergleich der Bild Annotationstools mit dem Fedora Annotation Service*

Die einzelnen Annotationstools wurden bereits in Abschnitt 2.2.2 und 2.2.3 genauer beschrieben. Sowohl der Bild- als auch der Videoclient des Fedora-Annotation-Services (FAS) sind in der Lage Teilbereiche des Bildes zu selektieren – beide sind dabei auf Rechtecke beschränkt. Der Videoclient kann

Video Annotationstools

	YouTube	Viddler	BubblePLY	FAS
Suche	⊖	⊖	⊖	⊘
Zugriffskontrolle	⊘	⊖	⊖	⊘
Diskusionen in Annotationen	⊖	⊕	⊖	⊖
Farbwahl einer Annotationen	⊖	⊖	⊕	⊘
Bildbereich markieren	⊘	⊖	⊘	⊕
Zeitbereich markieren	⊘	⊘	⊕	⊕

Tabelle 4.2: *Vergleich der Video Annotationstools mit dem Fedora Annotation Service*

zusätzlich die Zeitachse des Videos markieren und diese Zeitspanne wird auch im Playhead angezeigt. Er ist somit der einzige hier vorgestellte Client, der alle drei Dimensionen (Bild und Zeit) eines Videos gleichberechtigt behandelt. Zwar wurden im Zuge dieser Arbeit weder eine explizite Suche nach Annotationen noch eine Zugriffskontrolle implementiert, das Fedora-Repository bietet jedoch diese Features an. Eine Erweiterung des Frameworks dahingehend ist daher prinzipiell möglich (vgl. Abschnitt 6.1).

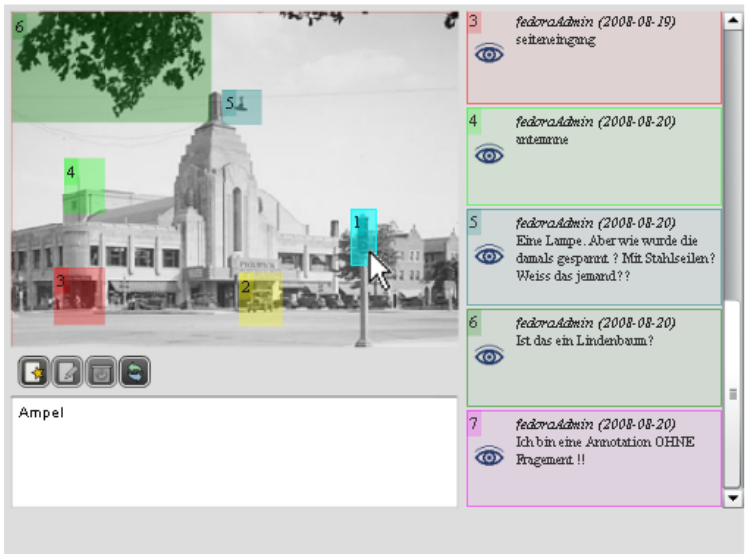
4.4 Zusammenfassung

In diesem Kapitel wurden die Details der Implementierung des Fedora-Annotation-Services beschrieben. Der Server wurde in Java programmiert und stellt seine Funktionalitäten über ein einfaches REST-Interface den Clienten zur Verfügung. Ein HTTP-GET Aufruf liefert immer eine Liste von Annotationen. Diese Liste kann entweder im internen XML-Format oder in einem anderen Format angefordert werden, wobei der Server die interne Liste mittels einem XSLT-Stylesheet und dem Fedora Saxon Web-Service transformiert bevor er das Ergebnis an den Aufrufer sendet. Mit einem HTTP-POST Aufruf kann eine Annotation neu angelegt, eine bestehende geändert oder gelöscht werden. Die Auswahl der gewünschten Operation erfolgt dabei über ein einfaches URL-Schema.

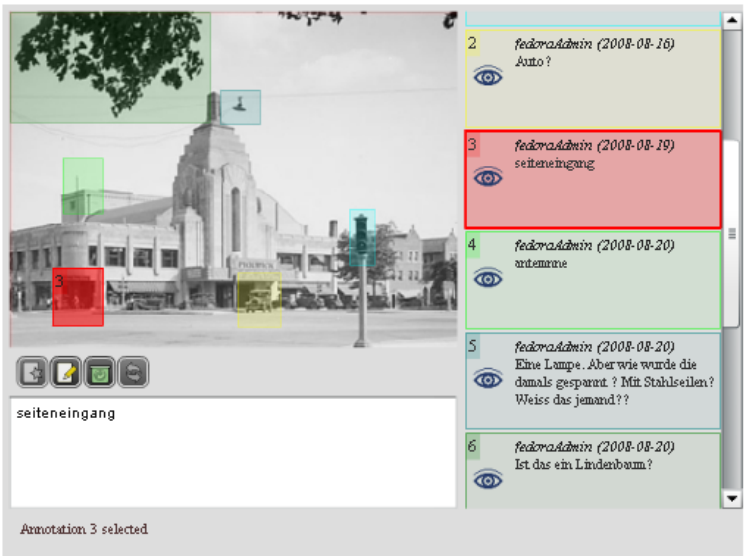
Die beiden implementierten Clienten (Bild und Video) wurden mit dem FlexBuilder für das Flash9-Webbrowser Plugin programmiert. Dadurch ist es möglich, dass die Clienten auf nahezu allen Webbrowsern und Betriebssystemen lauffähig sind und überall das exakt gleiche Verhalten und Aussehen haben. Beide Clienten beziehen ihre Mediendaten direkt vom Fedora-Repository und die Annotationsdaten vom Annotations-Server (im internen XML-Format).

Den Abschluss des Kapitels bildete ein nochmaliger tabellarischer Vergleich der in Abschnitt 2.2.4 untersuchten Bild- bzw. Video-Annotationstools mit dem Fedora-Annotation-Service. Die Praxistauglichkeit des Frameworks im

Allgemeinen und der beiden Clienten im Besonderen soll so übersichtlich gezeigt werden.

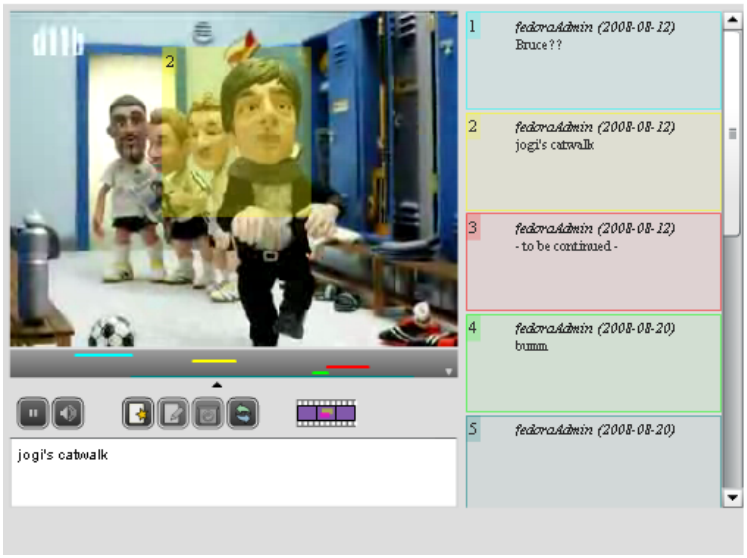


(a)

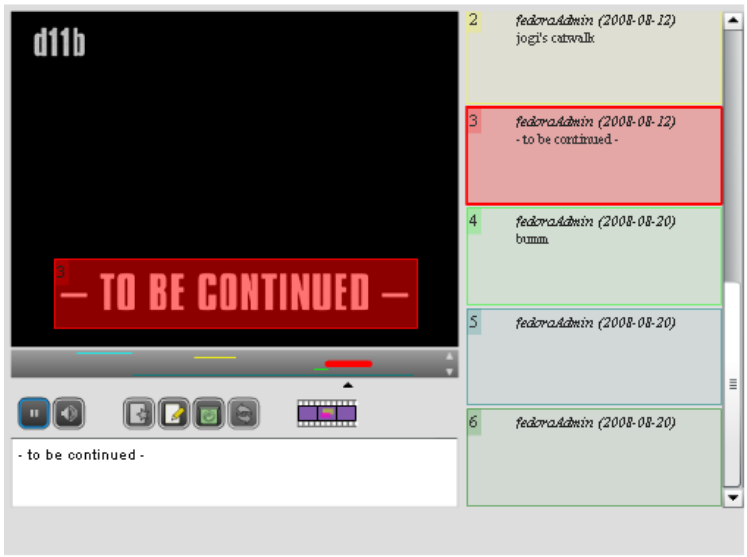


(b)

Abbildung 4.1: *Der Bildclient. In Abbildung (a) ist keine Annotation fokussiert. Nur der Mauszeiger liegt über dem Fragment der ersten Annotation (Farbe türkis). Der Text dieser Annotation wird im Eingabefeld angezeigt, das dazugehörige Fragment wird heller als die anderen dargestellt. In Abbildung (b) wurde die dritte Annotation (Farbe rot) mit der Maus angeklickt und daher fokussiert. Die Fragmente der anderen Annotationen sind noch transparenter als normal, damit der Benutzer sich auf die fokussierte Annotation konzentriert. In der Toolbar sind nur die Buttons zum ändern und löschen aktiv, da diese Operationen nur bei einer fokussierte Annotationen erlaubt sind.*



(a)



(b)

Abbildung 4.2: *Der Videoclient. In Abbildung (a) befindet sich der Playhead des Videos nur über der zweiten Annotation (Farbe gelb), daher ist nur dessen Bildfragment zu sehen. In Abbildung (b) besitzt die dritte Annotation (Farbe rot) den Fokus, weshalb dessen Bild- und Zeitfragment zu sehen und hervorgehoben sind. Zusätzlich "springt" der Videoplayer an den Beginn des Zeitfragments, wenn eine Annotation den Fokus erhält. Das restliche Verhalten des Videoclienten ist das selbe wie das des Bildclienten.*

Kapitel 5

Fallstudie

In diesem Kapitel wird ein typisches Anwendungsszenario für das Fedora-Annotation-Service vorgestellt. Das Framework soll dabei in ein bereits bestehende System integriert werden, damit dieses in Zukunft seinen Benutzern die Möglichkeiten bietet, die eigenen Medieninhalte zu annotieren. Es wird davon ausgegangen, dass das Ursprungssystem das Fedora-Repository als Backend einsetzt.

Es wird jeder Schritt der Integration im Detail besprochen und gezeigt, wie der Annotations-Server das Fedora-Repository des Ursprungssystems nutzt. Das Ursprungssystem muss anstelle seines eigenen Medieninhalts (z.B. eines Bildes) den entsprechenden Annotations-Clienten (z.B. Bildclient) an seinen Clienten (z.B. Webbrowser) schicken. Daher muss die Ausgabe des Ursprungssystems leicht geändert werden. Durch den Einsatz des Flash-Plugins ist diese Änderung jedoch minimal. Um an der Stelle eines Bildes im HTML-Code den Bildclienten zu laden reicht es aus, dass ursprüngliche Image-Tag (`<img>`) gegen ein Object- und/oder Embed-Tag (`<object>`, `<embed>`) zu tauschen.

Damit die folgenden Beschreibungen leichter nachvollziehbar sind, wird ein real existierendes System als Ursprungssystem angenommen. Dieses System ist eine öffentlich zugängliche Digital Library im Web. Konkret handelt es sich um die *Encyclopedia of Chicago*[1], die auch in der Realität das Fedora-Repository als Backend nutzt. In der Encyclopedia of Chicago werden historische Bilder, Pläne und Texte der Stadt Chicago gesammelt und veröffentlicht. Die Benutzer dieses Systems haben derzeit aber nicht die Möglichkeit aktiv am Inhalt der Library zu partizipieren. Dies soll durch die Hinzunahme von Annotationsfeatures geändert werden, wobei zum aktuellen Zeitpunkt

[1] http://www.encyclopedia.chicagohistory.org

nicht klar ist, ob später weitere Medientypen (z.B. Videos von Ansprachen) für die Benutzer zur Verfügung gestellt werden.

Abbildung 5.1 zeigt ein Bild des *Pickwick Theater*. Der Screenshot wurde dabei der Encyclopedia of Chicago entnommen und ist für jeden Benutzer unter der URL
`http://www.encyclopedia.chicagohistory.org/pages/10436.html`
abrufbar. Alle weiteren Beschreibungen zur Integration des Fedora-Annotation-Services in das Ursprungssystem beziehen sich auf dieses Bild bzw. auf den Inhalt dieser Webseite.

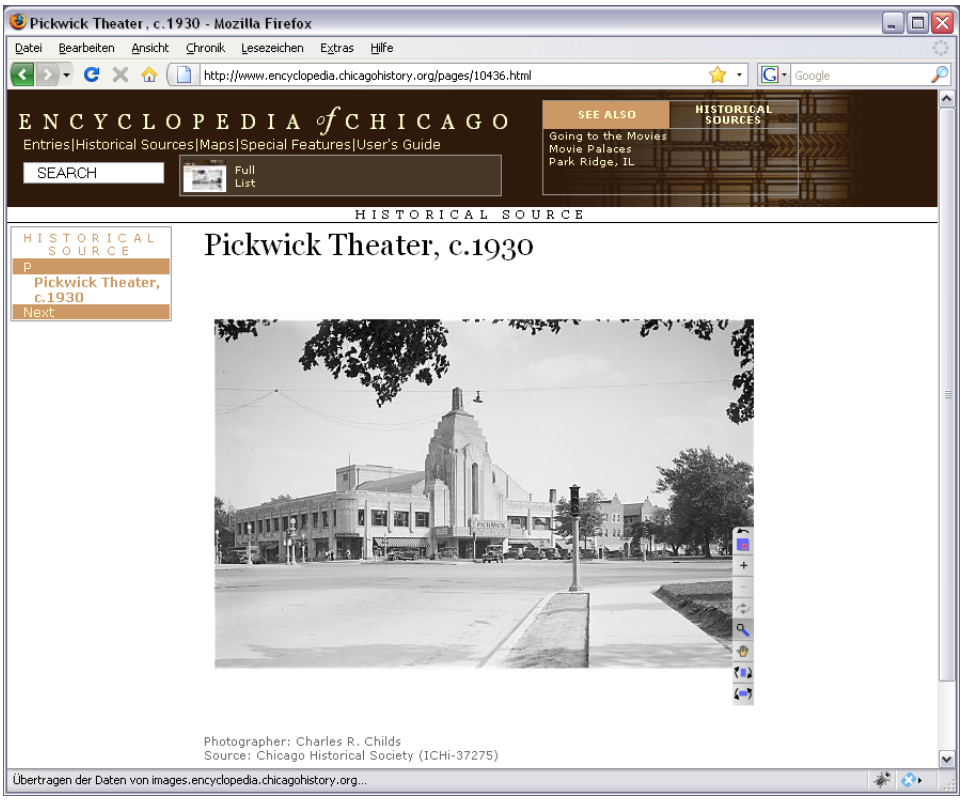


Abbildung 5.1: *Das Pickwick Theater. Dieser Screenshot wurde der original Encyclopedia of Chicago entnommen. Alle weiteren Angaben in dieser Fallstudie beziehen sich auf dieses Bild bzw. diese Webseite.*

5.1 Integration des Server

Das Ursprungssystem läuft bereits als Web-Applikation und benutzt das Fedora-Repository als sein Backend. Das bedeutet, dass bereits jeder einzelne Medieninhalt (Bild, Text, etc.) im Fedora-Repository als eigenes Digitales Objekt gespeichert ist. Die Benutzer rufen dabei die Medieninhalte nicht direkt vom Repository ab, sondern bekommen die Daten von der Web-Applikation zur Verfügung gestellt. Der Annotations-Server wiederum kann diese Digitalen Objekte bereits benutzen, um seine semantischen Relationen zwischen seinen Annotationsobjekten und den vorhandenen Medienobjekten herzustellen.

Abbildung 5.2 zeigt schematisch den technischen Aufbau der Encyclopedia of Chicago. Das Fedora-Repository stellt das Backend dar, in dem die einzelnen Medieninhalte, wie zum Beispiel Bilder oder Texte, gespeichert sind. Möglicherweise nutzt auch die Encyclopedia of Chicago semantische Relationen um zwischen einzelnen Digitalen Objekten eine Verbindung herzustellen. So ist es durchaus denkbar, dass ein Objekt als Container für einzelne Bilder dient, wie zum Beispiel im Fall von Scans einzelner Buchseiten. Der Container übernimmt dann die Aufgabe eines digitalen "Buchrückens", der die einzelnen digitalen Seiten zusammenhält. Das Digitale Objekt `content:1` stellt einen solchen Buchrücken dar, der die beiden digitalen Seiten `content:63` und `content:3` mit der semantische Relation `hasMember` zusammenhält.

