Entwicklung einer Web-basierten Bibliotheksverwaltung

Jonas Pohlandt

8. Oktober 2002

Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Studienarbeit

Entwicklung einer Web-basierten Bibliotheksverwaltung

von
cand. wirtsch.-inform. Jonas Pohlandt

Aufgabenstellung und Betreuung:
Prof. Dr. S. Fischer und Dipl. Inform. F. Strauß

Braunschweig, den 8. Oktober 2002

Kurzfassung

Aufgabe dieser praxisbezogenen Studienarbeit war es, eine Web-basierte Verwaltungsapplikation für die Institutsbibliothek des Instituts für Betriebssysteme und Rechnerverbund zu entwerfen und zu implementieren. Die Applikation soll das Suchen innerhalb des Bibliotheksdatenbestandes, die Pflege desselben sowie die Verwaltung des Verleihstatus ermöglichen. Desweiteren sollte ein leighweight CLI-Client entwickelt werden, welcher eine Suche innerhalb des Datenbestandes über die Kommandozeile erlaubt. Durch eine weitere Applikation sollte der in XML vorliegende Datenbestand in das BibTeX Format exportiert werden können. Vorgaben waren XML als Struktur des Datenbestandes sowie die innerhalb der Lehrveranstaltung „Web-Anwendungen mit Java und XML“ vorgestellte Java 2 Enterprise Edition als Basis für die Web Applikation.

Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Prof. Dr. S. Fischer

Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig · Postfach 3329 · 38023 Braunschweig

Mühlenpfordtstr. 23
38106 Braunschweig
Telefax: (05 31) 3 91 – 5936
WWW: http://www.ibr.cs.tu-bs.de/

Prof. Dr. S. Fischer
Telefon: (05 31) 3 91 – 3283

Braunschweig, den 17.01.2002

Aufgabenstellung für die Studienarbeit

Entwicklung einer Web-basierten Bibliotheksverwaltung

vergeben an

Herrn cand. wirtsch.-inform. Jonas Pohlandt

Das Institut für Betriebssysteme und Rechnerverbund verfügt über eine selbstverwaltete kleine Bilbiothek, deren Bestand demnächst elektronisch erfasst wird.

Im Rahmen dieser Studienarbeit soll eine geeignete erweiterbare XML-basierte Repräsentation des Datenbestandes gefunden und als XML Schema definiert werden. Darauf aufbauend sollen flexible Recherche-Operationen (nach Autoren, Titeln, Schlüsselworten, Ausleihzustand, ODER/UND-Verknüpfungen, ...) implementiert und Web- und CLI-basierte Benutzerschnittstellen für die Recherche und Leihstatusbearbeitung erstellt werden. Ferner soll ein Export zur Verwendung des Datenbestandes als BibTeX-Eingabe und für die Verwendung als Referenzen in den Instituts-Webseiten möglich sein.

Laufzeit: 3 Monate

Aufgabenstellung und Betreuung: Prof. Dr. S. Fischer
Dipl.-Inform. F. Strauß

(Prof. Dr. S. Fischer)

Abbildung 0.1: Die Aufgabenstellung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Arbeit beschreibt die Implementierung einer webbasierten Bibliotheksverwaltung, welche am Institut für Betriebssysteme und Rechnerverbund zum Einsatz kommen soll. Die der Lösung zugrundeliegenden Technologien, die Java 2 Enterprise Edition sowie XML (Extensible Markup Language) als universelle Markup Language wurden in der Lehrveranstaltung „Web-Anwendungen mit Java und XML“ im Wintersemester 2001/2002 ausführlich vorgestellt.

Kapitel zwei dieser Arbeit gibt Auskunft über die Motivation und verschafft einen Überblick über die verwendeten Produkte und Basistechnologien. Kapitel drei informiert über bereits vorhandene Lösungen sowie deren Vor- und Nachteile. Kapitel vier beschreibt und erläutert die im Zuge dieser Arbeit erstellte Lösung. Kapitel fünf beinhaltet schliesslich eine Zusammenfassung der Arbeit sowie einen Ausblick auf denkbare Fortführungen der entwickelten Anwendung.

2 Motivation und Überblick

2.1 Motivation

2.1.1 Zur Applikation

Die Aufgabe, eine Anwendung zur Verwaltung der Institutsbibliotheksdaten zu erstellen, enstand aus dem Bedarf nach einer solchen. Die Verwaltung wurde zu Begin der Studienarbeit noch nicht digital bewältigt und soll durch Umstellung auf die in dieser Arbeit beschreibene Anwendung beschleunigt und komfortabler gemacht werden. Die Möglichkeit des Exports der Bibliotheksdaten ins BibTeX Format ist hierbei ein angenehmer Nebeneffekt, welcher mit relativ geringem Aufwand ermöglicht werden kann.

2.1.2 Zur Verwendung der Java 2 Enterprise Edition

Die Applikation ist als Verteilte Anwendung konzipiert. Mehrere Benutzer müssen in der Lage sein simultan mehrere Suchen durchzuführen bzw. Datenpflege zu betreiben. Die J2EE (Java 2 Enterprise Edition[1]) beinhaltet alle hierfür notwendigen Komponenten unter Berücksichtigung einer 3-Tier Architektur (Presentation, Business Logic, Data). Weiterführende Literatur zum Thema J2EE bietet zum Beispiel Flanagan[1].

2.1.3 Zur Verwendung der XML

Es wird angenommen, daß von der XML die Rolle einer weltweit einheitlichen Sprache zum Datenaustausch Verteilter Systeme übernommen wird. Somit übernimmt sie (höchst wahrscheinlich) eine wesentliche Rolle in der zukünftigen Informationsverabeitung und bietet sich deshalb für eine Studienarbeit in diesem Rahmen an. Weiterhin ist die Möglichkeit einer Konvertierung von XML in andere Formate durch XSLT (Extensible Stylesheet Language Transformation) gebeben. Diese Art der Konvertierung ist komfortabel und äußerst flexibel. Weiterführende Literatur zum Thema XML bietet zum Beispiel Ray[2].

2.1.4 Zur Verwendung von PERL (Practical Extraction and Report Language)

Teil der Aufgabenstellung war es ebenfalls einen leightweight CLI (Command Line Interface) Client zur Suche in den Bibliotheksdaten zu implementieren. Zur Programmierung in Frage gekommen wäre jegliche Programmiersprache zu der ein Compiler für die Anwendungsplatform (in diesem Fall Linux/UNIX) existiert. Da zu fast allen

[1] http://java.sun.com/j2ee/

existierenden Hochsprachen Compiler für diese Plattformen existieren unterlag der Author bei der Wahl der Programmiersprache keinerlei Beschränkungen und konnte sich so auf die Anforderungen konzentrieren.

1. Performance

2. Geringe Komplexität

3. Ausnutzung bereits vorhandener Module (HTTP (Hyper Text Transfer Protocol), HTML (Hyper Text Markup Language), Reguläre Ausdrücke)

Besonders die reichliche Auswahl an auf das HTTP ausgerichteten, bereits vorhandenen Modulen, sowie die hervorragende Unterstützung regulärer Ausdrücke, ließ die Wahl auf PERL[2] fallen. Die Programiersprache Java, welche ebenfalls viele fertige Klassen zur Kommunikation über das HTTP bietet, disqualifizierte sich durch die, durch die Plattformunabhängigkeit entstehende, geringe Geschwindigkeit bei der Codeausführung. Weiteren weitverbereiteten Hochsprachen wie ANSI C, C++ oder Python wurde Perl aufgrund größerer Programmiererfahrung vorgezogen. Weiterführende Literatur zum Thema Perl bietet zum Beispiel Schwartz[3].

2.1.5 Zur Verwendung von HTML über das HTTP sowie des Jakarta Tomcat Servers

Eine Lösung als Webanwendung war Bestandteil der Aufgabenstellung. HTML ist der Standard für Webseiten seit dem Bestehen des WWW (World Wide Web), das eingesetzte Übertragungsprotokoll HTTP. Der Jakarta Tomcat Webserver[3] erfreut sich als kostenloser Servlet Container großer Beliebtheit und wird meist, wie auch am Institut für Betriebssysteme und Rechnerverbund, im Zusammenspiel mit dem ebenfalls kostenlosen Apache Webserver[4] eingesetzt, wobei die Stärken des letzteren bei der Erzeugung dynamischer Webseiten liegen, er aber keine Servlets ausführen kann. So ergänzen sich beide Produkte nahtlos, was ihnen weltweit große Marktanteile beschert[5]. Weiterführende Literatur zum Thema Apache Webserver bietet beispielsweise Wainwright[4]. Der Jakarta Tomcat Server wird ausführlich bei Goodwill[5] behandelt.