Um den Server in das Ursprungssystem zu integrieren sind drei Schritte nötig:

1. Der Annotations-Server muss Teil der Web-Applikation werden.
   Dazu muss die Java `war`-Datei, welche die beiden Servlets beinhaltet, im Servlet-Container[2] installiert werden. Üblicherweise reicht es die `war`-Datei in das Servlet-Verzeichnis zu kopieren und den Servlet-Container neu zu starten. Darauf hin wird die neue Web-Applikation (in diesem Fall der Annotations-Server) installiert und registriert.

2. Der Annotations-Server muss konfiguriert werden.
   Die Konfigurationsdatei muss dahingehend geändert werden, dass der Annotations-Server das Fedora-Repository des Ursprungssystems nutzt. Dabei reicht es die URL und die Logindaten richtig zu stellen.

3. Das Fedora-Repository muss für den Betrieb des Frameworks konfiguriert werden.
   Der Namespace `annotation` muss dem Repository hinzugefügt und

[2]Der Einfachheit halber wird hier angenommen, dass die Web-Applikation der Encyclopedia of Chicago selbst als Java Servlets ausgeführt ist.

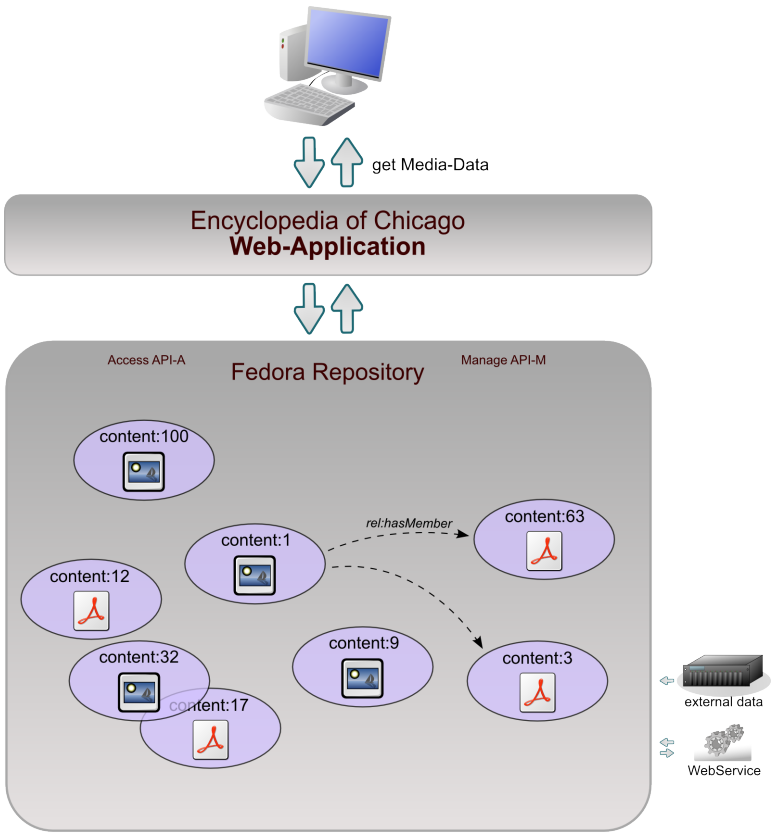


Abbildung 5.2: *Übersicht über den technischen Aufbau der Encyclopedia of Chicago. Das Fedora-Repository wird als Backend verwendet, um alle Medieninhalte zu speichern. Die Benutzer rufen nicht das Fedora-Repository direkt auf, sondern kommunizieren über eine Web-Applikation, die ihrerseits die Daten vom Repository bezieht.*

der Ressource Index muss aktiviert werden. Beides geschieht in der Konfigurationsdatei von Fedora und wird in Abschnitt A.1.4 genauer beschrieben.

Damit ist die Integration des Server abgeschlossen und der Annotations-Server residiert nun parallel und unabhängig zur Web-Applikation des Ursprungssystems. Abbildung zeigt 5.3 den Aufbau nach erfolgter Integration des Servers.

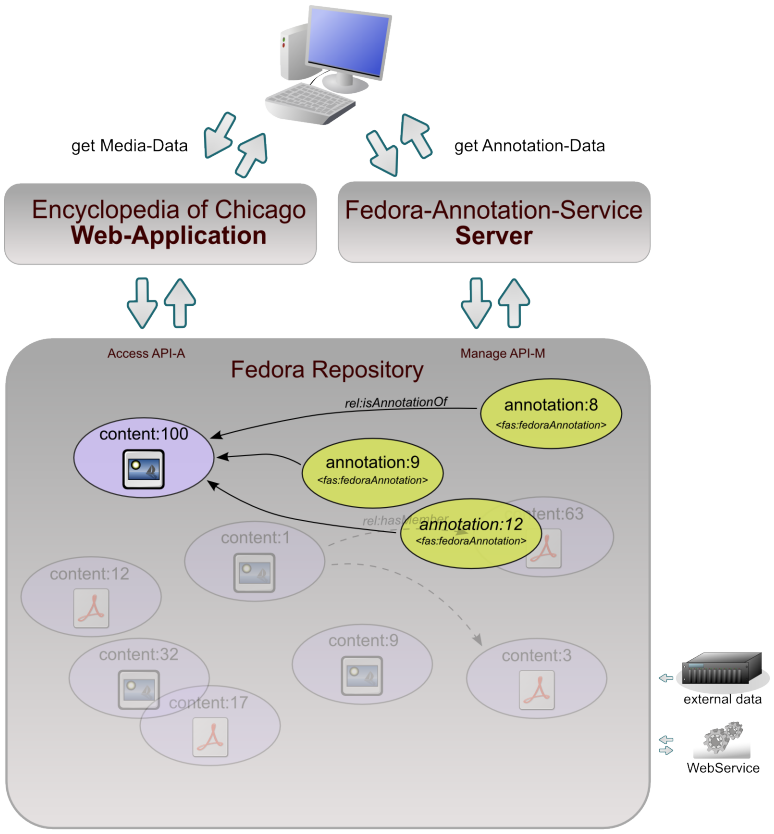


Abbildung 5.3: *Das Ursprungssystem mit dem integriertem Annotations-Server. Der Server läuft parallel zu dem Ursprungssystems, greift aber auf das gleiche Fedora-Repository zu.*

5.2 Integration des Clienten

Beide Flash Clienten besteht aus einer einzigen Datei, die jeweils im Flash `swf`-Format vorliegt. Um den, in diesem Beispiel, Bildclienten in das Ursprungssystem zu integrieren, muss die HTML-Ausgabe des Ursprungssy-

stems leicht angepasst werden. Üblicherweise wird ein `<img>` Tag benutzt, um ein Bild auf einer Webseite anzuzeigen[3]. Dieser `<img>` Tag muss gegen ein `<object>` und/oder einen `<embed>` Tag getauscht werden, damit anstelle des einfachen Bildes der Bildclient vom Webbrowser geladen wird.

Der `<object>` Tag wird vom MS Internet Explorer ausgewertet. Alle anderen Browser (z.B. Firefox, Opera) verwenden den `<embed>` Tag. Die hier gezeigt Kombination aus diesen beiden Tags ermöglicht es ohne Einsatz von Javascript das alle gängigen Webbrowser ein Flash-Plugin korrekt einbinden. Der Client muss vom Server nur noch mitgeteilt bekommen, woher er seine Daten beziehen soll – dies geschieht über die FlashVars. Diese werden einfach innerhalb der Tags zum Einbinden des Flash-Programms angegeben.

Nachfolgend wird der entscheidende Auszug aus dem originalen HTML-Code mit dem `<img>` Tag gezeigt, der das Bild vom Pickwick Theater anzeigt, wobei dieses Bild die Fedora-PID `content:100` hat.

```
  ...
<p>
  <!-- ** CONTENT start ** -->
  <img src="http://localhost:8080/fedora/get/content:100/DS1">
  <!-- ** CONTENT end   ** -->
</p>
  ...
```

Der originale HTML-Code, der mittels `<img>` Tag das Bild vom Pickwick Theater lädt.

Hier der geänderte HTML-Code, der den Bildclienten lädt. In den FlashVars werden dem Clienten alle benötigten Parameter angegeben, damit er das Bild und die dazugehörigen Annotationsdaten laden kann. Die Parameter geben dabei die jeweilige URL zum Repository und zum Annotations-Server sowie die PID des Medienobjekts an.

[3]Tatsächlich benutzt die Encyclopedia of Chicago kein `<img>` Tag für die Bilder, sondern greift ebenfalls auf ein Flash Programm zurück (siehe Abbildung 5.1). Dieses Flash-Plugin ist u.a. in der Lage das angezeigt Bild zu skalieren (Zoom). Dazu bedient es sich der im Repository gespeicherten unterschiedlichen Auflösungen des Bildes. Diese Vorgehensweise wurde bereits in Abschnitt 2.3.2 beschrieben und in Abbildung 2.15 an einem Beispiel gezeigt. In diesem Fall wird die Integration des Annotationsclienten sogar noch weiter vereinfacht.

```
  ...
<p>
  <!-- ** ANNOTATION-CLIENT start ** -->
  <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    id="AnnotationClient"
    width="504"
    height="375"
    codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.
      cab">
    <param name="bgcolor" value="#dddddd" />
    <param name="allowScriptAccess" value="sameDomain" />
    <param name="movie" value="AnnotationClient.swf" />
    <param name="flashVars" value="pid=content:100&ds=DS1&mime=image/jpeg&fedora=
      http://localhost:8080/fedora/get&fas=http://localhost:8080/
      fedoraAnnotations" />

    <embed type="application/x-shockwave-flash"
      name="AnnotationClient"
      width="504"
      height="375"
      pluginspage="http://www.adobe.com/go/getflashplayer"
      bgcolor="#dddddd"
      allowScriptAccess="sameDomain"
      src="AnnotationClient.swf"
      flashVars="pid=content:100&ds=DS1&mime=image/jpeg&fedora=http://localhost
        :8080/fedora/get&fas=http://localhost:8080/fedoraAnnotations"  >
    </embed>
  </object>
  <!-- ** ANNOTATION-CLIENT end ** -->
</p>
  ...
```

Der geänderte HTML-Code, wobei die Tags `<object>` *und* `<embed>` *den Annotations-Clienten für Bilder laden .*

Durch diesen HTML-Code wird der Bildclient auf der Webseite angezeigt. Dieser wiederum lädt das Bild vom Repository und die Annotationsdaten vom Annotations-Server. Abbildung 5.4 zeigt das Ergebnis, so wie es im Webbrowser aussieht. Zum Vergleich von diesem Screenshots siehe auch Abbildung 5.1, in dem das Original gezeigt wird.

5.3 Zusammenfassung

In diesem Kapitel wurde die Integration des Fedora-Annotation-Services in eine bestehende Digital Library gezeigt. Dabei wurden alle nötigen Schritte anhand eines konkreten, realen Beispiels – der Encyclopedia of Chicago – diskutiert. Der Annotations-Server arbeitet dabei parallel und unabhängig vom Ursprungssystem, greift aber auf dessen Fedora-Repository zurück. Um den Clienten auf der Webseite einzubinden, ist eine minimale Änderung der HTML-Ausgabe des Ursprungssystems notwendig. Diese Änderung betrifft aber in aller Regel nur einen einzigen HTML-Tag und zwar genau dort, wo der Client im Webbrowser angezeigt werden soll.

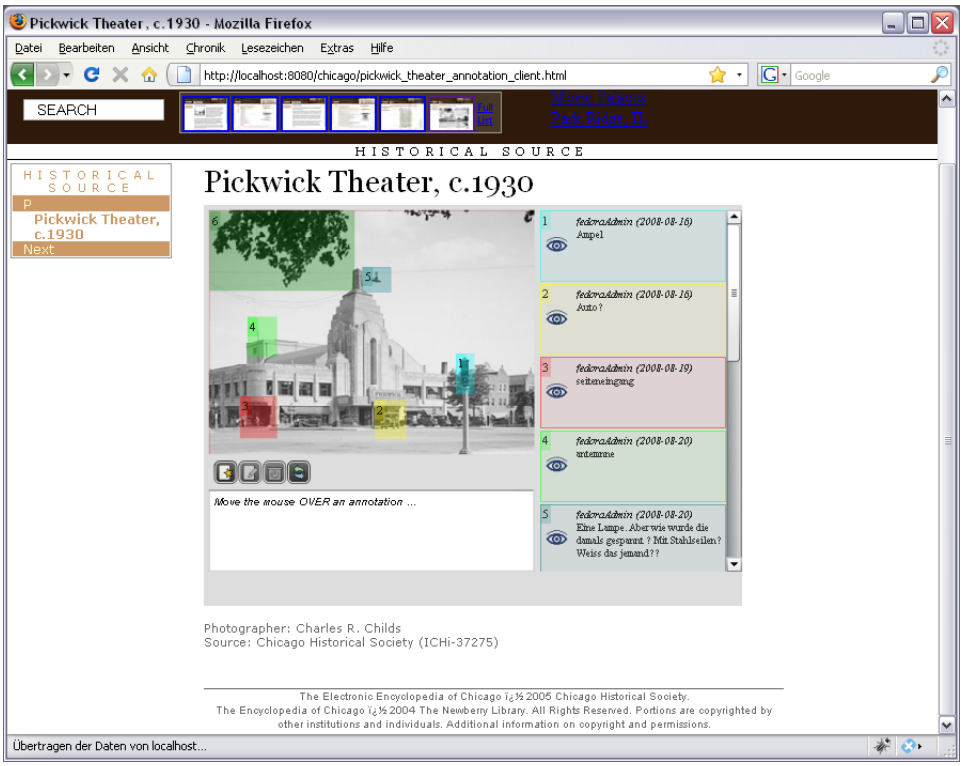


Abbildung 5.4: *Das Pickwick Theater im Bildclienten. Zum Vergleich siehe auch Abbildung 5.1, in dem das Original gezeigt wird.*

Kapitel 6

Mögliche Erweiterungen

Im Rahmen dieser Arbeit wurde ein Framework zur Annotation multimedialer Objekte, bestehend aus Metadaten, Server und Clienten vorgestellt. Verständlicherweise kann dieses Framework im Rahmen einer Diplomarbeit nicht alle Wünsche und Features abdecken.

Dieses Kapitel beschäftigt sich daher damit, welche Funktionalitäten nicht implementiert wurden, beziehungsweise welche zusätzlichen Features für ein Annotations-Framework, sei es nun für den Server oder für den Clienten, sinnvoll oder unbedingt erforderlich sind.

Abschnitt 6.1 erläutert mögliche Erweiterungen des Servers. In Abschnitt 6.2 werden sinnvolle Änderungen der Metadaten beschrieben und Abschnitt 6.3 zeigt schlussendlich neue oder geänderte Features für die beiden Clienten (Bild und Video) auf.

6.1 Server

6.1.1 REST Web-Service

Der Server bietet eine REST-Schnittstelle an. Dies ist jedoch nur zu einen Teil richtig: tatsächlich reagiert das Service nur auf HTTP-GET und HTTP-POST Anfragen. Dies entspricht nicht der eigentlichen Definition eines REST Web-Services [5].

Ist
Der Server liefert bei einer HTTP-GET Anfrage stets eine Resultatsliste. Sämtliche Datenänderungen werden mittels HTTP-POST angestoßen. Wie der Server die Daten ändern soll (neu anlegen, ändern oder löschen) wird

durch ein einfaches URL Schema aufgelöst. Der ausschlaggebende Grund für diese Herangehensweise war, dass das Flash-Plugin innerhalb eines Browsers nur diese beiden HTTP-Request versenden kann.

Soll
Gemäß der REST-Definition sollte der Server auch HTTP-PUT und HTTP-DELETE unterstützen. Daraus resultiert auch, dass in diesem Fall kein URL-Schema zum Auflösen der gewollten Operation nötig ist. Sinnvollerweise sollte der Server aber beide Varianten unterstützen, damit der Server trotzdem auch mit dem Flash-Plugin zusammenarbeiten kann. Aus diesen Forderungen ergibt sich folgendes:

1. Flash/Browser Schema
    - Geht Annotation-List
      HTTP-GET
      http://localhost:8080/fedoraAnnotations/pid/content:1
      Liefert eine Liste von Annotationen, die das Medien-Objekt *content:1* annotieren.
    - Get transformed Annotation-List (am Beispiel Annotea)
      HTTP-GET
      http://localhost:8080/fedoraAnnotations/pid/content:1/type/annotea
      Liefert eine Liste von Annotationen, die das Medien-Objekt *content:1* annotieren. Die Liste wird dabei im Annotea RDF/XML Format zurückgeliefert.
    - Create a new Annotation
      HTTP-POST
      http://localhost:8080/fedoraAnnotations/create/pid/content:1
      Erzeugt eine neue Annotation für das Medien-Objekt *content:1*.
    - Update Annotation-Data
      HTTP-POST
      http://localhost:8080/fedoraAnnotations/update/pid/annotation:12
      Ändert die Daten des Annotations-Objekts *annotation:12*.
    - Delete an Annotation
      HTTP-POST
      http://localhost:8080/fedoraAnnotations/delete/pid/annotation:12
      Löscht das Annotations-Objekt *annotation:12*.
2. REST Schema
    - Get Annotation-List
      HTTP-GET accept text/xml
      http://localhost:8080/fedoraAnnotations/pid/content:1
      Liefert eine Liste von Annotationen, die das Medien-Objekt *content:1* annotieren.

- Get transformed Annotation-List (am Beispiel Annotea)
  HTTP-GET accept text/rdf+xml
  http://localhost:8080/fedoraAnnotations/pid/content:1
  Liefert eine Liste von Annotationen, die das Medien-Objekt *content:1* annotieren. Die Liste wird dabei im Annotea RDF/XML Format zurückgeliefert.
- Create a new Annotation
  HTTP-POST
  http://localhost:8080/fedoraAnnotations/pid/content:1
  Erzeugt eine neue Annotation für das Medien-Objekt *content:1*.
- Update Annotation-Data
  HTTP-PUT
  http://localhost:8080/fedoraAnnotations/pid/annotation:12
  Ändert die Daten des Annotations-Objekts *annotation:12*.
- Delete an Annotation
  HTTP-DELETE
  http://localhost:8080/fedoraAnnotations/pid/annotation:12
  Löscht das Annotations-Objekt *annotation:12*.

6.1.2 XML Validierung

Der Server überprüft derzeit nicht die eingehenden XML-Daten. Dies ist für einen Produktionseinsatz selbstverständlich unbrauchbar. Der Server muss bei einem Create bzw. bei einem Update die empfangenen Daten gegen das Annotations-Schema validieren. Falsche oder fehlerhafte Daten können so erkannt und abgewiesen werden.

6.1.3 Rechtesystem

Derzeit verfügt der Server über kein Rechtesystem. Auch dies ist für den Produktionseinsatz nicht wünschenswert. Jedoch hat das Backend (das Fedora-Repository) ein ausgereiftes Rechtesystem auf XACML-Basis [37] [6]. Dies sollte aktiviert werden und der Server muss dieses auch nutzen.

Damit einhergehend sollte es für den einzelnen Benutzer möglich sein, seine eigenen Annotationen mit Zugriffsrechten zu versehen: soll die eigene Annotation für jeden lesbar sein? Darf nur der ursprüngliche Autor der Annotation diese wieder editieren? Das Fedora-Repository ist mit seinem Rechtesystem in der Lage solchen Anforderungen Gerecht zu werden. Der Annotations-Server muss auf die Rechte der Benutzer Rücksicht nehmen. Des weiteren brauchen die Clienten für diese Einstellungen ein eigenes User-Interface, damit die Benutzer die Rechte auch verwalten können.

6.1.4 Concurrency

Bei der Implementierung des Servers wurde gänzlich auf die Nebenläufigkeit bei Datenänderungen verzichtet. Aber gerade bei Web-Anwendungen kommt es immer wieder vor, dass mehrere Benutzer gleichzeitig an einem Dokument, respektive einer Annotation, arbeiten. Der Server muss diesem Umstand Rechnung tragen. Diese Forderung überschneidet sich teilweise mit der Forderung nach einem Rechtesystem: können mehrere unterschiedliche Benutzer tatsächlich ein und die selbe Annotation gleichzeitig bearbeiten?

6.1.5 Abspeichern von anderen Annotations-Standards

Der Server ist in der Lage seine Daten nicht nur im eigenen Format auszugeben. Auf Wunsch kann die Liste in ein anderes Format konvertiert werden bevor die Daten an den Clienten geschickt werden. So können im Prinzip verschiedene Clienten, die mit einem anderen Standard (wie zum Beispiel Annotea) operieren ebenso mit den Daten des Frameworks arbeiten.

Wünschenswert wäre es aber wenn die Daten nicht nur beim Lesen, sondern auch beim Schreiben transformiert werden könnten. Dadurch wäre es möglich, auch Annotea-Daten entgegenzunehmen, in das eigene, interne Format zu konvertieren und erst dann im Backend abzuspeichern. Dies würde die Interoperabilität weiter erhöhen.

Abbildung 6.1 zeigt einen möglichen Aufbau des Servers mit dieser Erweiterung. Die Transformationen der Eingabedaten funktioniert dabei analog zur Transformation der Ausgabedaten. Die Eingabedaten werden in einem Fremdformat (z.B. Annoteas RDF/XML Format) entgegengenommen, mittels geeigneten XML–Stylesheet in das interne XML Format konvertiert und erst dann im Fedora-Repository abgespeichert.

6.2 Metadaten

6.2.1 Redundanz der Daten

Ein Annotations-Objekt ist durch eine semantische Relation vom Typ `isAnnotationOf` im Backend mit dem Medien-Objekt verknüpft. Trotzdem wird zusätzlich innerhalb der Annotations-Daten ein Verweis auf das zu annotierende Objekt gespeichert. Dies stellt eine Redundanz dar, die nicht unbedingt notwendig ist.

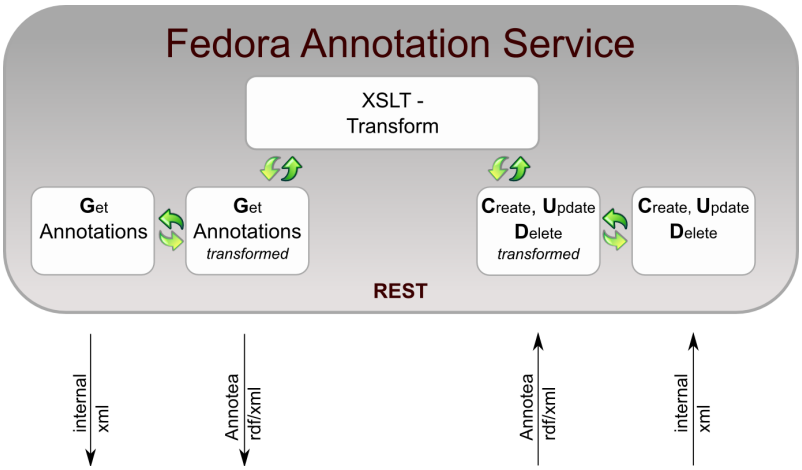


Abbildung 6.1: *Der Annotations–Server mit der Erweiterung zum Abspeichern von Annotea–Annotationen. Analog zur Ausgabe werden die Eingabedaten mittels XML-Stylesheet in die interne XML-Darstellung transformiert, bevor die Daten im Fedora–Repository abgespeichert werden.*

6.2.2 Farbgebung einer Annotation im Clienten

Im Clienten wird jeder Annotation eine spezielle Farbe zugewiesen. Diese Farbe wird vom Clienten selbst vergeben und kann sich von einem Aufruf zum nächsten ändern. Dies stellt insofern ein Problem dar, als dass sich die Benutzer durchaus über eine bereits existierende Annotation unterhalten können. Da eine Annotation eine Farbe hat, ist es leicht möglich, dass die Benutzer sich auf eben diese Farbe beziehen (z.B. '... wie in der roten Annotation zu sehen ...'). Solche, oder ähnliche Kommentare sind auf anderen System, wie zum Beispiel Flickr oder YouTube, durchaus häufig zu beobachten. Bei einer (mehr oder minder) zufällig vergebenen Farbe machen aber gerade solche Verweise auf andere Annotationen keinen Sinn. Denn sollte sich die Farbe einer Annotation ändern, stiftet dies nur unnötige Verwirrung bei den Benutzern.

Dem kann entgegengewirkt werden, indem die Farbe direkt in den Annotationsdaten mitgespeichert wird. So hat eine bestimmte Annotation immer und überall die gleiche Farbe. Die Benutzer können sich später auf diese Farbe in ihren Diskussionen beziehen.

6.2.3 XSD-Vererbung

Das XML-Schema der Annotations-Metadaten nutzt nur sehr rudimentär die Möglichkeiten der Vererbung. Nicht vorgesehen ist es, verschiedene Sche-

mas zu vereinen, um zum Beispiel externe Fragmentdefinitionen zu ermöglichen. Die gegenwärtige Herangehensweise hat zwar den Vorteil, dass die XML-Dokumente selbst einfacher gehalten sind, allerdings erschwert sie die nachträgliche Änderung oder Erweiterung des Schemas um externe Definitionen.

6.3 Client

6.3.1 Orientierung in der Annotationsliste

Die Clienten (sowohl für Bild als auch für Video) zeigen alle Annotationen in einer Liste als Übersicht auf der rechten Seite. Allerdings wird es für die Benutzer schwierig den Überblick zu wahren sobald die Anzahl der Annotationen eine kritische Masse erreicht. Schon eine einfache Suchfunktion könnte hier eine praktikable Abhilfe schaffen. Abbildung 6.2 zeigt den Video-Clienten mit der Suchfunktion oberhalb der Annotationsliste.

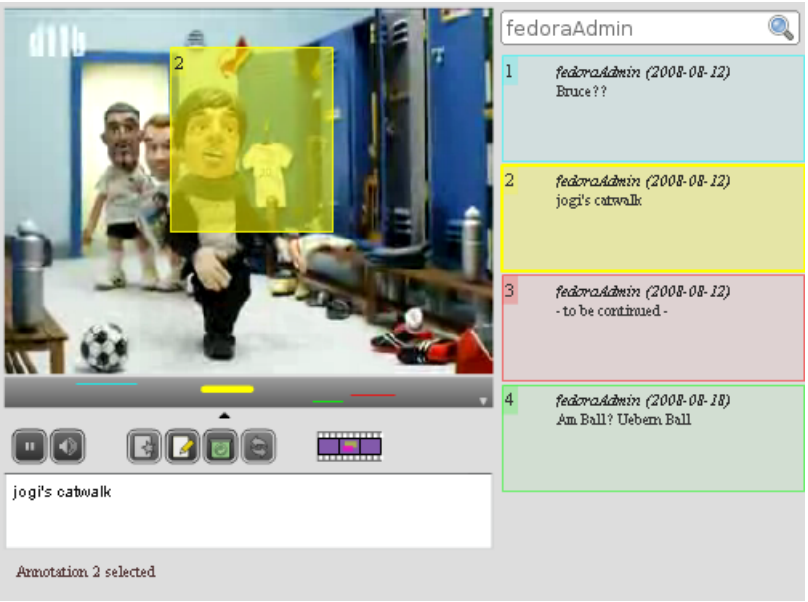


Abbildung 6.2: *Video-Client mit Suchfunktion. In diesem Beispiel zeigt der Client nur jene Annotationen an, in denen der Text 'fedoraAdmin' vorkommt.*

6.3.2 Zoom und Pan bei Bildern

Der Bild-Client stellt alle Bilder im Format 320x240 Pixel dar. Ist das Bild kleiner oder größer als dieses Format wird es automatisch auf diese Größe skaliert (vergrößert bzw. verkleinert). Aber gerade bei großen Bilder kann dies ein nicht zu unterschätzender Nachteil sein, da kleine Bildbereiche durch diesen Autozoom nicht mehr genau markiert werden können. Daher wäre eine manuelle Zoom & Pan Funktion unbedingt von Nöten.

6.3.3 Ungenaues Annotieren der Zeitachse

Der Video-Client bietet die Möglichkeit einen Teilbereich der Zeitachse zu markieren (vgl. Abbildung 6.2 direkt unterhalb des eigentlichen Videobereiches). Dieses Markieren der Zeitachse ist allerdings nur 'pixelgenau'. Das ist so zu verstehen: die Zeitachse ist stets 320 Pixel breit. Der Benutzer kann darauf mit der Maus einen Bereich markieren. Dieser Bereich wird in Pixel gemessen und anschließend in ein entsprechendes Zeitfenster umgerechnet. Bei kurzen Videos stellt dies im Allgemeinen kein Problem dar, da hier ein Pixel einer sehr kurzen Zeitspanne entspricht. Bei langen Videos, wie zum Beispiel einem Spielfilm, kann jedoch ein Pixel bereits eine halbe Minute oder mehr bedeuten. Diese Zeitspanne mag bereits länger sein, als der Benutzer eigentlich annotieren, also markieren, wollte. Dieser Umstand soll anhand zweier Beispiele genauer erläutert werden:

Ein kurzes YouTube Video
Das Video dauert 5 Minuten, das sind 300 Sekunden Ein Pixel auf der Zeitachse entspricht daher $\frac{300}{320} = 0.94$ Sekunden. Diese Genauigkeit wird in der Regel ausreichend sein.

Ein Spielfilm
Das Video dauert 90 Minuten, das sind 5400 Sekunden.Ein Pixel auf der Zeitachse entspricht daher $\frac{5400}{320} = 16.88$ Sekunden. Diese Genauigkeit kann schon nicht mehr ausreichend für die Benutzer sein.

Als Abhilfe könnte der Video-Client eine Zoomfunktion auf der Zeitachse anbieten. Oder die Maus 'snapt' auf die aktuelle Position des Playheads. Diese Funktion könnte ein ähnliches Verhalten haben, wie es auch bei diversen Zeichenprogrammen zur Anwendung kommt: der Mauszeiger rastet dabei auf Hilfslinien oder Kanten ein (vgl. Adobe Photoshop[1] oder The Gimp[2]). Generell scheint das Problem der genauen Zeitannotation kein leichtes.

[1] Weitere Infos zu dem Zeichenprogramm 'Adobe Photoshop' sind unter `http://www.adobe.com/de/products/photoshop/photoshop/` zu finden

[2] Weitere Infos zu dem Zeichenprogramm 'The Gimp' sind unter `http://www.gimp.org/` zu finden

Verschiedene GUI- und Feldstudien wären nötig, um eine einfache, sinnvolle und vor allem für die Benutzer intuitive Markierung zu ermöglichen.

6.3.4 FlashVars

Das Flash9-Plugin lädt entsprechend der Parameter in den FlashVars entweder den Bild- oder den Videoclienten. Dabei wird der annotierte Medieninhalt immer direkt aus einem Fedora-Repository bezogen. Im Produktionseinsatz mag es aber Sinn machen, den Medieninhalt von einer beliebigen, eindeutigen URL (vgl. Abbildung 5.2) und nicht direkt aus dem Fedora-Repository zu beziehen.

Eine Änderung der Flash Parameter (FlashVars) kann bei diesem Problem Abhilfe schaffen. Es werden sowohl für den Medieninhalt als auch für dessen Annotationen die eindeutigen URIs als Parameter übergeben.

In den zwei folgenden Beispielen gibt der Parameter `mediaURI` die eindeutige URI für den Medieninhalt an, `fasURI` stellt den, für den Medieninhalt entsprechenden, eindeutigen REST Aufruf des Annotations-Servers dar.

```
mediaURI=http://localhost:8080/fedora/get/content:3/DS1&fasURI=http://localhost
  :8080/fedoraAnnotations/pid/content:3&mime=video/x-flv
```

```
mediaURI=http://www.encyclopedia.chicagohistory.org/images/10436.jpeg&fasURI=http
  ://localhost:8080/fedoraAnnotations/pid/images:10436/DS&mime=image/jpeg
```

Anhand des angegeben MIME-Types wird entweder der Bild- oder der Videoclient geladen. Das Flash Programm könnte zuerst den Medieninhalt laden und dann dessen MIME-Type bestimmen. Erst in einem zweiten Schritt wird der dafür richtige Client geladen. Das Flash-Plugin benötigt somit auch den entsprechenden Flash Parameter nicht mehr.

```
mediaURI=http://localhost:8080/fedora/get/content:100/DS1&fasURI=http://localhost
  :8080/fedoraAnnotations/pid/content:100
```

6.4 Zusammenfassung

In diesem Kapitel wurden mögliche bzw. sinnvolle Erweiterungen für die einzelnen Bereichen des Frameworks gezeigt. Die einzelnen Erweiterungen wurden, der Übersichtlichkeit halber, in die entsprechenden Kategorien bzw. Unterkapiteln Server, Metadaten und Client unterteilt.

Kapitel 7

Zusammenfassung und Schlussfolgerungen

In diesem Kapitel wird noch einmal eine Zusammenfassung über die gesamte Arbeit gegeben. Den Abschluss bilden die daraus resultierenden Schlussfolgerungen.

7.1 Zusammenfassung

Annotationen werden dazu verwendet das verteilte Wissen der Benutzer zu sammeln. Die einzelnen Kommentare haben aber nicht nur für den Autor eine Bedeutung, sondern können auch für die Allgemeinheit von großem Nutzen sein. Denn sie können die Beschreibung existierender Medieninhalte erweitern und so um wichtige Informationen semantisch anreichern.

In den meisten derzeit am Markt befindlichen Systemen sind die Annotations-Features direkt integriert. Diese Herangehensweise hat jedoch einen entscheidenden Nachteil: ein nachträgliches Hinzufügen eines neuen Medientyps impliziert so gut wie immer eine nicht zu unterschätzende Änderung des bereits bestehenden Systems.