2.2 Überblick

2.2.1 J2EE

Die Java 2 Plattform in der Enterprise Edition definiert einen Industrie Standard zur Entwicklung von Multitier Enterprise Applikationen. J2EE macht es dem Entwicklungsteam sehr einfach solche mehrschichtigen Anwendungen zu produzieren, in dem

[2] http://www.perl.com
[3] http://jakarta.apache.org
[4] http://www.apache.org
[5] http://www.netcraft.com/Survey/

standartisierte, modulare Komponenten bereitgestellt werden, aus denen dann das Grundgerüst der verteilten Anwendung recht einfach gebaut werden kann. All diese Komponenten, in Verbindung mit den von ihnen benötigten Diensten sind in der J2EE enthalten und machen so die Entwicklung möglich ohne komplexe Programmierung zu erfordern.

Die J2EE übernimmt dabei viele Vorteile der bereits bewährten und vielseitig eingesetzten Java 2 Platform in der Standard Edition und fügt diesen die Unterstützung von JavaBeans, die Java Servlets API, Java Server Pages und die XML hinzu. Die Auslegung auf Entwicklung von multitier Anwendungen wird deutlich gemacht im J2EE application model:

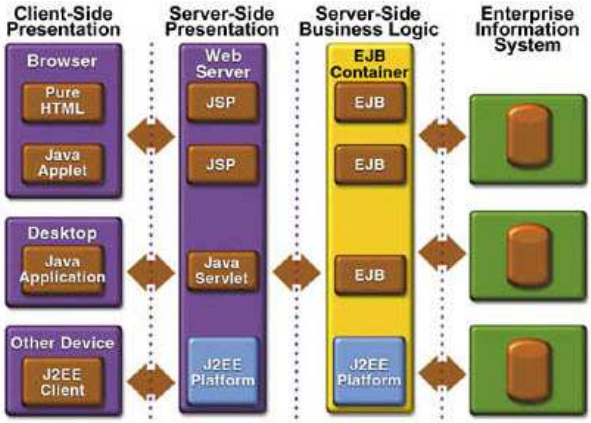

Abbildung 2.1: Das Application Model

Die Logikebene ist dabei eingebettet in Enterprise JavaBeans Komponenten, Interaktion mit dem Client kann statfinden durch einfache, statische HTML Web Seiten über Webseiten welche durch JavaServer Pages oder Servlets generiert wurden bis hin standalone Clients. Die Kommunikation erfolgt hierbei unter Einhaltung aktueller Standards wie HTML, XML und HTTP. Die Verbindung mit der Datatier erfolgt ebenfalls über standartisierte Schnittstellen wie z.B JDBC (Java Data Base Connectivity).

2.2.2 Die XML

In Zeiten, in denen Verteilte Anwendungen immer größere Bedeutung erlangen, wird eine Universalsprache für den reibungslosen Datenaustausch benötigt. Die Sprache XML dient solch einem Bearbeiten von strukturierten Daten und bleibt dennoch für Menschen lesbar, im Gegensatz zu z.B. binären Datenformaten. Dies, wie auch die Möglichkeit des syntaktischen und semantischen Validierens sowie die freie Verfügbarkeit von Werkzeugen macht sie für den universellen Einsatz äußerst interessant.

XML ist eine ganze „Familie“ von Technologien. XML 1.0 enthält die Spezifizierung der Notationsregeln. XSL ist die Sprache um Stylesheets für XML zu entwerfen und basiert auf XSLT, einer Transformationssprache, welche z.B. zum modifizieren, löschen und hinzufügen von Tags, Attributen oder komplett neuen Textstücken verwendet werden kann [6]. XML ist nicht lizensierungspflichtig, plattformunabhängig und wird von einer sehr großen Entwicklergemeinschaft ständig gepflegt und weiterentwickelt.

Definition der XML laut World Wide Web Consortium (W3C)

Die Extensible Markup Language, abgekürzt XML, beschreibt eine Klasse von Datenobjekten, genannt XML-Dokumente, und beschreibt teilweise das Verhalten von Computer-Programmen, die solche Dokumente verarbeiten. XML ist ein Anwendungsprofil (application profile) oder eine eingeschränkte Form von SGML[6], der Standard Generalized Markup Language. Durch ihre Konstruktion sind XML-Dokumente konforme SGML-Dokumente.

XML-Dokumente sind aus Speicherungseinheiten aufgebaut, genannt Entities, die entweder analysierte (parsed) oder nicht analysierte (unparsed) Daten enthalten. Analysierte Daten bestehen aus Zeichen, von denen einige Zeichendaten und andere Markup darstellen. Markup ist eine Beschreibung der Aufteilung auf Speicherungseinheiten und der logischen Struktur des Dokuments. XML bietet einen Mechanismus an, um Beschränkungen der Aufteilung und logischen Struktur zu formulieren[7].

Die Geschichte der XML

XML wurde von einer XML-Arbeitsgruppe (ursprünglich bekannt als das SGML Editorial Review Board) entwickelt, die 1996 unter der Schirmherrschaft des W3C[7] gegründet wurde. Den Vorsitz hatte Jon Bosak von Sun Microsystems inne, unter aktiver

[6] http://www.w3.org/MarkUp/SGML/
[7] http://www.w3.org

Beteiligung einer XML Special Interest Group (ehemals SGML-Arbeitsgruppe), die ebenfalls vom W3C organisiert wurde.

Die Entwurfsziele für XML waren (und sind)[8]:

1. XML soll sich im Internet auf einfache Weise nutzen lassen.
2. XML soll ein breites Spektrum von Anwendungen unterstützen.
3. XML soll zu SGML kompatibel sein.
4. Es soll einfach sein, Programme zu schreiben, die XML-Dokumente verarbeiten.
5. Die Zahl optionaler Merkmale in XML soll minimal sein, idealerweise Null.
6. XML-Dokumente sollten für Menschen lesbar und angemessen verständlich sein.
7. Der XML-Entwurf sollte zügig abgefaßt sein.
8. Der Entwurf von XML soll formal und präzise sein.
9. XML-Dokumente sollen leicht zu erstellen sein.
10. Knappheit von XML-Markup ist von minimaler Bedeutung.

2.2.3 BibTeX und LaTeX

Das weltweit verbereitete TEX Textsatzsystem, welches seit 1977 von Donald E. Knuth entwickelt wurde, fand erst durch das von Leslie Lamport entwickelte LATEX-System wirklich weite Verbreitung. Es erweitert die auf den Drucksatz von ausgerichteten Features von TEX um logische Strukturen mit denen der Autor die Möglichkeit hat, von Form und Inhalt zu abstrahieren. BibTeX ist ein Programm sowie ein Dateiformat welches von eben dieser Leslie Lamport und Oren Patashnik 1985 entworfen wurde. BibTeX erweitert LATEX um die Verwaltung von Literaturlisten. Solche Listen werden im BibTeX Format gespeichert. Mit dem Programm `bibtex` wird eine Subliste relevanter Einträge extrahiert und mit Hilfe von LATEX entsprechend einer BibTeX-Stildatei eingebunden und formatiert. Daten, welche in XML vorliegen, sind per XSLT leicht in andere Formate (in diesem Falle das BibTeX-Format) umzuwandeln, so daß im Zuge dieser Arbeit auch eine Transformationsanwendung erstellt wurde. Eine sehr gelungene Einführung zu LATEX existiert zum Beispiel von Kopka[9].

3 Alternative Lösungssysteme

3.1 Überblick

Abgesehen von der Möglichkeit das gesamte System ohne Berücksichtigung von bereits vorhandenen Technologien wie J2EE und XML zu realisieren, existieren auch bereits erfolgreich eingesetzte Lösungen. Diese reichen von Komplettlösungen bis hin zu Lösungen in Teilbereichen wie des Datenbackends oder der Präsentationsschicht.

3.2 Komplettlösungen

Es existieren bereits etliche, teils freie, teils komerzielle Informationsverwaltungssysteme die sich zum Großteil nicht auf Bibliotheken beschränken sondern eine Vielzahl an Datensätzen verwalten könen, wie z.B. Dokumente, Literatur, Adressen, Hardware, Software, CD's, etc. Viele dieser Lösungen eignen sich jedoch nicht oder nur bedingt für die angestrebte Verwaltung der Bibliotheksdaten der Institutsbibiokthek. Teils, da der Zugriff nur auf Einzelrechner Ebene erfolgen kann, teils da der Funktionsumfang sehr groß ist und dadurch hohe Komplexität und hohe Ressourcenanforderung gegeben sind. Einige wenige bieten wiederum nicht genügend Funktionsumfang um die Aufgabe zufriedenstellend zu lösen.

Stellvertretend für die kommerziellen Lösungen sollen hier i3v-Library[1] sowie das von der Universitätsbibliothek der TU-Braunschweig verwendete allegro-c[2] genannt werden. Eine freie Lösungen ist z.B. EMILE[3]. Eine große Übersicht über vorhandene Komplettlösungen bietet der entsprechende Eintrag im Google Web Directory[4].

3.3 Teillösungen

Eine Verteilte Anwendung, wie sie die in dieser Arbeit gewählte Lösung darstellt, lässt sich grob in eine 3 Schichten Architektur (3-tier-architecture) einteilen. Für jede dieser 3 Schichten existieren mehrere Lösungsmöglichkeiten.

3.3.1 FrontEnd

Die in dieser Arbeit erstellte Lösung verwendet als Benutzerschnittstelle (Client Tier) einen herkömmlichen Webbrowser. Dieser bekommt vom Webserver der Gegenseite

[1] http://www.ginit.de
[2] http://www.biblio.tu-bs.de/allegro/allegro.htm
[3] http://ginko.kfunigraz.ac.at
[4] http://directory.google.com/Top/Reference/Libraries/Library_and_Information_Science/Software/

die benötigten Daten und stellt die Eingabemasken bzw. die Ausgabe entsprechend dar. Alternative Lösungen wären z.B. ein Standalone-Client. Dieser wäre jedoch nicht (wie ein Webbrowser) auf jedem herkömmlichen Arbeitsplatz vorhanden und müsste zuerst überall installiert werden. Würde die Anwendung verändert, müsste auf jedem einzelnen Arbeitsplatz der Client ausgetauscht werden. Dieser Aufwand lässt sich durch die Wahl des Webbrowser zur Anwendungspräsentation vermeiden.

3.3.2 MiddleWare

Sämtliche Anwendungslogik findet sich bei der erarbeiteten Lösung in Servlets welche vom servlet container (in diesem Fall dem Jakarta Tomcat) bei Bedarf, also einer Anfrage eines Clients, geladen und ausgeführt werden. Für diese Schicht (Middle Tier) existieren in der J2EE, neben den bereits erwähnten Servlets, wiederum mehrere Implementierungsmöglichlkeiten. Java Server Pages (JSP) bieten die Möglichkeit Java Code in HTML Seiten einzubetten. Dieses wird bei etlichen Zeilen Code jedoch schnell unübersichtlich und eignet sich eher für kleinere Aufgaben wie z.B. das Einfügen des aktuellen Datums o.ä.. Eine weitere Möglichkeit sind Enterprise Java Beans (EJB). Diese Module bieten vom Entwickler definierte Schinttstellen an über die andere Anwendungen dann via CORBA-IIOP (Common Request Broker Architecture-Internet Inter-ORB Protocol) zur Verfügung gestellte Dienste in Anspruch nehmen können. Diese Lösung bietet sich aufgrund von Skalierbarkeit und Übersichtlichkeit vorallem bei größeren Projekten, also „echten“ Enterprise Anwendungen an, bei welchen die einmal erstellten EJB von verschiedenen anderen Anwendungen benutzt werden. Der Nachteil der EJB ist, daß sie einen Application Server benötigen, der die Systemressourcen stark in Anspruch nimmt. Auch ist die Implementierung wesentlich komplexer als bei den, sehr einfach einzusetzenden, Servlets. Die Spezifikation der J2EE Servlets findet sich auf den Webseiten von Sun Microsystems [10].
Die J2EE kann sich als einzig freie Lösung für Enterprise Applications seit ihres Erscheinens auf dem Markt behaupten. Proprietäre Produkte wie Microsofts .NET haben gegenüber der J2EE keinerlei Vorteile und scheiden somit aus Kostengründen aus. Weitere Standards wie CORBA mögen bei Sprachen wie Borlands Delphi durchaus ihre Berechtigung haben, sie sind jedoch in Verbindung mit Java seit dem Erscheinen der J2EE in reinen Java Umgebungen überflüssig.

3.3.3 BackEnd

Um die Komplexität gering zu halten wurde bei der in dieser Arbeit erstellten Lösung zur Speicherung der Anwendungsdaten (EIS Tier) eine simple Datei gewählt. Dies ist dadurch begründet, daß die Bibliotheksdaten relativ wenig Speicherplatz benötigen ($<$1 Megabyte) und somit zur Bearbeitung während der Laufzeit komplett in den Hauptspeicher des servlet container geladen werden können. Bei höherem Datenaufkommen (z.B. einer wesentlich größeren Bibliothek) ist diese Lösung sicherlich nicht mehr als praktikabel anzusehen. Alternativen wie DBS (Datenbankensysteme) gibt es in äußerst zahlreichen Varianten. Zu den erfolgreichsten gehören sicherlich Post-

greSQL[5], MySQL[6] sowie die diversen Produkte der Firmen Oracle[7] und IBM[8].
Die einzigen, zu XML in Frage kommenden, Alternativen sind das BibTeX- sowie ein eigen spezifiziertes Format. Zur komfortablen Bearbeitung sowie vorallem zur Umwandlung in HTML mit Java müssten diese Formate jedoch zuvor in XML umgewandelt werden, was die Wahl einer der Alternativen wiederum ad absurdum führen würde.

[5] http://www.postgresql.org
[6] http://www.mysql.com
[7] http://www.oracle.com
[8] http://www.ibm.com

4 Entwurf

Die im Zuge dieser Arbeit entworfene Lösung verwendet Java Servlets welche sowohl die Präsentations- als auch die Logikschicht beinhalten. Die Datenspeicherung findet in Form von XML Dateien statt. Die Servlets sind nach Aufgabe getrennt. So gibt es

- ein Servlet zur Datensatzsuche und Anzeige von Verleihinformationen (`BibSearchServlet`)
- ein Servlet zum Hinzufügen, Bearbeiten und Löschen von Datensätzen (`BibMaintainServlet`)
- ein Servlet zur Verwaltung von Verleihinformationen (`BibLendServlet`)

Weiterhin wurden ein PERL Skript welches Suchanfragen aus der Kommandozeile heraus erlaubt sowie ein Java Programm welches den in XML vorliegenden Datenbestand in das BibTeX-Format exportieren kann entworfen.

4.1 Randbedingungen

Gegebene Randbedingungen beinhalten

- das Betriebssystem Debian Linux Kernel Version 2.4
- der Webserver Apache Version 1.3
- der Webserver / Servlet Container Jakarta Tomcat Version 3.3
- die Java 2 Platform in der Standard Edition Version 1.4.0
- die Java 2 Platform in der Enterprise Edition Version 1.3
- PERL Version 5.6.1

4.2 Die Servlets

Initialisiert wird die Anwedung bei Aufruf der zugeordneten URL durch den Browser eines Benutzers. Die Initialisierung findet durch den Servlet Container Jakarta Tomcat statt. Die Initialisierung wird durch ein HTTP GET Request hervorgerufen welches im Servlet Container den Aufruf der `doGet()` Methode des entsprechenden Servlets hervorruft. Diese sendet eine eine Eingabemaske enthaltende HTML Seite zurück, die dann durch den Browser des Benutzers dargestellt wird. Nachdem der Nutzer alle

erforderlichen Eingaben getätigt hat, schickt er die Informationen an den Webserver per HTTP POST Request zurück, welcher über einen im Webformular vorhandenen Knopf ausgelöst wird. Der Webserver übergibt die Informationen dem Servlet Container als Bestandteil des HTTP Headers. Dieser führt wiederum die `doPost()` Methode des aktuellen Servlets aus. In dieser Methode wird eine HTML Ausgabeseite, entsprechend den Eingaben des Benutzers, generiert und an ihn zurückgesendet sowie gegebenenfalls der Datenbestand geändert.