Im Rahmen dieser Diplomarbeit wurde daher ein Annotations-Framework entwickelt, dass *unabhängig* vom eingebetteten System arbeitet. Das Framework stützt sich dabei auf das Fedora-Repository, um seine eigenen Annotationsdaten zu speichern. Ein Server übernimmt die Aufgabe alle Zugriffe auf die Annotationsdaten zu regeln. Er kann zu einem bestehenden Medienobjekt alle Annotationen selektieren und diese Liste von Annotationen in einem XML-Dokument an den aufrufenden Clienten zurückschicken. Er kann neue

Annotationen anlegen, bestehende ändern oder löschen. Alle Operationen des Servers sind dabei als einfaches REST-Interface ausgeführt.

Der Server kann, aus Gründen der Interoperabilität mit anderen Systemen bzw. Standards, die Liste auch in andere Formate transformieren bevor er die Daten an den Clienten schickt. Diese Transformationen sind frei konfigurierbar und stützen sich auf XSLT-Stylesheets. Im Zuge der Implementierung wurden bereits zwei Transformationen definiert: `annotea` transformiert die Liste in das, vom W3C standardisierte, Annotea Format und `html` wandelt die Liste in XHTML um, sodass sie von einem beliebigen Webbrowser direkt interpretiert werden kann.

Jeder Medientyp (Text, Bild, Video, etc) benötigt einen speziellen Clienten, denn einen Text zu annotieren funktioniert grundsätzlich anders als ein Video zu annotieren. Um die Funktionstüchtigkeit des Gesamtsystem zu zeigen, wurden für diese Arbeit zwei Web-Clienten für zwei verschiedene Medientypen implementiert. Der eine Client kann Bilder, der andere Client kann Videos annotieren. Die beiden Clienten wurden für das Flash9-Webbrowser Plugin programmiert. Diese Technologie hat den Vorteil, dass sie für nahezu jeden Webbrowser und jedes Betriebsystem zur Verfügung steht. Auch ist die Einbindung der Clienten dadurch sehr einfach zu realisieren, wodurch die Änderungen am Ursprungssystem minimal bleiben.

7.2 Schlussfolgerungen

Der Trend im Web geht immer mehr in Richtung Multimedia. Zu Beginn bestanden die Webseiten aus formatiertem Text, später kamen auch Bilder dazu. Mittlerweile gibt es unzählige Plattformen, die Bilder oder Fotos sammeln und seit geraumer Zeit finden auch immer mehr Videos den Einzug in das tägliche Geschehen des Webs. Auf der anderen Seite bilden Annotationen einen weiteren Grundpfeiler im Web. Die Benutzer sind so in der Lage ihre Meinungen und ihr Wissen mit der Allgemeinheit zu teilen. Bilder, Videos oder ganze Webseiten werden mit Annotationen semantisch angereichert. Sie geben damit den einzelnen Benutzern die Möglichkeit aktiv am Inhalt der Plattformen zu partizipieren und in Kontakt mit der Allgemeinheit zu treten.

Das Fedora-Repository wurde für die vielschichtigen Anforderungen von Digital Libraries entworfen. Alle Zugriffe auf das Repository und der darin gespeicherten Daten sind über wohldefinierte Web-Services gelöst, wodurch eine prinzipielle technische Interoperabilität gewährleistet ist. Die Daten müssen dabei nicht direkt im der Datenbank des Repositories gespeichert sein, denn Fedora ist in der Lage die Daten entweder von externen

Quellen zu beziehen oder sie von Web-Services berechnen zu lassen. Das Fedora-Repository bietet auch die Möglichkeit semantische Relationen zwischen den gespeicherten Objekten des Repositories herzustellen. Auch das implementierte Annotations-Framework benutzt das Fedora-Repository um seine eigenen Annotationsdaten zu speichern und zu verwalten. Dabei werden die Verknüpfungen zwischen dem annotierten Medienobjekt und den Annotationsobjekten mithilfe der semantischen Relationen von Fedora ausgedrückt. Der Ressource-Index von Fedora hilft dabei alle Annotationen eines Medienobjekts wieder zu finden.

Die Annotationsclienten stellen das Bindeglied zwischen dem Framework und den Benutzern dar. Daher ist eine einfache und intuitive Bedienung dieser Software die Grundvoraussetzung für den Erfolg des gesamten Frameworks. In dieser Arbeit wurden die Clienten mit der Flash9 Technologie implementiert, da das Framework in einer webbasierten Umgebung arbeitet. Dabei hat sich die Flash9-Technologie gegenüber der Javascript-Technologie als die bessere erwiesen, denn das Flash9-Browserplugin ist nahezu für jeden Browser und jedes Betriebssystem erhältlich, wobei sowohl das Aussehen als auch das Verhalten der Applikation auf jeder Plattform absolut ident ist. Des weiteren ist derzeit nur Flash als Webtechnologie in der Lage Videos programmatisch abzuspielen und zu manipulieren. Dies wiederum ist für einen Annotationsclienten, der auf Videos spezialisiert ist, unbedingt notwendig. Nicht zu letzt ist ein Flash-Client mit weniger Aufwand in ein bereits bestehendes System zu integrieren als dies mit einer reinen Javascript-Lösung möglich wäre.

Anhang A

Das Fedora Repository

A.1 Installation und Konfiguration

In diesem Abschnitt soll kurz die Installation und eine minimale Konfiguration des Fedora Repositories vorgestellt werden. Dabei beziehen sich alle Angaben auf die stabile Version 2.2.1 des Softwarepakets.

A.1.1 Vorbereiten des Zielsystems

Als erstes muss der Zielrechner alle Voraussetzungen für die Installation erfüllen. Fedora wurde durchgängig in Java implementiert, daher setzt die Software auch Java vorraus. Selbst wenn der oder die Benutzer das gesamte Paket nicht selber kompilieren wollen, muss eine Java SDK vorhanden sein. Die Java SDK muss in einer Version größer gleich 5 vorliegen. Für diese Arbeit wurde die gerade aktuelle Java SE SDK Version 6 Update 5 herangezogen.

Optional kann Fedora auch einen bereits installierten Servlet-Container benutzen. Dieser muss die Servlet-Spezifikation Servlet 2.4 / JSP 2.0 oder höher entsprechen – Apache Tomcat 5.0.28 oder höher entspricht dieser Spezifikation. Des weiteren kann Fedora eine bereits existierende Relationale Datenbank verwenden. Für diese müssen geeignete JDBC-3 Treiber vorhanden sein. Zum Beispiel eigenen sich MySQL, Oracle9 oder Postgres[1]. Soll Fedora neu übersetzt werden, so muss auch Apache Ant 1.6.5 oder höher am Zielrechner verfügbar sein.

Vor der eigentlichen Installation von Fedora müssen einige Environment-Variablen korrekt gesetzt werden:

[1] Mit diesen drei Datenbanken wurde Fedora erfolgreich getestet

- JAVA_HOME
  Diese Variable muss auf den Installationspfad der Java-SDK zeigen.
- FEDORA_HOME
  Diese Variable muss auf den zukünftigen Installationspfad von Fedora zeigen.
- CATALINA_HOME
  Diese Variable wird nur dann benötigt, wenn Fedora mit einen bereits vorhandenen Servlet-Container (z.B. Apache Tomcat) verwenden soll. Die Fedora Web-Services werden dann in diesem Container installiert.

A.1.2 Fedora beziehen

Im zweiten Schritt kann die Installationsroutine von Fedora aus dem Internet heruntergeladen werden. Die Adresse dazu lautet
`http://www.fedora.info/download`.
Die Datei `fedora-2.2.1-installer.jar` selbst ist eine ausführbares Java Archiv.

A.1.3 Fedora installieren

Im dritten Schritt wird die Fedora Installationsroutine aufgerufen:

```
java -jar fedora-2.2.1-installer.jar
```

Selbstverständlich muss der aufrufende User auch die Berechtigung zum schreiben in den Zielverzeichnissen haben. Konkret: in dem Verzeichnis, auf dem die Environment-Variable `FEDORA_HOME` zeigt. Wenn Fedora in einem bereits existierenden Servlet-Container installiert werden soll, so muss der User auch in den Verzeichnissen des Servlet-Containers schreiben können.

Nach dem Aufruf fragt die Routine, wie Fedora installiert werden soll. Es stehen drei Varianten zur Auswahl:

Quick Fedora und all seine Komponenten werden in ein eigenes Verzeichnis installiert. Zusätzlich werden sowohl ein eigener Tomcat-Server (für die Web-Services, die Fedora anbietet), ein Kowari-Triplestore-Server (für die Objekt-Relationen) als auch eine integrierte McKoi SQL-Datenbank (als Storage) mitinstalliert. Alle Serverkomponenten werden bereits für den Einsatz von Fedora vorkonfiguriert. Zusätzlich

werden alle Client-Komponenten installiert. Diese Variante ist die Einfachste und besonders zum Testen von Fedora geeignet. Jedoch bietet sie keine verschlüsselten Netzwerkdienste (SSL) und keine XACML-Policy. Diese können natürlich zu einem späteren Zeitpunkt nachinstalliert bzw. konfiguriert werden.

Costum Diese Variante ist für den Produktionseinsatz gedacht. Der User hat die Möglichkeit so gut wie alle Parameter selbst zu wählen: Sei es nun die zu verwendende Datenbank, die Fedora für sein Storage nutzt. Sei es der Servlet-Container, den Fedora für seine Web-Services (API-A und API-M) nutzt. Oder seien es einfach nur die Host- und Porteinstellungen für die Web-Services. Erweiterte Sicherheitseinstellungen, wie SSL und XACML, lassen sich installieren. Wenn Fedora Verschlüsselung via SSL für die Management-API (API-M) verwenden soll, und dies ist für den Produktionseinsatz unbedingt zu empfehlen!, so muss eine zusätzliche Environment-Variable gesetzt werden: `JAVA_OPTS`. Diese Variable muss javax.net.ssl.trustStore und javax.net.ssl.trustStorePassword abbilden. Auch hier werden zusätzlich alle Client-Komponenten installiert.

Client Diese Variante installiert nur die Client-Komponenten von Fedora. All diese Programme, mit Ausnahme von `fedora-admin`, sind für die Kommandozeile konzipiert und befinden sich im Verzeichnis `FEDORA_HOME/client`. Die folgenden Clients werden mit Fedora mit ausgeliefert:

- `fedora-admin`
  Ist ein komplettes Administrations-Interface mit einer GUI. Dieser Client wird noch genauer in Abschnitt A.2 beschrieben.
- `fedora-dsinfo`
  Zeigt genaue Informationen zu einem Datenstrom eines digitalen Objektes an.
- `fedora-export`
  Export von digitalen Objekten in verschiedenen Formaten (derzeit FOXML oder METS).
- `fedora-find`
  Bietet eine einfache Suche über die indizierten Felder innerhalb des Repositories. Für eine erweiterte Suche bietet sich das Such-Service oder aber der `fedora-admin` Client an.
- `fedora-ingest`
  Dieser Client kann ein neues digitales Objekt in das Fedora Repository einspielen.
- `fedora-ingest-demos`
  Hierbei handelt es sich um ein Skript, das die mitgelieferten Demo-

Objekte in das Repository einspielt. Das Skript bedient sich hierbei des `fedora-ingest`-Clients.

– `fedora-purge`
Löscht digitale Objekte wieder aus dem Repository.

Es folgt ein kompletter Auszug einer Installation in der Variante 'Quick'. Die Environment-Variable `JAVA_HOME` wurde zuvor Systemweit[2] gesetzt; `FEDORA_HOME` wird unmittelbar vor dem Aufruf der Installations-Routine gesetzt. Für den weiteren Betrieb muss auch diese Variable für das gesamte System gesetzt sein, damit alle einzelnen Pakete (Server und Client) korrekt funktionieren.

```
root@Kubuntu1:~/installers# export FEDORA_HOME=/opt/fedora-2.2.1
root@Kubuntu1:~/installers# java -jar fedora-2.2.1-installer.jar

**********************
  Fedora Installation
**********************

To install Fedora, please answer the following questions.
Enter CANCEL at any time to abort the installation.
Detailed installation instructions are available at:
        http://www.fedora.info/download/

Installation type
-----------------
The 'quick' install is designed to get you up and running with Fedora
as quickly and easily as possible. It will install Tomcat and an
embedded version of the McKoi database. SSL support and XACML policy
enforcement will be disabled.
For more options, including the choice of hostname, ports, security,
and databases, select 'custom'.
To install only the Fedora client software, enter 'client'.

Options : quick, custom, client

Enter a value ==> quick


Fedora home directory
---------------------
This is the base directory for Fedora scripts, configuration files, etc.
Enter the full path where you want to install these files.


Enter a value [default is /opt/fedora-2.2.1] ==>


Fedora administrator password
-----------------------------
Enter the password to use for the Fedora administrator
(fedoraAdmin) account.
```

[2]Unter Ubunutu-Linux ist dafür das file `/etc/environment` zuständig. Unter MS-Windows sind diese Einstellungen bei den Computer-Eigenschaften zu finden.

```
Enter a value ==> admin


Preparing FEDORA_HOME...
        Configuring fedora.fcfg
        Installing beSecurity
Installing Tomcat...
Preparing fedora.war...
Processing web.xml
Deploying fedora.war...
Deploying fop.war...
Deploying imagemanip.war...
Deploying saxon.war...
Deploying fedora-demo.war...
Installing embedded McKoi...
Installation complete.

-----------------------------------------------------------------------
Before starting Fedora, please ensure that any required environment
variables are correctly defined
        (e.g. FEDORA_HOME, JAVA_HOME, JAVA_OPTS, CATALINA_HOME).
For more information, please consult the Installation & Configuration
Guide, located online at
        http://www.fedora.info/download/ or locally at
        /opt/fedora-2.2.1/docs/userdocs/distribution/installation.html
-----------------------------------------------------------------------

root@Kubuntu1:~/installers#
```

A.1.4 Fedora konfigurieren

Im weiteren wird, der Einfachheit halber, davon ausgegangen, dass Fedora mit der Installations-Variante 'Quick' installiert wurde.

Fedora benutzt für seine Serverkomponenten ein einziges File für alle Konfigurationseinstellungen. Dieses File befindet sich im Verzeichnis `FEDORA_HOME/server/config` und heißt `fedora.fcfg`.
Dieses File soll nun zum editieren geöffnet werden. Dies kann mit einem beliebigen Text-Editor geschehen (zum Beispiel Vim), da es sich bei dem Konfigurationsfile um ein normales XML-Dokument handelt.

Resource-Index aktivieren

Als erstes soll der Resource-Index (siehe auch Abschnitt 2.3.3) aktiviert werden. Standardmäßig ist dieses Service inaktiv. Der Resource-Index ermöglicht eine Suche nach Dublin-Core Feldern und vor allen nach Relationen zwischen digitalen Objekten, wie sie zum Beispiel eine Annotation für ein digitales Objekt darstellt. Die Relationen werden im Kowari-Triplestore[3] als

[3]Das Kowari-Triplestore ist eine OpenSource-Datenbank, die auf das Speichern von RDF Tripel spezialisiert ist. Mit der Abfragesparche iTQL können die gespeicherten Daten abgefragt werden. [21]

Ontologie[4] abgelegt.

Der folgende Eintrag muss gesucht und geändert werden:

```
...
<module role="fedora.server.resourceIndex.ResourceIndex" class="
  fedora.server.resourceIndex.ResourceIndexModule">
  <comment>Supports the ResourceIndex.</comment>
  <param name="level" value="2">
    <comment>(required)
          Index level. Currently, only 0, 1 and 2 are supported
          levels. 0 = off: do not load the ResourceIndex 1 = basic:
            system
          metadata, RELS-EXT, disseminations 2 = basic + method
            permutations
          WARNING: changing the level (except to 0) requires
          running the Resource Index Rebuilder.</comment>
  </param>
  ...
```

Der Wert von `value` muss von `0` auf `1` oder `2` geändert werden, damit der Resource-Index aktiv ist. Nur dann werden die entsprechenden Daten in der Ontologie des Triplestore ablegt.

```
<param name="level" value="2">
```

Dadurch wird der Resource-Index beim Starten von Fedora aktiviert. Sollten schon zuvor Daten in Fedora eingepflegt worden sein, so wurden diese noch nicht im Resource-Index gespeichert. Sie können somit bei einer Suche auch noch nicht gefunden werden. Allerdings kann der Resource-Index mit dem *Resource Index Rebuilder* wieder komplett neu aufgebaut werden. Dabei handelt es sich um ein kleines Programm (`fedora-rebuild`) für die Kommandozeile, das ebenfalls von Fedora mitinstalliert wurde. Es befindet sich im Verzeichnis `FEDORA_HOME/server`.

Namespace für die Annotationen definieren

Das Fedora-Annotation-Framework sammelt all seine Annotationen in einem eigenen Namespace. Dieser Namespace muss `annotation` heißen, der allerdings nach der Installation von Fedora noch nicht existiert. Er muss er händisch nachgetragen werden, damit er für das Annotations-Framework zur Verfügung steht.