4.3 UML Klassen Diagramme und Hilfsklassen

4.3.1 Übersicht

Das UML Klassendiagramm zeigt den Zusammenhang der Klassen gemäß UML Standard auf. Sämtliche Utility Klassen wurden zur besseren Übersichtlichkeit im Diagramm nicht berücksichtigt.

4.3.2 Klassen Diagramm

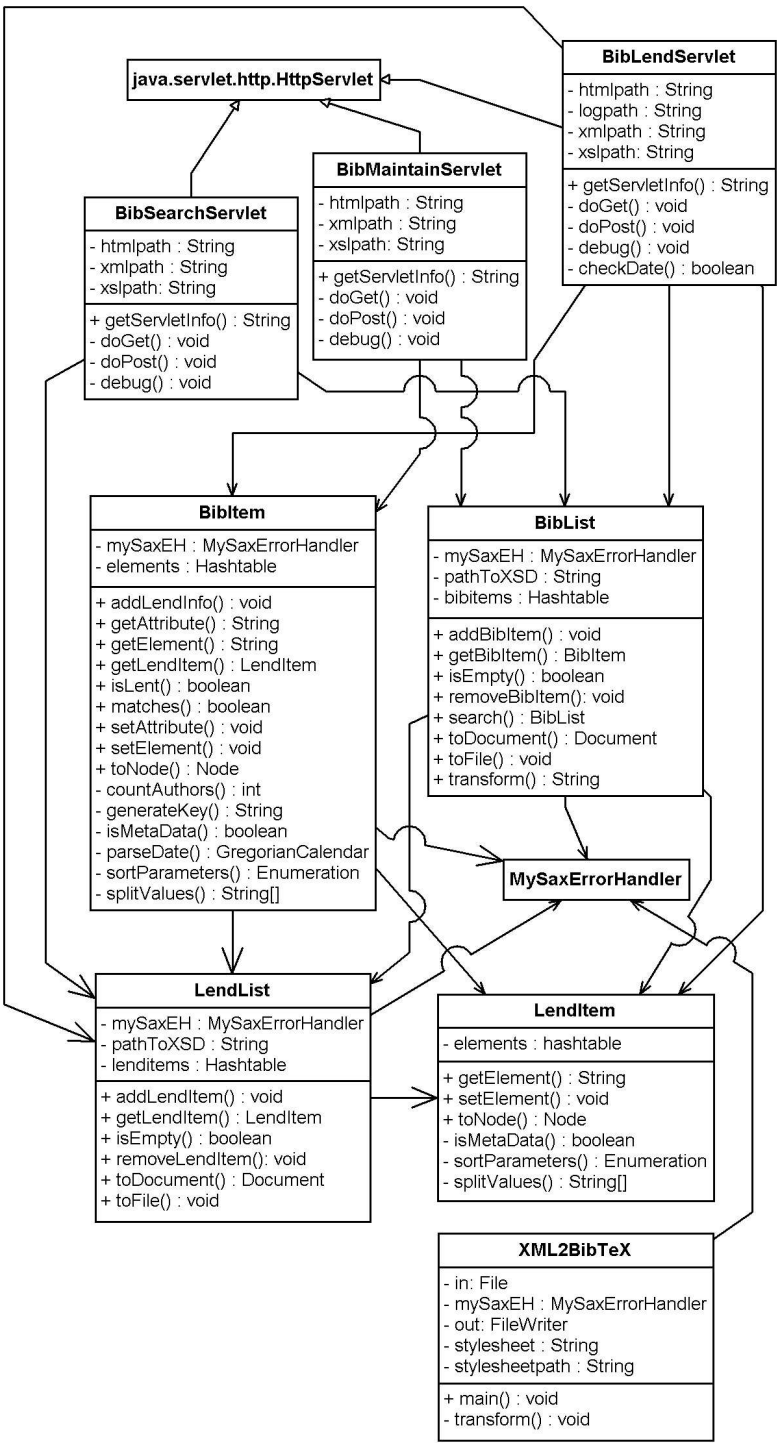

Abbildung 4.1: UML Klassen Diagramm

BibSearchServlet

Die Klasse BibSearchServlet enthält das Servlet zur Suchfunktion der Anwendung. Sie bearbeitet die HTTP Requests in dem sie den Servlet Container in Verbindung mit dem Web Server dazu veranlasst entweder Eingabemasken oder Ausgabeseiten an den Requester zurück zu senden.

BibMaintainServlet

Die Klasse BibMaintainServlet enthält das Servlet zur Datenpflegefunktion der Anwendung. Sie bearbeitet die HTTP Requests in dem sie den Servlet Container in Verbindung mit dem Web Server dazu veranlasst entweder Eingabemasken oder Ausgabeseiten an den Requester zurück zu senden.

BibLendServlet

Die Klasse BibLendServlet enthält das Servlet zur Verleihverwaltungsfunktion der Anwendung. Sie bearbeitet die HTTP Requests in dem sie den Servlet Container in Verbindung mit dem Web Server dazu veranlasst entweder Eingabemasken oder Ausgabeseiten an den Requester zurück zu senden.

BibItem

Die Klasse BibItem repräsentiert ein BibItem Element. Intern wird dies durch eine mit den Kind Elementen gefüllte Hash Tabelle dargestellt.

BibList

Die Klasse BibList repräsentiert ein BibList Element und damit sämtliche eingetragenen Werke. Intern wird dies durch eine mit BibItems gefüllte Hash Tabelle dargestellt.

LendItem

Die Klasse LendItem repräsentiert ein LendItem Element. Dieses enthält Verleihinformationen zu genau einem BibItem. Intern wird dies durch eine mit den Kind Elementen gefüllte Hash Tabelle dargestellt.

LendList

Die Klasse LendList repräsentiert ein LendList Element und damit sämtliche Verleihinformationen. Intern wird dies durch eine mit LendItems gefüllte Hash Tabelle dargestellt.

XML2BibTeX

Die Klasse XML2BibTeX enthält das Transformations Tool zur Umwandlung der XML Datei, welche die eingetragenen Werke enthält, in das BibTeX-Format.

4.4 Use Case Diagramm

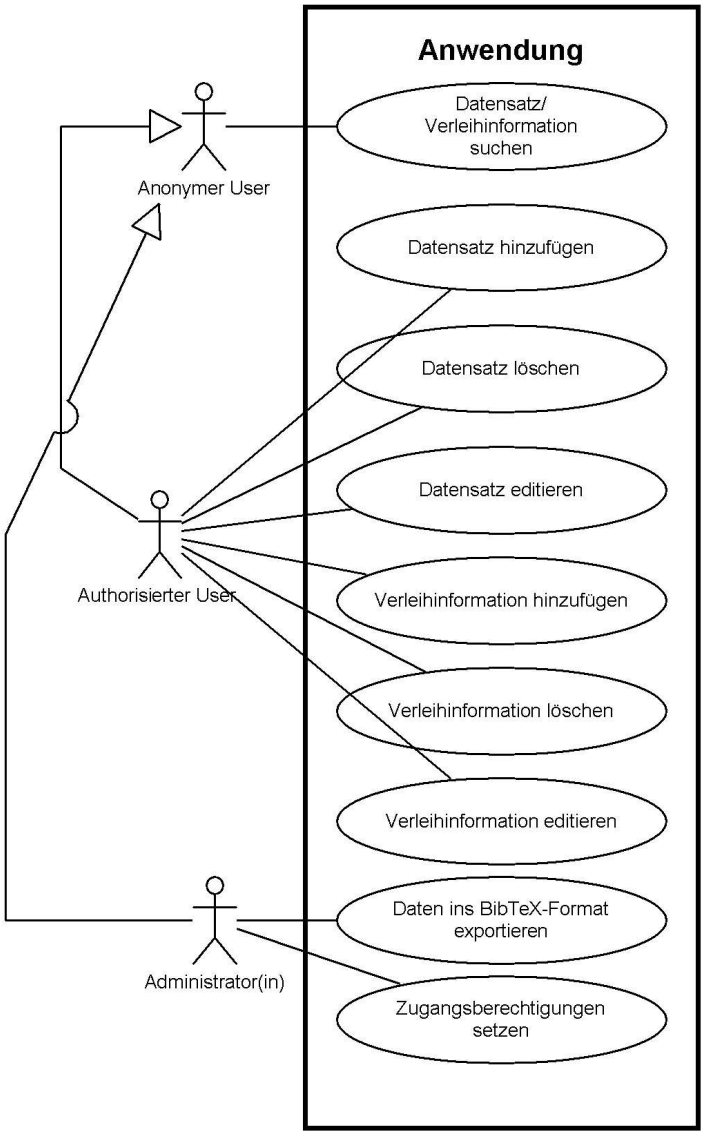

Abbildung 4.2: Das Use Case Diagramm

4.4.1 Use Cases

Folgende Use Cases existieren im System:

„Datensatz / Verleihinformation suchen“

Dieser Use Case beginnt, sobald ein Benutzer seinen Browser auf den mit dem `BibSearchServlet` verknüpften URL richtet. Daraufhin wird an den Browser eine HTML Seite mit einer Eingabemaske gesendet. Der Benutzer füllt diese aus und sendet sie zurück an das

Servlet. Dieses berechnet die Ausgabe und sendet sie, wiederum in Form einer HTML Seite an den Benutzer zurück.

„Datensatz hinzufügen“

Dieser Use Case beginnt, sobald ein Benutzer seinen Browser auf den mit dem `BibMaintainServlet` verknüpften URL richtet. Daraufhin wird an den Browser eine HTML Seite mit einer Eingabemaske gesendet. Der Benutzer füllt den für das Hinzufügen eines Datensatzes bestimmten Teil aus und sendet sie zurück an das Servlet. Dieses berechnet die Ausgabe und sendet sie, wiederum in Form einer HTML Seite an den Benutzer zurück.

„Datensatz löschen“

Dieser Use Case beginnt, sobald ein Benutzer seinen Browser auf den mit dem `BibMaintainServlet` verknüpften URL richtet. Daraufhin wird an den Browser eine HTML Seite mit einer Eingabemaske gesendet. Der Benutzer füllt den für das Suchen eines zu löschenden Datensatzes bestimmten Teil aus und sendet sie zurück an das Servlet. Dieses berechnet die Ausgabe aller gefundenen Datensätze und sendet sie, wiederum in Form einer HTML Seite, an den Benutzer zurück. Dieser drückt nun im Abschnitt des zu löschenden Datensatzes den „Löschen“-Knopf. Das Servlet berechnet die Ausgabe und sendet sie widerum an den Benutzer zurück.

„Datensatz editieren“

Dieser Use Case beginnt, sobald ein Benutzer seinen Browser auf den mit dem BibMaintainServlet verknüpften URL richtet. Daraufhin wird an den Browser eine HTML Seite mit einer Eingabemaske gesendet. Der Benutzer füllt den für das Suchen eines zu editierenden Datensatzes bestimmten Teil aus und sendet sie zurück an das Servlet. Dieses berechnet die Ausgabe aller gefundenen Datensätze und sendet sie, wiederum in Form einer HTML Seite, an den Benutzer zurück. Dieser editiert nun den entsprechenden Datensatz und drückt in dem Abschnitt des zu ändernden Datensatzes den „Absenden“-Knopf. Das Servlet berechnet die Ausgabe und sendet sie widerum an den Benutzer zurück.

„Verleihinformation hinzufügen“

Dieser Use Case beginnt, sobald ein Benutzer seinen Browser auf den mit dem BibLendServlet verknüpften URL richtet. Daraufhin wird an den Browser eine HTML Seite mit einer Eingabemaske zur Suche nach Datensätzen gesendet. Der Benutzer füllt diese aus und sendet sie zurück an das Servlet. Dieses berechnet die Ausgabe aller gefundenen Datensätze und sendet sie, wiederum in Form einer HTML Seite, an den Benutzer zurück. Dieser füllt nun für den entsprechenden Datensatz die Felder für die Verleihinformation aus und drückt den „Abschicken“-Knopf woraufhin das Servlet die Ausgabe berechnet und dem Benutzer präsentiert.

„Verleihinformation löschen“

Dieser Use Case beginnt, sobald ein Benutzer seinen Browser auf den mit dem BibLendServlet verknüpften URL richtet. Daraufhin wird an den Browser eine HTML Seite mit einer Eingabemaske zur Suche nach Datensätzen gesendet. Der Benutzer füllt diese aus aus und sendet sie zurück an das Servlet. Dieses berechnet die Ausgabe aller gefundenen Datensättze und sendet sie, wiederum in Form einer HTML Seite, an den Benutzer zurück. Dieser drückt nun für den entsprechenden Datensatz den „Zurückgegeben“-Knopf woraufhin das Servlet die Ausgabe berechnet und dem Benutzer präsentiert.

„Verleihinformation editieren“

Dieser Use Case beginnt, sobald ein Benutzer seinen Browser auf den mit dem BibLendServlet verknüpften URL richtet. Daraufhin wird an den Browser eine HTML Seite mit einer Eingabemaske zur Suche nach Datensätzen gesendet. Der Benutzer füllt diese aus aus und sendet sie zurück an das Servlet. Dieses berechnet die Ausgabe aller gefundenen Datensättze und sendet sie, wiederum in Form einer HTML Seite, an den Benutzer zurück. Dieser editiert nun bei dem entsprechenden Datensatz die Felder für die Verleihinformation und drückt den „Abschicken“-Knopf woraufhin das Servlet die Ausgabe berechnet und dem Benutzer präsentiert.

„Datenformat in das BibTeX-Format exportieren“

Dieser Use Case beginnt sobald ein Benutzer das Transformations Tool startet. Das Tool beendet und gibt evtl. Fehlermeldungen aus.

„Zugangsberechtigungen setzen“

Dieser Use Case beginnt sobald ein Benutzer Zugriffserechtigungen pflegen will. Der Benutzer macht in den dafür verantwortlichen Konfigurationsdateien des Servlet Containers entsprechende Eintragungen und startet diesen neu.

4.4.2 Aktoren

Folgende Aktoren sind Bestandteil des Systems:

„Anonymer User“

Anonyme User sind alle Benutzer, die die Applikation (anonym) über das Webinterface oder das CLI benutzen. Beispielsweise Studenten oder wissenschaftliche Mitarbeiter auf Literatursuche.

„Authorisierter User“

Authorisierte User sind alle User,die Zugriffsrechte auf die Pflege des Datenbestandes sowie die Verwaltung der Verleihinformationen haben. Beispielsweise studentische Hilfskräfte im Bibliotheksdienst.

„Administrator“

Administratoren sind Benutzer, die berechtigt sind Zugrifsrechte für die Applikation zu pflegen.

4.5 Beispiele

4.5.1 Suchanfrage

Der Web Browser des Benutzers sendet an den Web Server einen HTTP GET Request. Dieser stellt eine Verknüpfung dieses URL mit dem Servlet Container fest und leitet das Request an diesen weiter. Der Servlet Container erstellt eine neue Instanz von BibSearchServlet oder verwendet falls möglich eine bereits vorhandene und führt die `doGet()` Methode aus. BibSearchServlet erstellt eine neue, leere Instanz von BibList und nutzt deren `transform()` Funktion um durch eine XSLT eine HTML Seite mit der Eingabemaske zu erstellen. Diese wird dann über den Web Server an den Browser des Benutzers zurückgesandt. Dort von ihm ausgefüllt wird ein HTTP Post Request an den Web Server gesandt welches sämtliche eingegebene Informationen enthält. Der Web Server leitet den Request wieder an den Servlet Container weiter, welcher die `doPost()` Methode der Servlet Instanz ausführt. Diese erstellt eine neue Instanz von BibList, welche mit den Daten der XML Datei, die den Datenbestand enthält, gefüllt wird. Danach wird die `search()` Methode des BibList Objektes aufgerufen, welche die Liste nach Datensätzen durchsucht, die auf die im HTTP Requests enthaltenen Parameter passen. Das Ergebnis wird in Form einer weiteren BibList zurückgegeben und per `transform()` Methode in HTML Umgewandelt und an den Browser zurückgesandt.