[4] Unter Ontologie versteht man eine Wissenrepräsentation, die u.a. als eine Menge von Tripeln dargestellt werden kann. Ein Tripel besteht dabei aus eine Subjekt, einem Prädikat und einem Objekt. Das Subjekt gibt an, auf was sich das Tripel bezieht, das Prädikat gibt die zu beschreibende Eigenschaft an, das Objekt stellt die Ausprägung der Eigenschaft dar. Beispiel: `Katze -- Haare -- Schwarz`. Hier ist das Subjekt die Katze, Haare ist das Prädikat und Schwarz das Objekt. Dieses Tripel sagt aus, das die Katze schwarze Haare hat. [46] [41]

Der folgende Eintrag muss gesucht und geändert werden:

```
...
<param name="retainPIDs" value="demo test changeme fedora-bdef
  fedora-bmech tutorial">
  <comment>Namespaces of PIDs to retain during the ingest process.
     When an
            object is ingested, Fedora normally allocates a unique
               PID within
            pidNamespace for it regardless of what the object says
               its PID is. This
            option provides a way to override that behavior on a
              per-pid-namespace
            basis. If specified, this should be a space-delimited
              list of pid
            namespaces that will be accepted in the object as-is.
              Default value is
            &amp;quot;demo test&amp;quot;.</comment>
</param>
...
```

Diese Liste von `retainPIDs` stellt alle bekannten Namespaces des Repositories dar. Um den neuen Namespace für die Annotationen aufzunehmen, muss dieser einfach zu der Liste hinzugefügt werden:

```
<param name="retainPIDs" value="demo test changeme fedora-bdef
  fedora-bmech tutorial annotation">
```

Des weiteren definiert Fedora einen Default-Namespace. Dieser wird von Fedora immer dann verwendet, wenn ein neues digitales Objekt in das Repository eingespielt wird, ohne dass ein Namespace für dieses neue Objekt explizit angegeben wird. Nach der Installation von Fedora ist der Default-Namespace `changeme`. Wenn dieser Name nicht gewünscht wird, so kann er vom Administrator geändert werden, denn auch der Default-Namespace ist im Konfigurationsfile gespeichert.

```
...
<param name="pidNamespace" value="changeme">
  <comment>This is the namespace id for pids of newly-created
     objects.
            This should be unique for a repository. It can be from
               1 to 17
            characters, and may only contain A-Z, a-z, 0-9, &amp;apos
              ;.&amp;apos;, or &amp;apos;-&amp;apos; (dash).</comment>
</param>
...
```

A.1.5 Fedora starten

Wurde bei der Installation die Variante 'Quick' gewählt, so reicht es den installierten Tomcat Servlet-Container zu starten. Dies geschieht auf der Kommandozeile mit

```
FEDORA_HOME/tomcat/bin/startup.sh
```

für Unix/Linux System bzw.

```
FEDORA_HOME\tomcat\bin\startup.bat
```

für MS-Windows.

Wurde die Variante 'Costum' bei der Installation gewählt, muss zuerst die verwendete Datenbank und erst danach der verwendete Servlet-Container gestartet werden.

A.1.6 FOXML

Fedora speichert seine digitalen Objekte als XML-Daten. Diese XML-Daten folgen dem FOXML-Schema [31], welches eine direkte Umsetzung des Fedora digital Object Models ist. Zur Illustration sei hier das exportierte FOXML aus dem Bespiel-Objekt von Abschnitt 2.3.2 gezeigt:

```
<?xml version="1.0" encoding="UTF-8"?>
<foxml:digitalObject PID="content:1"
   fedoraxsi:schemaLocation="info:fedora/fedora-system:def/foxml# http://www.
      fedora.info/definitions/1/0/foxml1-0.xsd"
   xmlns:audit="info:fedora/fedora-system:def/audit#" xmlns:fedoraxsi="http://www.
      w3.org/2001/XMLSchema-instance" xmlns:foxml="info:fedora/fedora-system:def/
      foxml#">
  <foxml:objectProperties>
    <foxml:property NAME="http://www.w3.org/1999/02/22-rdf-syntax-ns#type" VALUE
       ="FedoraObject"/>
    <foxml:property NAME="info:fedora/fedora-system:def/model#state" VALUE="
       Active"/>
    <foxml:property NAME="info:fedora/fedora-system:def/model#label" VALUE="
       Picture of a House"/>
    <foxml:property NAME="info:fedora/fedora-system:def/model#ownerId" VALUE="
       fedoraAdmin"/>
    <foxml:property NAME="info:fedora/fedora-system:def/model#createdDate" VALUE
       ="2008-04-14T14:07:43.843Z"/>
    <foxml:property NAME="info:fedora/fedora-system:def/view#lastModifiedDate"
       VALUE="2008-04-14T14:10:43.812Z"/>
  </foxml:objectProperties>
  <foxml:datastream CONTROL_GROUP="X" ID="AUDIT" STATE="A" VERSIONABLE="false">
    <foxml:datastreamVersion CREATED="2008-04-14T14:07:43.843Z"
      FORMAT_URI="info:fedora/fedora-system:format/xml.fedora.audit" ID="AUDIT.0"
         LABEL="Fedora Object Audit Trail" MIMETYPE="text/xml">
      <foxml:xmlContent>
        <audit:auditTrail xmlns:audit="info:fedora/fedora-system:def/audit#">
          <audit:record ID="AUDREC1">
            <audit:process type="Fedora API-M"/>
            <audit:action>addDatastream</audit:action>
            <audit:componentID>DS1</audit:componentID>
            <audit:responsibility>fedoraAdmin</audit:responsibility>
            <audit:date>2008-04-14T14:09:13.593Z</audit:date>
            <audit:justification>DatastreamsPane generated this logMessage.</
               audit:justification>
          </audit:record>
          <audit:record ID="AUDREC2">
            <audit:process type="Fedora API-M"/>
            <audit:action>addDatastream</audit:action>
            <audit:componentID>DS2</audit:componentID>
            <audit:responsibility>fedoraAdmin</audit:responsibility>
            <audit:date>2008-04-14T14:09:55.265Z</audit:date>
            <audit:justification>DatastreamsPane generated this logMessage.</
               audit:justification>
          </audit:record>
          <audit:record ID="AUDREC3">
```

```
            <audit:process type="Fedora API-M"/>
            <audit:action>modifyDatastreamByValue</audit:action>
            <audit:componentID>DC</audit:componentID>
            <audit:responsibility>fedoraAdmin</audit:responsibility>
            <audit:date>2008-04-14T14:10:43.812Z</audit:date>
            <audit:justification>Title changed</audit:justification>
          </audit:record>
        </audit:auditTrail>
      </foxml:xmlContent>
    </foxml:datastreamVersion>
  </foxml:datastream>
  <foxml:datastream CONTROL_GROUP="X" ID="DC" STATE="A" VERSIONABLE="true">
    <foxml:datastreamVersion CREATED="2008-04-14T14:07:43.843Z" ID="DC1.0" LABEL
      ="Dublin Core Metadata"
      MIMETYPE="text/xml" SIZE="303">
      <foxml:contentDigest DIGEST="none" TYPE="DISABLED"/>
      <foxml:xmlContent>
        <oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http
          ://www.openarchives.org/OAI/2.0/oai_dc/">
          <dc:title>Enter a one-line description of the object.</dc:title>
          <dc:identifier>content:1</dc:identifier>
        </oai_dc:dc>
      </foxml:xmlContent>
    </foxml:datastreamVersion>
    <foxml:datastreamVersion CREATED="2008-04-14T14:10:43.812Z" ID="DC.1" LABEL="
      Dublin Core Metadata"
      MIMETYPE="text/xml" SIZE="226">
      <foxml:contentDigest DIGEST="none" TYPE="DISABLED"/>
      <foxml:xmlContent>
        <oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http
          ://www.openarchives.org/OAI/2.0/oai_dc/">
          <dc:title>Picture of a house</dc:title>
          <dc:identifier>content:1</dc:identifier>
        </oai_dc:dc>
      </foxml:xmlContent>
    </foxml:datastreamVersion>
  </foxml:datastream>
  <foxml:datastream CONTROL_GROUP="M" ID="DS1" STATE="A" VERSIONABLE="true">
    <foxml:datastreamVersion CREATED="2008-04-14T14:09:13.593Z" ID="DS1.0" LABEL
      ="House in normal Resolution"
      MIMETYPE="image/jpeg" SIZE="0">
      <foxml:contentDigest DIGEST="none" TYPE="DISABLED"/>
      <foxml:binaryContent>
              /9j/4AAQSkZJRgABAQEBLAEsAAD/2
                wBDAAUDBAQEAwUEBAQFBQUGBwwIBwcHBw8LCwkMEQ8SEhEPERET
                      <<< image-data of datastream DS1 removed >>>
              2gZqE2NrgLGZIx97B/hqGR7hCdrA5GV5qZriTG519unb29aOZlc6fQ//2Q==
            </foxml:binaryContent>
    </foxml:datastreamVersion>
  </foxml:datastream>
  <foxml:datastream CONTROL_GROUP="M" ID="DS2" STATE="A" VERSIONABLE="true">
    <foxml:datastreamVersion CREATED="2008-04-14T14:09:55.265Z" ID="DS2.0" LABEL
      ="House as Thumbnail."
      MIMETYPE="image/jpeg" SIZE="0">
      <foxml:contentDigest DIGEST="none" TYPE="DISABLED"/>
      <foxml:binaryContent>
              /9j/4AAQSkZJRgABAQEBLAEsAAD/2
                wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
                      <<< image-data of datastream DS2 removed >>>
              EaEoKJFqy6y45JKurAlRLJQIFvS38jX2L//Z
            </foxml:binaryContent>
    </foxml:datastreamVersion>
  </foxml:datastream>
</foxml:digitalObject>
```

Das Objekt hat die PID `content:1`. Dies bedeutet, dass das Objekt im Namespace mit dem Namen `content` abgelegt ist, und in diesem Namespace den Identifier `1` hat.
Der Datenstrom `DC` ist die Beschreibung des Objekts im Dublin-Core Metadaten Format.
Aus dem Datenstrom `AUDIT` geht u.a. hervor, dass der Titel im Datenstrom `DC` bereits einmal von dem User `fedoraAdmin` geändert wurde.
Das Objekt besitzt zwei Datenströme vom MIME-TYPE `image/jpeg`. Diese Datenströme stellen jeweils die gleiche Aufnahme von einem Haus dar

(vgl. Abbildung 2.15). Im Datenstrom `DS1` wurde das Bild mit der vollen Auflösung gespeichert. Der Datenstrom `DS2` enthält die selbe Aufnahme, jedoch in einer geringeren Auflösung.

A.2 Das Fedora-Admin Interface

Zusätzlich liefert Fedora eine sehr mächtige Client-Software mit aus: `fedora-admin`. Dabei handelt es sich um ein Programm mit einem graphischen Interface mit dem sich alle gängigen Aufgaben im Zuge der Administration der digitalen Objekte einfach oder nahezu ohne Vorwissen schnell und bequem erledigen lassen.

Dieser Abschnitt stellt dieses Programm anhand von vier Beispielen vor. Das erste Beispiel behandelt das Anlegen eines Objekts mit mehreren Datenströmen (vgl. Abschnitt 2.3.2 und Abbildung 2.15). Im zweiten Beispiel wird ein neues Objekt angelegt, dessen Daten von einem anderen Rechner verwaltet werden. An Hand des dritten Beispiels wird die prinzipielle Vorgehensweise des Fedora-Annotation-Services vorgestellt: ein neues Annotationsobjekt mit der dazugehörigen Relation `isAnnotationOf` wird manuell erzeugt. Das vierte Beispiel zeigt wie aus einem bestehenden Bild ein neues Bild mit Hilfe der Disseminatoren generiert werden kann.

A.2.1 Beispiel 1: ein neues Objekt mit mehreren Datenströmen

Das Programm `fedora-admin` befindet sich im Verzeichnis `FEDORA_HOME/client`. Nach dem Start erscheint der obligatorische Login-Dialog. Das entsprechende Admin-Passwort wurde während der Installation von Fedora vergeben.

Ein neues digitales Objekt wird mit der Tastenkombination `Crtl-N` bzw. über das Menü `File` → `New` → `Data Object` erzeugt. Danach erscheint ein Dialog, in dem der User eine neue, eindeutige PID vergeben kann. Des weiteren kann bereits der Dublin-Core Titel in diesem Dialog eingetragen werden. Beide Angaben sind optional. Zum einem kann Fedora selbst eine PID vergeben (im Default-Namespace). Zum anderen kann der Titel jederzeit im Dublin-Core Datenstrom (`DC`) geändert werden. Für dieses Beispiel sollen beide Daten vorgeben werden, wie in Abbildung A.1 gezeigt.

Damit wurde das neue Objekt im Repository mit der angegebenen PID `demo:700` erzeugt. Ausserdem hat Fedora einen minimalen Dublin-Core Da-

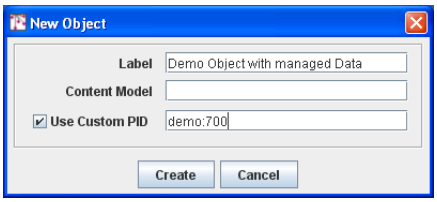


Abbildung A.1: *Fedora-Admin Interface: Ein neues Objekt anlegen.*

tenstrom für uns erzeugt. Abbildung A.2 zeigt die Eigenschaften des neuen Objekts.

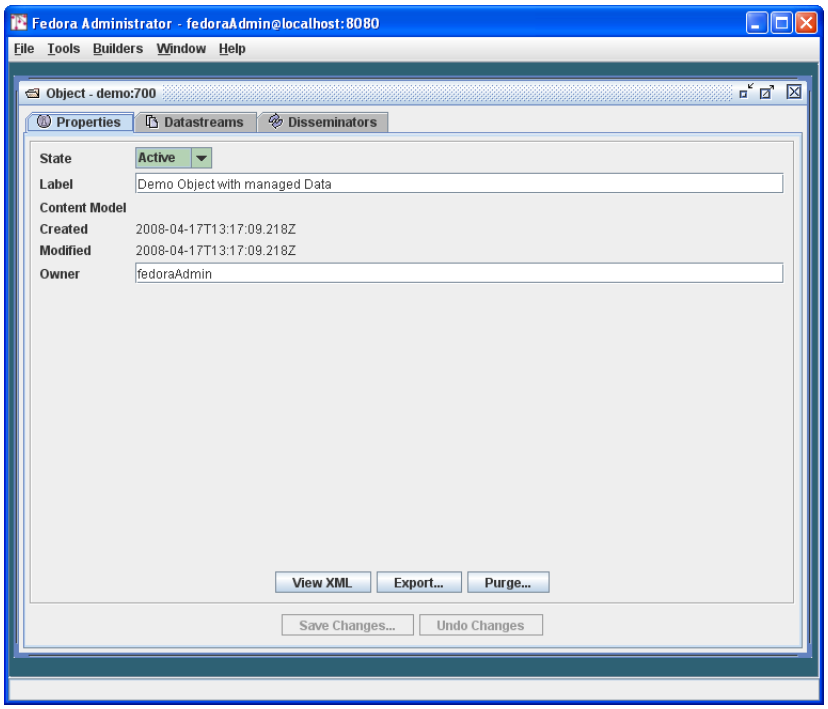


Abbildung A.2: *Fedora-Admin Interface: Eigenschaften eines Objekts.*

Als nächstes werden die eigentlichen Daten im Objekt gespeichert. Alle Datenströme sind im Reiter `Datastreams` zu sehen. So auch der bereits angelegte Dublin-Core. Ein Klick auf `New...` leert das Fenster, ein neuer Datenstrom kann angelegt werden. In diesem Beispiel sollen Bilddaten von Fedora verwaltet werden. Vergeben Sie den Namen `DS1`, wählen Sie `Managed Content`, stellen Sie den MIME-Type auf `image/jpeg`. Klicken Sie anschliessend auf `Import...` und suchen Sie ein jpeg-Bild zum importieren. Wenn alles komplett ist, kann der neue der Datenstrom mit `Save Datastream` ab-

gespeichert werden. Abbildung A.3 zeigt das fertig ausgefüllte Formular.

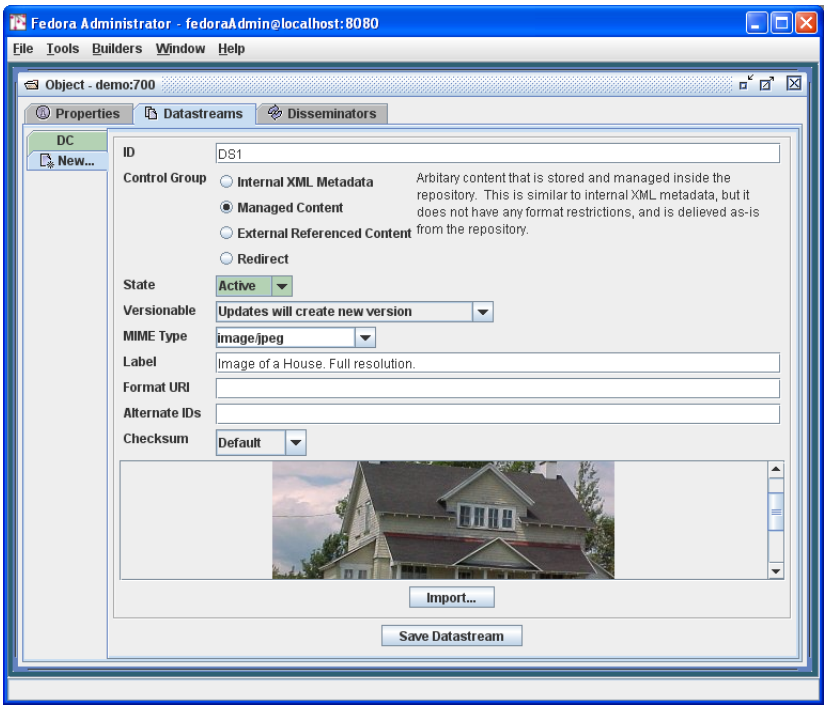


Abbildung A.3: *Fedora-Admin Interface: Ein grosses Bild als Datenstrom.*

Als nächstes wird ein neuer, zweiter Datenstrom angelegt. Dieser enthält das gleiche Bild wie der Datenstrom `DS1`, nur in einer geringeren Auflösung. Das Anlegen und speichern funktioniert genauso wie bei dem Datenstrom `DS1`, mit der Ausnahme das der neue Datenstrom den Namen `DS2` erhält. Abbildung A.4 zeigt das fertig ausgefüllte Formular.

Nach dem speichern wird bei jedem Datenstrom die dazugehörige `Fedora-URL` angezeigt. Mit dieser URL kann der entsprechende Datenstrom über Fedoras API-A REST Interface angesprochen bzw. ausgelesen werden. Mithilfe eines Web-Browsers kann dies leicht überprüft werden, indem eine der drei URLs (aus `DC`, `DS1` oder `DS2`) in dessen Adresszeile kopiert wird.

Abbildung A.5 zeigt nochmals das Endergebnis in einer schematischen Darstellung. Dabei wurden die Daten des Objektes so gezeichnet, wie sie auch von Fedora im Triplestore des Resource Indexes (vgl. Tabelle 2.4 bis 2.7 aus Abschnitt 2.3.3) aufgenommen wurden.

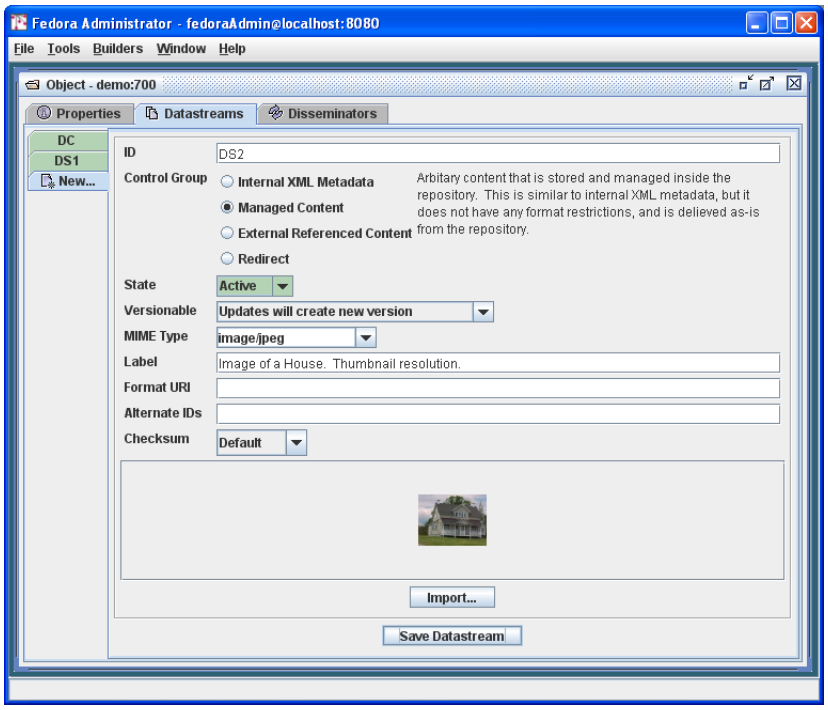


Abbildung A.4: *Fedora-Admin Interface: Ein kleines Bild als Datenstrom.*

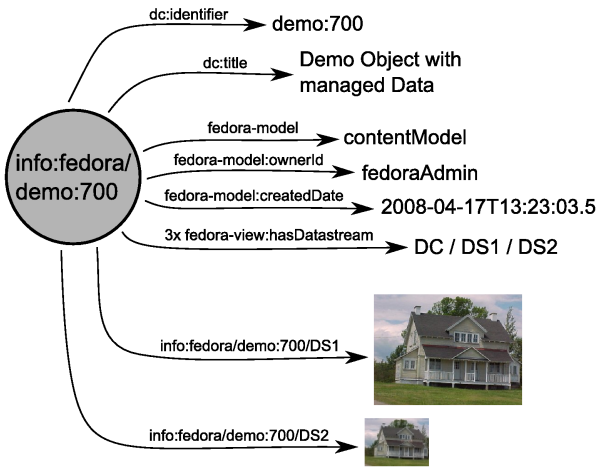


Abbildung A.5: *Das neue digitale Objekt* `demo:700` *und seine Daten. Die Pfeile stellen die Prädikate im Triplestore des Resource Indexes dar, so wie sie von Fedora gespeichert wurden.*

A.2.2 Beispiel 2: ein neues Objekt mit externen Daten

In diesem Beispiel soll Fedora fremde Daten verwalten. Legen Sie wieder ein neues Objekt mit der PID `demo:701` an. Legen Sie einen neuen Datenstrom

mit dem Namen `DS1` an. Diesmal soll der Datenstrom vom Typ `External Referenced Content` sein und der MIME-Type ist `image/gif`. Geben Sie in der Location die URL
`http://www.univie.ac.at/fileadmin/templates/uni_startseite/bilder/uni_logo.gif`
ein. Dies sollte das Logo der Universität Wien sein. Speichern Sie den Datenstrom. Abbildung A.6 zeigt das Ergebnis.

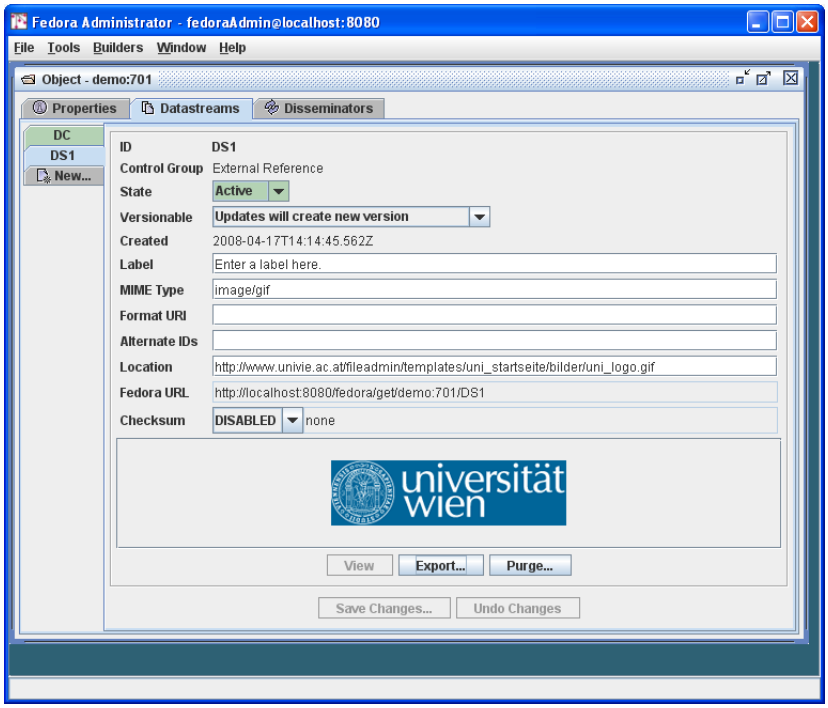


Abbildung A.6: *Fedora-Admin Interface: Ein externes Bild als Datenstrom.*

Sie können auch die Einstiegsseite der Universität Wien
`http://www.univie.ac.at`
als einen eigenen Datenstrom verwalten. In diesem Fall sollte der Datenstrom vom Typ `Redirect` sein, da die Webseite etliche relative Links beinhaltet. Redirect schickt den Browser, wenn er den Datenstrom von Fedora anfordert, direkt zu der entsprechenden URL weiter (mit einem HTTP-Redirect). Der Typ `External` hingegen veranlasst Fedora, die Daten zuerst von der externen Quelle zu holen und erst dann an den Clienten weiterzuschicken.

Abbildung A.7 zeigt eine schematische Darstellung von dem Objekt `demo:702` mit den zwei externen Datenströmen. Das Bild in `DS1` ist als External Referenced Content gespeichert, die Homepage in `DS2` als Redirect. Die Pfeile zeigen den jeweiligen Datenfluss zwischen Client, Fedora und und

den externen Daten.

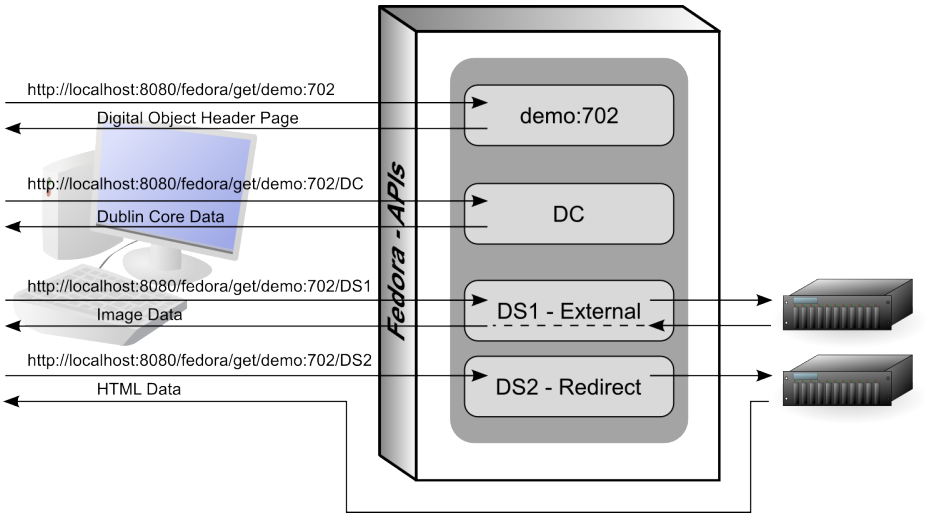


Abbildung A.7: *Das neue digitale Objekt* `demo:702` *und seine externen Daten. Die Pfeile zeigen den jeweiligen Datenfluss zwischen Client, Fedora und und den externen Daten.*

A.2.3 Beispiel 3: ein neues Annotations-Objekt erzeugen

Dieses Beispiel soll zeigen wie das Fedora-Annotation-Service seine Annotationen anlegt. Zu diesem Zweck wird ein neues Objekt mit der PID `demo:702` angelegt.

Legen Sie einen Datenstrom vom Typ `Internal XML Metadata` an. In diesem Beispiel wollen wir uns nicht um den Inhalt der Daten kümmern – es soll zur Veranschaulichung des Prinzips dienen. Geben Sie daher die gekürzten Daten wie in Abbildung A.8 gezeigt ein.