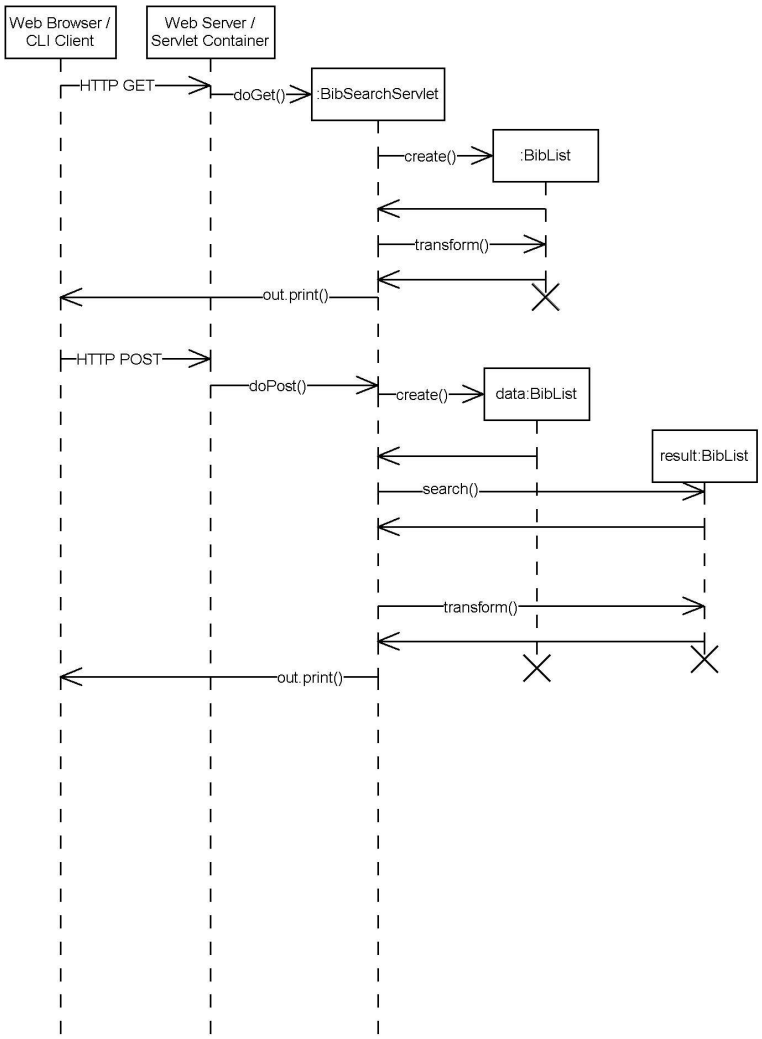

Abbildung 4.3: Sequenz Diagramm Suchanfrage

4.5.2 Werk hinzufügen

Der Web Browser des Benutzers sendet an den Web Server ein HTTP GET Request. Dieser stellt eine Verknüpfung dieses URL mit dem Servlet Container fest und leitet den Request an diesen weiter. Der Servlet Container erstellt eine neue Instanz von BibMaintainServlet oder verwendet falls möglich eine bereits vorhandene und führt die `doGet()` Methode des Servlets aus. BibMaintainServlet erstellt eine neue, leere Instanz von BibList und nutzt deren `transform()` Funktion um durch eine XSL Transformation eine HTML Seite mit der Eingabemaske zu erstellen. Diese wird dann über den Web Server an den Browser des Benutzers zurückgesandt. Dort von ihm ausgefüllt wird ein HTTP Post Request an den Web Server gesandt, welches im HTTP Header sämtliche eingegebene Informationen enthält. Der Web Server leitet den Re-

quest wieder an den Servlet Container weiter, welcher die `doPost()` Methode der Servlet Instanz ausführt. Nun werden zwei Instanzen von BibList angelegt. Eine gefüllt mit dem Datenbestand aus der XML Datei, die andere leer. Aus den im HTTP Request vorhandenen Informationen wird jetzt eine Instanz von BibItem erzeugt, welche dann der leeren BibList hinzugefügt wird. Das key-Attribut dieses neuen BibItem wird wie folgt automatisch generiert. Ist das `author` Feld nicht vorhanden oder leer wird der Schlüssel aus dem Prefix `ID` sowie dem Suffix „Vergangene Zeit seit dem 1.1.1970 in Millisekunden“ generiert. Ist genau ein Author angegeben setzt sich der Schlüssel aus dem Prefix „die ersten drei Buchstaben des Nachnamen des Authors“ sowie dem Suffix „die letzten zwei Stellen des angegebenen `year` Feldes“ zusammen. Sind zwei oder mehrere Authoren angebenen wird der Schlüssel aus dem Prefix „Anfangsbuchstaben der Nachnamen der Authoren aneinander gereiht“ sowie dem vorangehenden Suffix erzeugt. Ist das `year` Feld nicht vorhanden oder leer, so wird die Jahresangabe durch `XX` ersetzt. Die mit dem neuen BibItem erweiterte BibList wird nun mit der `transform()` Methode in das HTML-Format konvertiert und an den Benutzer zurückgesandt. Der Benutzer sieht das automatisch generierte key-Attribut mit dem das Werk abgespeichert werden soll sowie die restlichen Felder des BibItems und bestätigt daraufhin seine Eingaben und den Schlüssel. Damit wird vom Browser ein weiterer HTTP POST Request gesandt, welcher auf bekanntem Weg zur Ausführung der `doPost()` Methode der Servlet Instanz führt. Nun wird eine Instanz der Klasse File erstellt, welche die XML Datei repräsentiert. Weiterhin werden wieder zwei Instanzen von BibList erstellt, wiederum die eine gefüllt mit dem Datenbestand und die andere leer. Aus den im HTTP Request vorhandenen Informationen wird ein BibItem instanziiert und der bereits gefüllten BibList hinzugefügt. Diese wird dann per `toFile()` Methode in das File Objekt geschrieben und anschliessend gespeichert. Nun wird das BibItem auch zu der noch leeren BibList hinzugefügt, welche dann in das HTML-Format transformiert wird und als Ergebnis an den Browser des Benutzers zurückgesandt wird.