Legen Sie einen weiteren Datenstrom vom Typ `Internal XML Metadata` an. Dieser Datenstrom *muss* `RELS-EXT` heissen, damit diese Daten in den Resource Index aufgenommen werden! Die Daten, die Sie hier eingeben, sollten exakt so aussehen:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rel="info:fedora/fedora-system:def/relations-external
           #">
  <rdf:Description rdf:about="info:fedora/demo:702">
    <rel:isAnnotationOf rdf:resource="info:fedora/demo:700"/>
  </rdf:Description>
</rdf:RDF>
```

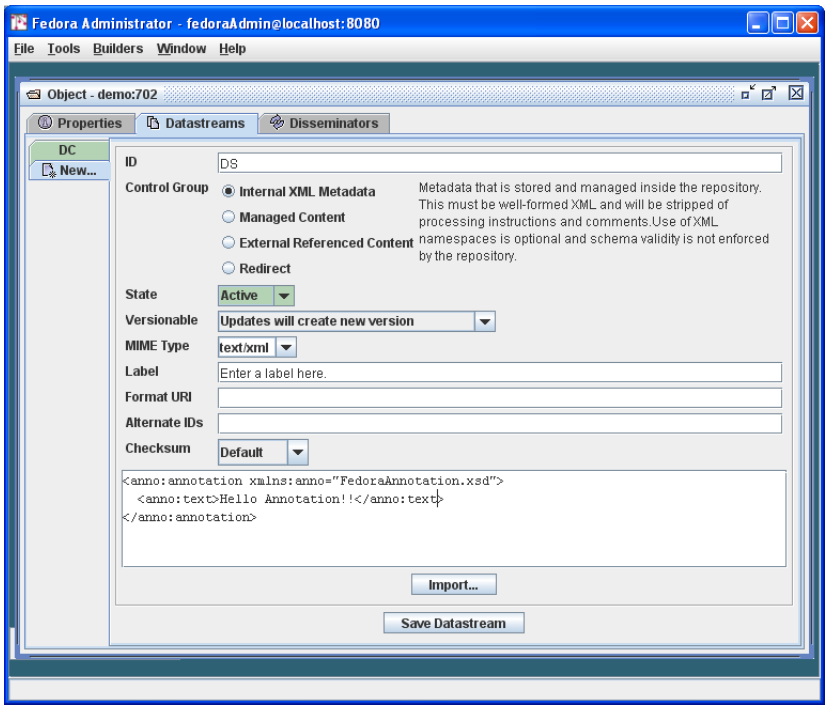


Abbildung A.8: *Fedora-Admin Interface: Ein Datenstrom mit XML Daten.*

Diese Ontologie besagt, dass das Object mit der PID `demo:702` eine Annotation von dem Objekt mit der PID `demo:700` ist. Abbildung A.9 zeigt wieder das ausgefüllte Formular. Wenn Sie nun diesen Datenstrom abspeichern, wird das entsprechendes Trippel im Resource Index aufgenommen.

Nun können Sie diese Relation im RISearch-Interface, wie in Abschnitt 2.3.3 – Abbildung 2.19 beschrieben, suchen.

A.2.4 Beispiel 4: ein Bild dynamisch erzeugen

In diesem Beispiel soll ein Datenstrom dynamisch mit einem Disseminator erzeugt werden. Als Grundlage für dieses Beispiel soll das Objekt mit der PID `demo:700` aus Abschnitt A.2.1 dienen. Zur Erinnerung: es handelt sich um ein digitales Objekt mit zwei Datenströmen. `DS1` beinhaltet ein Bild in der vollen Auflösung. `DS2` enthält das selbe Bild in einer geringeren Auflösung. Es soll von dem größeren Bild, das im Datenstrom `DS1` gespeichert ist, ein Bild in schwarzweiss erzeugt werden. Dieses neue Bild soll durch ein Web-Service, dem `ImageManipulation` Web-Service, berechnet werden.

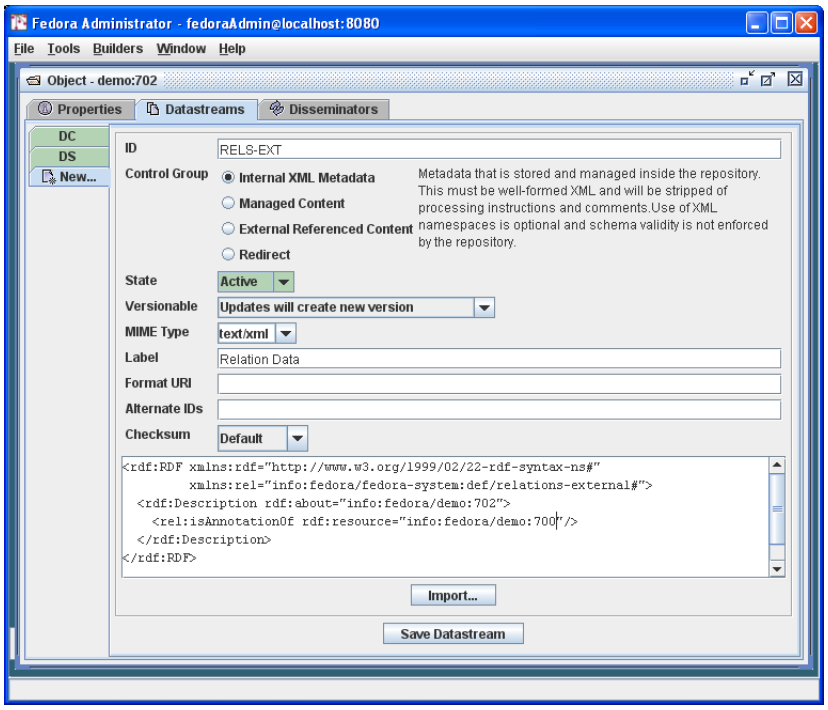


Abbildung A.9: *Fedora-Admin Interface: Objektrelationen definieren.*

Ein neues bDef-Objekt erzeugen

Als erstes soll ein neues bDef-Objekt erzeugt werden. Dieses Objekt definiert die Methoden, die der Disseminator anbieten kann. In diesem Beispiel soll der Disseminator eine einzige Methode, `Grayscale`, haben. Diese Methode soll ein JPEG-Farbbild in ein JPEG-Schwarzweiss Bild umwandeln.

Ein neues bDef-Objekt wird in `fedora-admin` mit `Builders` → `Behavior Definition Builder` erzeugt. Die erste Maske (`General`) ist wie in Abbildung A.10 auszufüllen. Die neue PID soll `fedora-bdef:1`[5] lauten, der Behavior Object Name soll `GrayscaleBDef` sein.

Als nächstes werden die abstrakten Methoden des bDef-Objekts definiert. Hier wird dies nur eine Methode mit dem Namen `Grayscale` sein. Durch einen klick auf den Button `New` wird eine neue Methode angelegt. Der Name und die Beschreibung für die neue Methode ist in Abbildung A.11 ersichtlich. Ein klick auf den Button `Properties` öffnet ein Fenster, in dem mehrere User-Parameter für die Methode angegeben werden können. Grayscale benötigt jedoch keine Parameter von dem Benutzer, deshalb müssen

[5]Der Namespace `fedora-bdef` ist in Fedora 2.2.1 standardmäßig vorhanden.

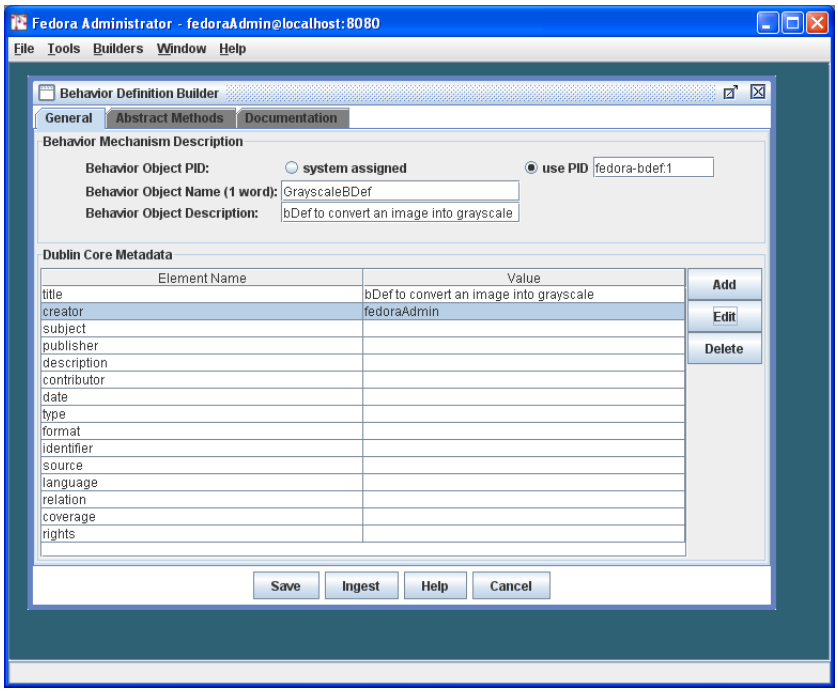


Abbildung A.10: *Fedora-Admin Interface: Ein neues bDef-Objekt anlegen.*

in diesem speziellen Fall auch keine Properties für die Methode angegeben werden[6]. Ein bDef-Objekt braucht zwingend eine Dokumentation, die sich im Document-Tab befindet. Für dieses Beispiel reichen die folgenden Fake-Angaben:

Document Label	temp
Document URL	http://temp.org
Document MIME Type	text/html

Ein klick auf den Button `Ingest` speichert das neue bDef-Objekt im Repository.

Ein neues bMech-Objekt erzeugen

Ein bMech-Objekt stellt eine konkrete Implementierung eines bDef-Objekts dar. Dies bedeutet, dass ein bMech-Objekt angibt welches Web-Service eine abstrakte bDef-Methode implementiert. Das bMech-Objekt speichert dabei

[6]Eine Methode, die ein Bild zum Beispiel skaliert, könnte vom Benutzer eine Eingabe erwarten: den Zoom-Faktor in Prozent. In diesem Fall müsste für diese Methode ein Parameter für den Zoom-Faktor definiert werden. Dieser Parameter mit dem Namen Zoom wäre vom Param Typ USER, würde als Wert übergeben (Pass By = VALUE) und dürfte nur Werte zwischen 0 und 100 annehmen.

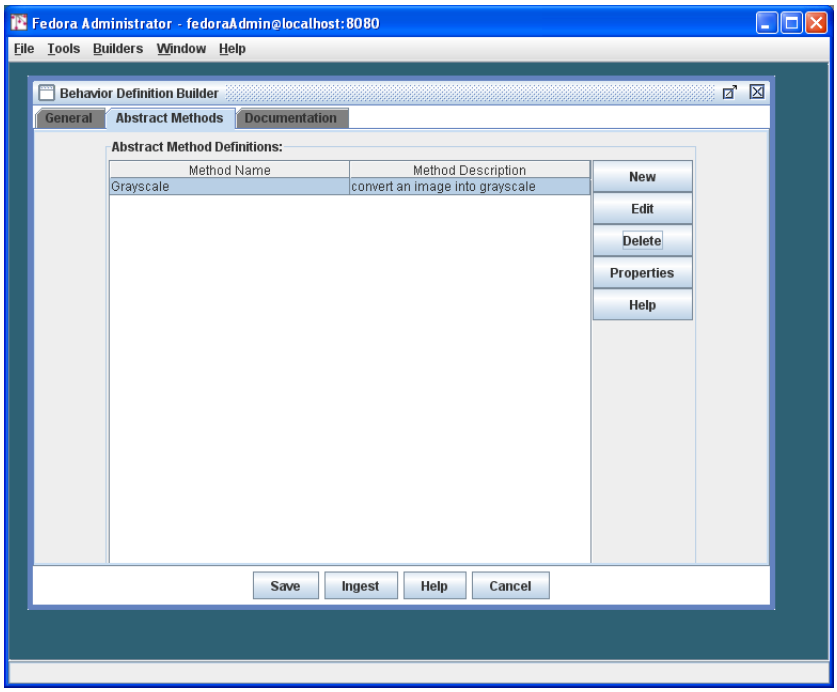


Abbildung A.11: *Fedora-Admin Interface: Eine neue Methode in einem bDef-Objekt anlegen.*

den Ort des Web-Services (URL und/oder WSDL), welche Parameter der Aufruf benötigt (URL der Eingabedaten), wie die Eingabedaten an das Web-Service weitergeleitet werden sollen (als URL oder als Datenstrom), sowie die MIME-TYPES der Eingabe- und Ausgabedaten. Ein bDef-Objekt kann dabei von mehreren verschiedenen bMech-Objekten implementiert werden. Dies hat den Vorteil, dass für ein und die selbe abstrakte Methode mehrere, verschiedene Web-Services verwendet werden können.

Ein neues bMech-Objekt wird in `fedora-admin` mit `Builders` → `Behavior Mechanism Builder` erzeugt. Die erste Maske (`General`) ist wie in Abbildung A.12 auszufüllen. Die neue PID soll `fedora-bmech:1`[7] lauten, der Behavior Object Name soll `GrayscaleBMech` sein. Als das zu Grunde liegende bDef-Objekt soll das gerade erstellte Objekt `fedora-bdef:1` aus der Liste ausgewählt werden.

Im Tab `Service Profile` werden Metadaten zu dem Web-Service, das die abstrakten Methoden des bDef-Objekts implementiert, angegeben. Die `Service Test URL` gibt die Adresse zu dem Web-Service (ohne URL-Parameter) an. `Service Name` und `Service Describtion` dienen zum Doku-

[7]Der Namespace `fedora-bmech` ist in Fedora 2.2.1 standardmäßig vorhanden.

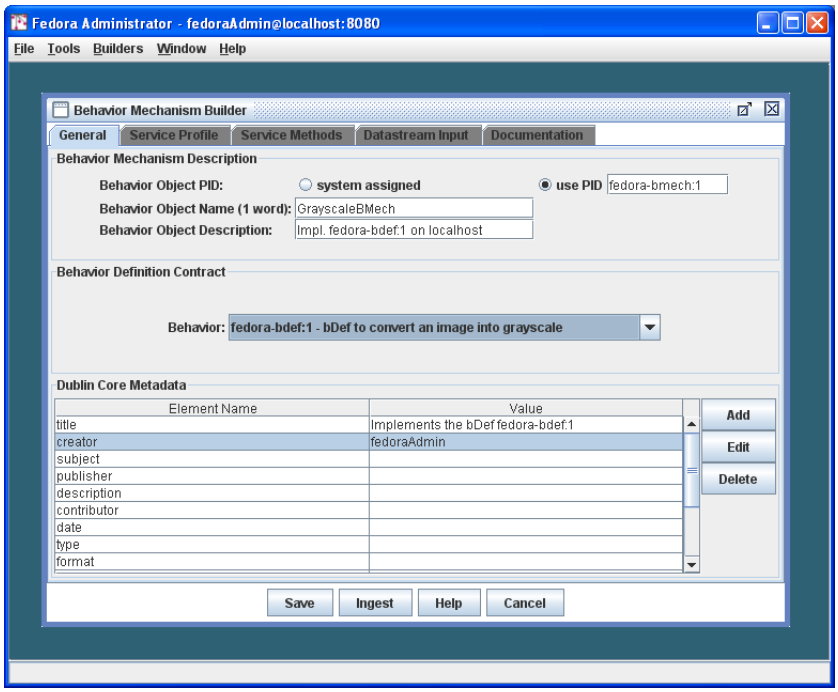


Abbildung A.12: *Fedora-Admin Interface: Ein neues bMech-Objekt anlegen.*

mentieren und haben keinen Einfluss auf den Aufruf des Web-Services. Unter `Messaging Protokoll` kann die Art des Aufrufs eingestellt werden. Mögliche Varianten sind `HTTP-GET`, `HTTP-POST` und `SOAP`. Der `Input MIME Type` bzw. `Output MIME Type` geben das Format der Eingabe- und Ausgabedaten an. Für dieses Beispiel wird der Service `ImageManipulation` verwendet. Abbildung A.13 zeigt die nötigen Einstellungen des Service Profiles.

Nachdem das Profil des Web-Services eingestellt wurden, können die Profile der Methoden, die vom bDef-Objekt geerbt wurden, angegeben werden. Im Tab `Service Methods` werden die konkreten Ausprägungen der abstrakten Methoden des bDef-Objekts konfiguriert. Dieses Tab zeigt bereits alle Methoden an, die durch das bDef-Objekt definiert wurden. In diesem Beispiel ist das die Methode `Grayscale`. Da die Methodennamen bereits vorausgefüllt sind, fehlt nur mehr die `Base URL` und die exakten Angaben für die einzelnen Aufrufe des Web-Services. Zu diesem Zweck muss eine Methode (hier ist dies nur `Grayscale`) selektiert und anschließend der Button `Properties` angeklickt werden. Dies öffnet den `Method Properties` Dialog für die gewählte Methode. Der Prefix der Web-Service URL ist bereits vorausgefüllt, somit fehlt nur noch der folgender Eintrag
`ImageManipulation?op=grayscale&url=(url)`.
Die Angabe "`(url)`" definiert einen Parameter für den Aufruf des Web-

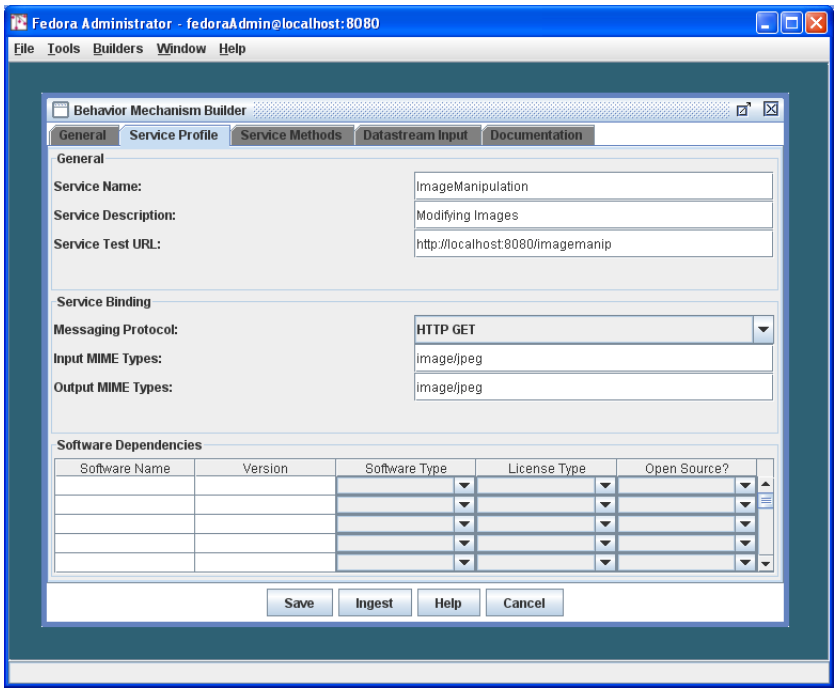


Abbildung A.13: *Fedora-Admin Interface: Das Service Profile eines bMech-Objekts. Hier werden die Daten für ein konkretes Web-Service definiert. Die URL, unter der das Service erreichbar ist, die Art des Aufrufs (HTTP-Get, HTTP-POST oder SOAP) und die MIME Types der Eingabe- und Ausgabedaten.*

Services, der in den `Method Parameter Definitions` genauer spezifiert werden muss. Abbildung A.14 das Profil des Web-Service und die Eigenschaften der konkreten Methode.

Disseminator in einem digital Objekt aufnehmen

Im letzten Schritt wird der erzeugte Disseminator in einem digitalen Objekt verwendet. Für dieses Beispiel soll das Objekt mit der PID `demo:700` aus Abschnitt A.2.1 verwendet werden. Das Tab `Disseminators` des Objekts dient dazu das Objekt mit einem bDef- bzw. bMech-Objekt zu verknüpfen. Sobald in diesem Tab ein Behavior Object aus der Dropdown Liste gewählt wurde, sind dessen abstrakten Methoden und alle möglichen Implementierungen des bDef-Objektes zu sehen. Auf diese Art können nur passende bMech-Objekte zu einem bDef-Objekt ausgewählt werden. Sobald das gewünschte bMech-Objekt ausgewählt wurde, erscheint eine Tabelle mit den nötigen `Bindings`. Binding stellt die Verbindung eines bMech-Parameters mit einem Datenstrom her. Durch einen klick auf den `Add...` Button wird ein

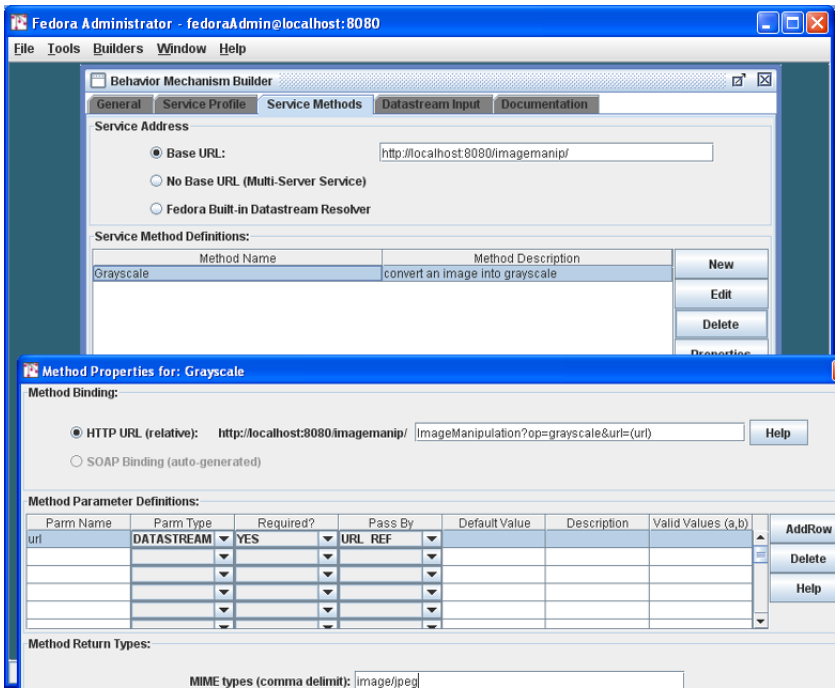


Abbildung A.14: *Fedora-Admin Interface: Service Methods und deren Eigenschaften eines bMech-Objekts. In dem Service Profile werden die Daten für die konkreten Methoden eines Web-Services definiert. Die Properties spezifizieren den exakten Aufruf des Web-Services und können, falls nötig, Parameter enthalten. Hier wird in der URL der Parameter* `(url)` *definiert, der in den Method Parameter Definitions genau spezifiert wird.*

Dialog geöffnet, in dem alle möglichen Datastreams des digitalen Objekts angezeigt werden, die den korrekten MIME-Type haben.

In diesem Bespiel soll das Objekt `demo:700` die Methoden des bDef-Objekts `fedora-bdef:1` bekommen. Das bMech-Objekt `fedora-bmech:1` stellt die Implementierung bereit und benötigt seinerseits einen Parameter mit den Namen `url`. Dieser Parameter soll mit dem Datenstrom `DS1` verbunden werden, damit dieser Datenstrom die Eingabe für das Web-Service wird. Abbildung A.15 zeigt den vollständig ausgefüllten Dialog, `Save Datastream` speichert den neuen Disseminator im digitalen Objekt.

Unter der URL `http://localhost:8080/fedora/get/demo:700/`↩
`fedora-bdef:1/Grayscale`
ist nun das dynamisch berechnete Bild in schwarzweiss abrufbar. Die URL setzt sich dabei aus zwei Komponeten zusammen. Der erste Teil ist die bereits bekannte Zugriffs-URL für ein digitales Objekt. In diesem Fall `fedora/get/demo:700`. Der zweite Teil, der direkt auf den Ersten folgt,

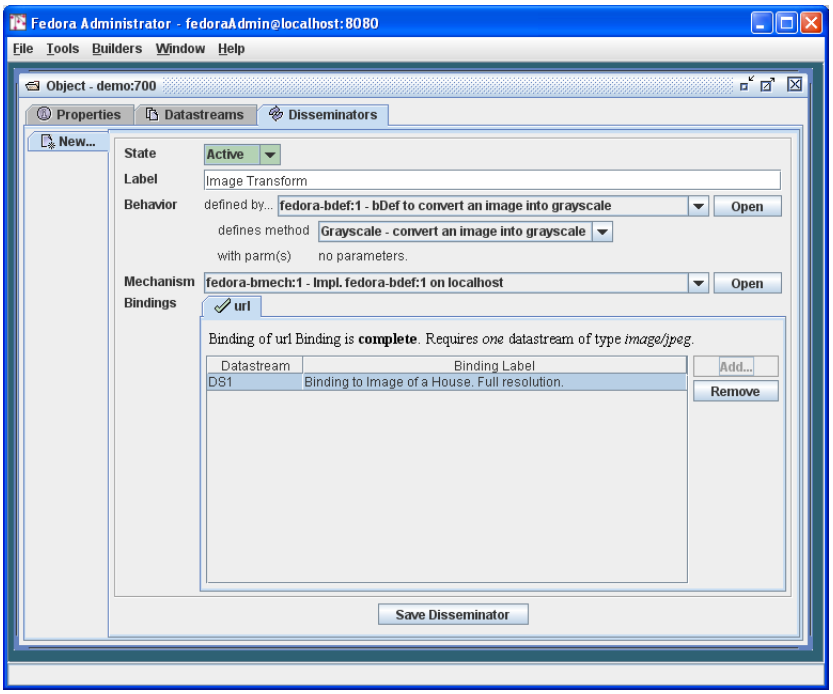


Abbildung A.15: *Disseminator in einem disgital Objekt anlegen. Sobald ein bDef-Objekt aus der Liste gewählt wurde, erscheinen dessen abstrakten Methoden und eine Liste mit allen möglichen bMech-Objekten. Sobald das bMech-Objekt gewählt wurde, erscheint die Bindings Tabelle. In dieser Tabelle werden alle Parameter des bMech-Objekts an einen bestimmtem Datenstrom des digitalen Objekts gebunden.*

bestimmt die Methode des Disseminators, die angewendet werden soll. Hier ist dies die Methode `Grayscale` vom bDef-Objekt `fedora-bdef:1`.

Anhang B

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Wien, im November 2008

Bernd Pinter, BSc

Anhang C

Nachweis der Urheberrechte

Sämtliche Graphiken, mit Ausnahme von Abbildung 2.21 auf Seite 45, wurden zur Gänze von mir selbst angefertigt.

Der Auszug einer eingescanten Seite in Abbildung 1.1 wurde dem Buch 'Nacht und Schimmel' von Stanislaw Lem entnommen [23].

Einige Graphiken, die das Fedora-Repository mit seinen digitalen Objekten betreffen (in Abschnitt 2.3 und A.2), sind aus den Dokumentationen des Fedora-Repositories inspiriert.

Alle Screenshots aus Abschnitt 2.2 (Abbildungen 2.1 bis 2.11) sowie aus Abschnitt 5 (Abbildung 5.1) wurden direkt den jeweiligen Webseiten entnommen. Die Urherber mit den entsprechenden Web-Adressen sind in den Begleittexten der einzelnen Abbildungen angegeben.

Ich habe mich bemüht, sämtliche Inhaber der Bildrechte ausfindig zu machen und ihre Zustimmung zur Verwendung der Bilder in dieser Arbeit eingeholt. Sollte dennoch eine Urheberrechtsverletzung bekannt werden, ersuche ich um Meldung bei mir.

Anhang D

Sourcecode auf der CD-Rom

Der Diplomarbeit liegt eine CD-Rom bei, auf der sich sämtliche Sourcecodes des Frameworks befinden. Neben den Sourcecodes finden sich auch alle Latex- und Bilddatein auf dem Datenträger. Zur besseren Übersicht werden hier die einzelnen Verzeichnisse und ihr Inhalt wiedergegeben.

Verzeichnis 'da_text'

In diesem Verzeichnis befindet sich der LaTeX 2_εSourcecode der Diplomarbeit, inklusive aller Graphiken und Screenshots.

Verzeichnis 'source'

In diesem Verzeichnis befinden sich alle Sourcecodes des Fedora-Annotation-Services, inklusive aller erzeugten XML-Schemas und XSLT-Stylesheets für die implementierten Transformationen. Innerhalb des Verzeichnisses sind die einzelnen Komponenten des Frameworks in eigenen Unterverzeichnissen (Server, Client) zusammengefasst.

Jene Komponenten die in Java programmiert wurden (Server und ein Testclient) wurden mit der Netbeans-IDE implememntiert, wobei auch die dazugeörigen Projektfiles in den Verzeichnissen zu finden sind. Die Clienten wurden mit dem Adobe-FlexBuilder programmiert, wobei beide Clienten in einem FlexBuilder Projektverzeichnis zusammengefasst sind.

Literaturverzeichnis

[1] Maristella Agosti, Nicola Ferro, Ingo Frommholz, and Ulrich Thiel. Annotations in digital libraries and collaboratories – facets, models and usage. In *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries*, pages 244–255, 2004. URL `http://springerlink.metapress.com/content/ah229da5je3ka8y7/?p=9ad245de59374355b7861b8a10506432&pi=1`.

[2] David Bargeron, Anoop Gupta, and A. J. Bernheim Brush. A common annotation framework. Technical report, 2001. URL `ftp://ftp.research.microsoft.com/pub/tr/tr-2001-108.pdf`.

[3] A. J. Bernheim Brush, David Bargeron, Jonathan Grudin, Alan Borning, and Anoop Gupta. Supporting interactions outside of class: Anchored discussions vs. discussion boards. In *Stahl, G. (Ed.): Proc. of CSCL 2002*, pages 425–434, 2002.

[4] Adobe Develper Connetction. Video technology center, 2008. URL `http://www.adobe.com/devnet/video`. [Online; Stand 1, Oktober 2008].

[5] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000. URL `http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf`. [Online; Stand 1, Oktober 2008].

[6] Organization for the Advancement of Structured Information Standards. OASIS eXtensible Access Control Markup Language (XACML), Februar 2005. URL `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#overview`. [Online; Stand 27, Oktober 2008].

[7] Dublin Core Metadata Initiative. Expressing qualified dublin core in rdf/xml, Mai 2002. URL `http://dublincore.org/documents/dcq-rdf-xml/`. [Online; Stand 26, Oktober 2008].

[8] Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1, Jänner 2008. URL http://dublincore.org/documents/dces/. [Online; Stand 26, Oktober 2008].

[9] Open Archives Initiative. The open archives initiative protocol for metadata harvesting — version 2.0, 2004. URL http://www.openarchives.org/OAI/openarchivesprotocol.html. [Online; Stand 1, Oktober 2008].

[10] Bibliographisches Institut. *Duden — die deutsche Rechtschreibung, 22. Auflage*. Brockhaus AG, Mannheim, 2000. ISBN 3-441-04012-2.

[11] ECMA International. Standard ECMA-262 – ECMAScript Language Specification 3rd edition., Dezember 1999. URL http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf. [Online; Stand 7, November 2008].

[12] Jose Kahan and Marja-Ritta Koivunen. Annotea: an open rdf infrastructure for shared web annotations. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 623–632, New York, NY, USA, 2001. ACM Press. ISBN 1581133480. URL http://dx.doi.org/10.1145/371920.372166.

[13] Michael Kay. SAXON — The XSLT and XQuery Processor, 2007. URL http://saxon.sourceforge.net/. [Online; Stand 15. Oktober 2008].

[14] Marja-Riitta Koivunen. Annotea and semantic web supported collaboration. URL http://www.annotea.org/eswc2005/01_koivunen_final.pdf.

[15] Marja-Riitta Koivunen. Web tagging with annotea shared/social bookmarks and topics, 2006. URL http://annotea.org/www2006/annotea.htm.

[16] W3C Kosortium. HTTP/1.1, part 1: URIs, Connections, and Message Parsing, August 2008. URL http://www.ietf.org/internet-drafts/draft-ietf-httpbis-p1-messaging-04.txt. [Online; Stand 7, November 2008].

[17] W3C Kosortium. HTTP/1.1, part 2: Message Semantics, August 2008. URL http://www.ietf.org/internet-drafts/draft-ietf-httpbis-p2-semantics-04.txt. [Online; Stand 7, November 2008].

[18] W3C Kosortium. SOAP version 1.2, 2007. URL http://www.w3.org/TR/soap12-part1. [Online; Stand 1, Oktober 2008].

[19] W3C Kosortium. SPARQL Variable Binding Results XML Format, Jänner 2004. URL `http://www.w3.org/TR/2004/WD-rdf-sparql-XMLres-20041221`. [Online; Stand 1, Oktober 2008].

[20] W3C Kosortium. RDQL - a query language for rdf, Jänner 2004. URL `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109`. [Online; Stand 1, Oktober 2008].

[21] kowari.org. Kowari triplestore version 1.0.3, 2004. URL `http://kowari.org/oldsite/index.htm`. [Online; Stand 1, Oktober 2008].

[22] kowari.org. Kowari's iTQL command reference (kowari triplestore version 1.0.3), 2004. URL `http://kowari.org/oldsite/271.htm`. [Online; Stand 1, Oktober 2008].

[23] Stanislaw Lem. *Nacht und Schimmel (deutsche Ausgabe)*. Suhrkamp, Frankfurt am Main, Germany, Jänner 1976. ISBN 978-3-518-36856-5.

[24] Catherine C. Marshall. Annotation: from paper books to the digital library. *Proceedings of the ACM Digital Libraries '97 Conference, Philadelphia*, pages 131–140, Juli 1997. URL `http://www.csdl.tamu.edu/~marshall/dl97.pdf`.

[25] Catherine C. Marshall and A. J. Bernheim Brush. Exploring the relationship between personal and public annotations. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 349–357, New York, NY, USA, 2004. ACM. ISBN 1-58113-832-6. doi: http://doi.acm.org/10.1145/996350.996432.

[26] Christian Mogensen and Terry Winograd. Shared web annotations as a platform for third-party value-added. Technical report, Stanford University, 1994. URL `http://DLIB2.STANFORD.EDU/diglib/pub/reports/commentor/commentor.ps`.

[27] Erich Neuhold, Claudia Niedereé, Avaré Stewart, Ingo Frommholz, and Bhaskar Mehta. The role of context for information mediation in digital libraries. In Z. Chen, H. Chen, Q. Miao, Y. Fu, E.A. Fox, and E.-P. Lim, editors, *Proc. of the 7th International Conference on Asian Digital Libraries: International Collaboration and Cross-Fertilization (ICADL 2004)*, volume 3334 of *Lecture Notes in Computer Science (LNCS)*, pages 133–143, Heidelberg et al., 2004. Springer. URL `http://www.is.informatik.uni-duisburg.de/bib/pdf/ir/Neuhold_etal:04.pdf`.

[28] Aral Balkan OSFlash. Real time messaging protocol. URL `http://osflash.org/documentation/rtmp`. [Online; Stand 7, November 2008].

[29] The Apache XML Graphics Project. Apache formatting objects processor, 2008. URL http://xmlgraphics.apache.org/fop. [Online; Stand 15. Oktober 2008].

[30] The Fedora Project. Introduction to Fedora Object XML (FOXML), 2008. URL http://www.fedora.info/download/2.2.1/userdocs/digitalobjects/introFOXML.html. [Online; Stand 1, Oktober 2008].

[31] The Fedora Project. Fedora Object XML (FOXML) schema, 2008. URL http://www.fedora.info/definitions/1/0/foxml1-0.xsd. [Online; Stand 1, Oktober 2008].

[32] The Fedora Project. Fedora generic search service version 2.0, 2008. URL http://www.fedora.info/download/2.2.1/services/genericsearch/doc/index.html. [Online; Stand 1, Oktober 2008].

[33] The Fedora Project. Fedora installation and configuration guide, 2008. URL http://www.fedora.info/download/2.2.1/userdocs/distribution/installation.html. [Online; Stand 1. Oktober 2008].

[34] The Fedora Project. Fedora identifiers, 2008. URL http://www.fedora.info/download/2.2.1/userdocs/distribution/installation.html. [Online; Stand 1, Obktober 2008].

[35] The Fedora Project. Fedora RELS-EXT ontologie. rdf-schema, 2008. URL http://www.fedora.info/definitions/1/0/fedora-relsext-ontology.rdfs. [Online; Stand 1, Oktober 2008].

[36] The Fedora Project. Triples in the resource index, 2008. URL http://www.fedora.info/download/2.2.1/userdocs/server/resourceIndex/triples.html. [Online; Stand 1, Oktober 2008].

[37] The Fedora Project. Fedora authorization with XACML policy enforcement, 2008. URL http://www.fedora.info/download/2.2.1/userdocs/server/security/AuthorizationXACML.htm. [Online; Stand 27, Oktober 2008].

[38] Frank Shipman, Morgan Price, Catherine C. Marshall, and Gene Golovchinsky. Identifying useful passages in documents based on annotation patterns. In *In Proceedings of ECDL'03*, pages 17–22. Springer Verlag, 2003.

[39] Andrew S. Tanenbaum. *Computernetzwerke, 4. deutsche Auflage*. Pearson Studium, München, 2003. ISBN 978-3-8273-7046-4.

[40] Stefan Tilkov. A brief introduction to REST, 2007. URL http://www.infoq.com/articles/rest-introduction. [Online; Stand 1, Oktober 2008].

[41] Mike Uschold, Mike Uschold, Michael Gruninger, and Michael Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11:93–136, 1996.

[42] E. M. Voorhees and Lori P. Buckland, editors. *Applying the Annotation View on Messages for Discussion Search*, Gaithersburg, MD, USA, 2005. NIST. URL `http://www.is.informatik.uni-duisburg.de/bib/pdf/ir/Frommholz:05.pdf`.

[43] Wikipedia. Annotation — wikipedia, die freie enzyklopädie, 2008. URL `http://de.wikipedia.org/w/index.php?title=Annotation`. [Online; Stand 8. Oktober 2008].

[44] Wikipedia. Base64 — wikipedia, die freie enzyklopädie, 2008. URL `http://de.wikipedia.org/w/index.php?title=Base64\&oldid=49821555`. [Online; Stand 15. Oktober 2008].

[45] Wikipedia. Flash video — wikipedia, die freie enzyklopädie, 2008. URL `http://de.wikipedia.org/w/index.php?title=Flash_Video\&oldid=50220617`. [Online; Stand 15. Oktober 2008].

[46] Wikipedia. Ontologie (informatik) — wikipedia, die freie enzyklopädie, 2008. URL `http://de.wikipedia.org/w/index.php?title=Ontologie_(Informatik)\&oldid=51531830`. [Online; Stand 15. Oktober 2008].

[47] Wikipedia. Representational state transfer — wikipedia, die freie enzyklopädie, 2008. URL `http://de.wikipedia.org/w/index.php?title=Representational_State_Transfer\&oldid=51374706`. [Online; Stand 15. Oktober 2008].

[48] Wikipedia. Real time messaging protocol — wikipedia, the free encyclopedia, 2008. URL `http://en.wikipedia.org/w/index.php?title=Real_Time_Messaging_Protocol&oldid=247208808`. [Online; Stand 7, November 2008].

[49] Wikipedia. SOAP — wikipedia, die freie enzyklopädie, 2008. URL `http://de.wikipedia.org/w/index.php?title=SOAP\&oldid=51649460`. [Online; Stand 15. Oktober 2008].