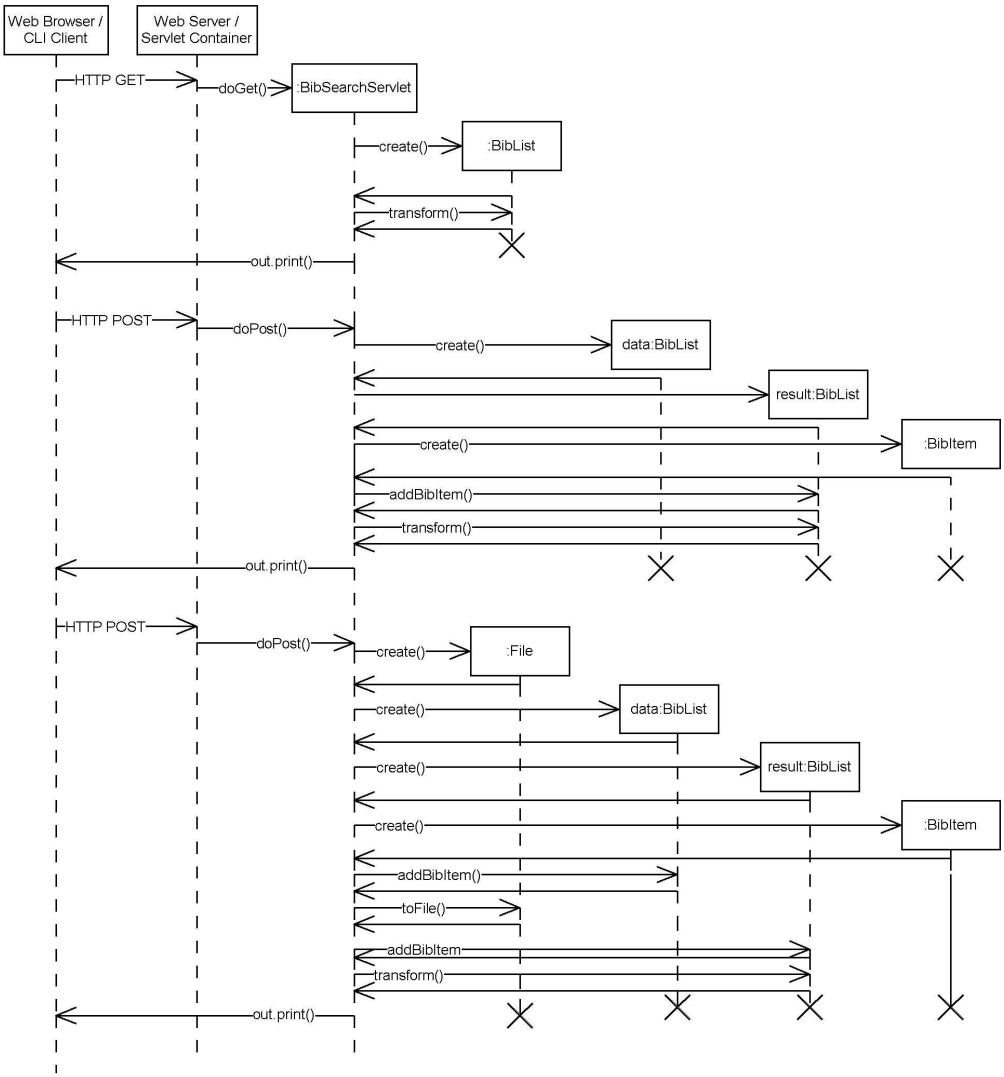

Abbildung 4.4: Sequenz Diagramm Werk hinzufügen

4.6 Das Protokoll

Nach dem Ausfüllen einer Eingabemaske durch den Benutzer wird diese von ihm per Knopfdruck an den Webserver und damit an das entsprechende Servlet geschickt. Bestandteil dieses HTTP Requests sind nun auch die Eingaben, welche zuvor in der Maske gemacht wurden. Diese Eingaben existieren im HTTP Request als Wortpaare in der Form `<Attributname>=<Attributwert>`. Mehre dieser Paare werden durch ein & miteinander verbunden. Es werden drei verschiedene Arten von Attributen übertragen. Zum einen die eigentlichen Eingaben. Sonderzeichen werden dabei durch den Browser automatisch in ihre zugehörigen Hexwerte umgewandelt. Beispielsweise wird ein Leerzeichen zu `%32` umgewandelt.

Hat das HTML Formular einen Eintrag
`<INPUT type="text" length="100" name="address" value="{address}" />`
und hat der Benutzer in das entsprechende Feld den Eintrag `Pockelstr. 1` gemacht, beinhaltet der HTTP Request das Paar `adress=Pockelstr.%321`.
Die zweite Art von Attribut bezieht sich auf die Suchlogik. Diese Attribute haben immer den Namen des Feldes auf das sie sich beziehen, erweitert durch ein „bool“. Diese Suchlogikattribute können die Werte `AND`, `OR`, `<`, `<=`, `=`, `>=` oder `>` haben. Hat das HTML Formular einen Eintrag
`<SELECT name="adressbool"><OPTION>AND</OPTION><OPTION>OR</OPTION></SELECT>`
und hat der Benutzer die Option `AND` gewählt, beinhaltet der HTTP Request das Paar `adressbool=AND`. Zusammengefasst würde der Beispiel Request bis jetzt also `adress=Pockelstr.%321&adressbool=AND` beinhalten.
Attribute der dritten Art sind im HTML Formular versteckte („hidden“) Attribute. Sie werden benötigt um die Eingabemaske und den gedrückten Knopf zu identifizieren. Das die Eingabemaske identifizierende Attribut hat dabei immer den Namen `requester`, das den Knopf identifizierende den Namen `button`. Hat das HTML Formular einen Eintrag
`<INPUT type="submit" name="button" value="Bestaetigen"/>`
`<INPUT type="hidden" name="requester" value="add"/>`
beinhaltet der HTTP Request die Paare `button=Bestaetigen&requester=add`.

4.7 XML Schemata

Ein XML Schema anthält die Grammatik eines XML Dokumentes.

4.7.1 Das Schema der BibList.xml Datei

Das Schema legt den Aufbau der Datei folgendermaßen fest:

- Wurzelelement ist immer ein BibList Element welches aus mindestens einem BibItem Element besteht.

```
<xs:element name="biblist">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="bibitem"
                            type="bibitemType"
                            maxOccurs="unbounded"
                />
            </xs:sequence>
        </xs:complexType>
</xs:element>
```

- Ein BibItem Element hat genau zwei Attribute. Zum einen `key`, welches einen eindeutigen Schlüssel enthält und zum anderen `type`, in welchem die Art des Werkes (z.B. article, book, manual, etc.) zu finden ist. Beide Attribute sind obligatorisch. Weiterhin besteht ein BibItem aus mehreren Kind Elementen welche wiederum nur Text enthalten können. Diese Kind Elemente kommen genau einmal oder garnicht vor. Die Reihenfolge ist beliebig.

```
<xs:complexType name="bibitemType">
        <xs:all>
            <xs:element ref="address"/>
            <xs:element ref="annote"/>
            <xs:element ref="author"/>
            <xs:element ref="booktitle"/>
            <xs:element ref="chapter"/>
            <xs:element ref="crossref"/>
            <xs:element ref="descriptor"/>
            <xs:element ref="edition"/>
            <xs:element ref="editor"/>
            <xs:element ref="howpublished"/>
            <xs:element ref="institution"/>
            <xs:element ref="isbn"/>
            <xs:element ref="journal"/>
            <xs:element ref="key"/>
            <xs:element ref="month"/>
            <xs:element ref="note"/>
            <xs:element ref="number"/>
            <xs:element ref="organization"/>
            <xs:element ref="pages"/>
            <xs:element ref="publisher"/>
            <xs:element ref="school"/>
            <xs:element ref="series"/>
            <xs:element ref="title"/>
            <xs:element ref="type"/>
            <xs:element ref="url"/>
            <xs:element ref="volume"/>
            <xs:element ref="year"/>
        </xs:all>
        <xs:attribute name="key" type="xs:string" use="required"/>
        <xs:attribute name="type" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="Article"/>
                    <xs:enumeration value="Book"/>
                    <xs:enumeration value="Booklet"/>
                    <xs:enumeration value="InBook"/>
                    <xs:enumeration value="InCollection"/>
                    <xs:enumeration value="InProceedings"/>
```

```
                    <xs:enumeration value="Manual"/>
                    <xs:enumeration value="MastersThesis"/>
                    <xs:enumeration value="Misc"/>
                    <xs:enumeration value="PhdThesis"/>
                    <xs:enumeration value="Proceedings"/>
                    <xs:enumeration value="TechReport"/>
                    <xs:enumeration value="Thesis"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
</xs:complexType>
```

- Weiterhin sind in dem Schema die Namen aller möglichen BibItem Kind Elemente aufgeführt. Im wesentlichen handelt es sich um die Namen der auch im BibTeX-Format vorkommenden Datenfelder. Alle Felder die nicht zum BibTeX-Standard gehören wurden aufgenommen da sie Teil des Datenaltbestandes des Institutes waren. Es handelt sich bei ihnen um sinvolle Ergänzungen zum BibTeX-Standard.

  `<xs:element name="address" type="xs:string"/>`
  Normalerweise die Adresse des Verlegers oder Herausgebers. Gehört zum BibTeX-Standard.

  `<xs:element name="annote" type="xs:string"/>`
  Eine Anmerkung. Gehört zum BibTeX-Standard.

  `<xs:element name="=author" type="xs:string"/>`
  Der Name des Authors (der Authoren) im LaTeX üblichen Format. Gehört zum BibTeX-Standard.

  `<xs:element name="booktitle" type="xs:string"/>`
  Der Titel des Buches. Gehört zum BibTeX-Standard.

  `<xs:element name="chapter" type="xs:string"/>`
  Eine Kapitel- (oder Sektions- ö.ä.) Nummer. Gehört zum BibTeX-Standard.

  `<xs:element name="crossref" type="xs:string"/>`
  Der Datenbankschlüssel des Eintrages der crossreferred wird. Gehört zum BibTeX-Standard.

  `<xs:element name="descriptor" type="xs:string"/>`
  Kurze Beschreibung des Werkes. Gehört nicht zum BibTeX-Standard.

  `<xs:element name="edition" type="xs:string"/>`
  Die Auflage des Werkes. Gehört zum BibTeX-Standard.

  `<xs:element name="editor" type="xs:string"/>`

Der Name des Lektors des Werkes. Gehört zum BibTeX-Standard.

`<xs:element name="howpublished" type="xs:string"/>`
Beschreibung wie das Werk veröffentlicht wurde. Gehört zum BibTeX-Standard.

`<xs:element name="institution" type="xs:string"/>`
Der Name der protegierenden Institution eines Fachberichts. Gehört zum BibTeX-Standard.

`<xs:element name="isbn" type="xs:string"/>`
Die ISBN des Werkes. Gehört nicht zum BibTeX-Standard.

`<xs:element name="journal" type="xs:string"/>`
Der Name des das Werk enthaltende Journals. Gehört zum BibTeX-Standard.

`<xs:element name="key" type="xs:string"/>`
Information die zum sortieren und cross referencing benutzt wird wenn das author Feld leer ist. Gehört zum BibTeX-Standard.

`<xs:element name="month" type="xs:string"/>`
Der Monat in dem das Werk veröffentlicht wurde oder, bei einem unveröffentlichtem Werk, der Monat in dem es geschrieben wurde. Gehört zum BibTeX-Standard.

`<xs:element name="note" type="xs:string"/>`
Jegliche zusätzliche Information, die für den Leser hilfreich sein könnte. Gehört zum BibTeX-Standard.

`<xs:element name="number" type="xs:string"/>`
Die Nummer des Journals oder Magazins, in dem das Werk erschienen ist.Gehört zum BibTeX-Standard.

`<xs:element name="organization" type="xs:string"/>`
Der Name einer Organisation, die als Sponsor bei einer Konferenz auftritt oder ein Handbuch herausgibt. Gehört zum BibTeX-Standard.

`<xs:element name="pages" type="xs:string"/>`
Die Seiten in einem Buch, Journal o.ä.. Gehört zum BibTeX-Standard.

`<xs:element name="publisher" type="xs:string"/>`
Der Name des Verlegers oder Herausgebers. Gehört zum BibTeX-Standard.

`<xs:element name="school" type="xs:string"/>`
Der Name der Lehranstalt,in welcher das Werk geschrieben wurde. Gehört zum BibTeX-Standard.

`<xs:element name="series" type="xs:string"/>`

Der Name der Veröffentlichungsserie. Gehört nicht zum BibTeX-Standard.

`<xs:element name="title" type="xs:string"/>`
Der Titel des Werkes. Gehört zum BibTeX-Standard.

`<xs:element name="type" type="xs:string"/>`
Der Typ eines Fachberichtes. Gehört zum BibTeX-Standard.

`<xs:element name="url" type="xs:string"/>`
Der URL unter dem das Werk abrufbar ist. Gehört nicht zum BibTeX-Standard.

`<xs:element name="="volume" type="xs:string"/>`
Der Jahrgang einer Zeitschrift. Gehört zum BibTeX-Standard.

`<xs:element name="year" type="xs:string"/>`
Das Jahr in dem das Werk veröffentlicht wurde oder, bei einem unveröffentlichtem Werk, das Jahr in dem es geschrieben wurde. Gehört zum BibTeX-Standard.

Das vollständige Schema ist im Anhang zu finden.

4.7.2 Das Schema der LendList.xml Datei

Das Schema legt den Aufbau der Datei folgendermaßen fest:

- Wurzelelement ist immer ein LendList Element welches aus beliebig vielen LendItem Element besteht

```
<xs:element name="lendlist">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="lenditem"
                            type="lenditemType"
                            minOccurs="0"
                            maxOccurs="unbounded"
                />
            </xs:sequence>
        </xs:complexType>
</xs:element>
```

- Ein BibItem besteht aus mehreren Kind Elementen welche jeweils genau einmal vorkommen. Die Reihenfolge ist beliebig

```
<xs:complexType name="lenditemType">
        <xs:all minOccurs="1">
            <xs:element ref="key"/>
```

```
            <xs:element ref="lentbymatno"/>
            <xs:element ref="lentbyname"/>
            <xs:element ref="lenddate"/>
        </xs:all>
    </xs:complexType>
```

- Weiterhin sind in der DTD die Namen aller möglichen BibItem Kind Elemente aufgeführt.

  `<xs:element name="lentbyname" type="xs:string"/>`
  Der Name des Ausleihers.

  `<xs:element name="lentbymatno" type="xs:string"/>`
  Die Matrikelnummer des Ausleihers.

  `<xs:element name="lenddate" type="xs:string"/>`
  Das Datum an dem ausgeliehen wurde.

  `<xs:element name="key" type="xs:string"/>`
  Der Wert des key-Attributes des geliehenen BibItems.

Die gesamte DTD ist im Anhang zu finden.

4.8 Zur Sicherheit

Die Sicherheit wird bei dieser Lösung „Container Based“ geregelt. Das heißt der Servlet Container übernimmt sämtliche Sicherheitsfunktionen in dem der Zugriff zu den Verwaltungsservlets durch eine Eingabeaufforderung zu Username und Passwort geschützt ist. Hierzu legt der Administrator in der `tomcat-users.xml` Konfigurationsdatei des Jakarta Tomcat Servers zunächst Benutzer an. Dies geschieht beispielsweise über folgenden Eintrag:

```
<tomcat-users>
  <user name="bib" password="pass" roles="bibrole" />
</tomcat-users>
```

Damit wird ein neuer Benutzer mit dem Benutzernamen *bib* und dem Passwort *pass* angelegt. Dieser neue Benutzer hat die Rolle („role“) *bibrole*. Anschließend muss die `web.xml` Konfigurationsdatei des Jakarta Tomcat Servers (der Web Application Deployment Descriptor) angepasst werden. Folgende Eintragungen sind zu machen:

```
<web-app>
    <servlet>
        <servlet-name>BibSearchServlet</servlet-name>
        <servlet-class>BibSearchServlet</servlet-class>
    </servlet>
```

```
<servlet>
    <servlet-name>BibMaintainServlet</servlet-name>
    <servlet-class>BibMaintainServlet</servlet-class>
</servlet>

<servlet>
    <servlet-name>BibLendServlet</servlet-name>
    <servlet-class>BibLendServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>BibSearchServlet</servlet-name>
    <url-pattern>/BibSearchServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>BibMaintainServlet</servlet-name>
    <url-pattern>/BibMaintainServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>BibLendServlet</servlet-name>
    <url-pattern>/BibLendServlet</url-pattern>
</servlet-mapping>

<security-constraint>
    <web-resource-collection>
        <web-resource-name>
            SecretProtection
        </web-resource-name>
        <url-pattern>
            /BibMaintainServlet
        </url-pattern>
        <http-method>
            GET
        </http-method>
        <http-method>
            POST
        </http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>
            bibrole
        </role-name>
    </auth-constraint>
</security-constraint>
```

```
    <login-config>
        <auth-method>
            BASIC
        </auth-method>
        <realm-name>
            Default
        </realm-name>
    </login-config>
    <security-role>
        <role-name>
            bibrole
        </role-name>
    </security-role>
</web-app>
```

Das `<web-app>` Element dient hierbei als Wurzelelement welches sämtliche Konfigurationen beinhaltet. In den `<servlet>` Elementen werden Servlets definiert. Es wird ein (freiwählbarer) Name für das Servlet sowie der Name der Java-Klassen-Datei angegeben. In den `<servlet-mapping>` Elementen werden die URLs, über die die Servlets angesprochen werden, festgelegt. Die Servlets können nun direkt über den URL *http://servername/ServletName* angesprochen werden. Im `<security-constraint>` Element wird nun schließlich die Zugangsbeschränkung festgelegt. Dabei wird eine zum Zugang berechtigte Benutzergruppe (alle Benutzer mit der angegebenen Rolle) sowie eine Webressource angegeben. Die Webressource besteht dabei aus einem ihr gegebenen Namen, dem URL und den zu beschränkenden HTTP Request Methoden. Bei der Login Konfiguration im `<login-config>` Element wird die Art der Authentifizierung (`BASIC`, `DIGEST`, `FORM` oder `CLIENT-CERT`; Die `BASIC` Authentifizierung wird im Zusammenspiel mit einem SSL-fähigen Web-Server als ausreichend angesehen) sowie das `realm` angegeben. Das `realm` ist dabei eine Menge von Benutzernamen und Passwörtern (hier das `realm` *default* und damit auch der Benutzer der zu Beginn angelegt wurde). Zuletzt wird noch die Rolle `bibrole` als Sicherheits-Rolle festgelegt, so daß sie im `<auth-constraint>` Element verwendet werden kann.

Versucht nun ein Benutzer auf die zugangsbeschränkten Servlets zuzugreifen, muss er sich mit Benutzername und Passwort authentifizieren. Gelingt ihm das nicht sendet der Webserver einen `403 Access Denied` Fehler zurück.

Sinnvollerweise werden Benutzer und roles zum Verwaltungsdienst (BibMaintainServlet) und Verleihdienst (BibLendServlet) angelegt. Das BibSearchServlet bleibt unbeschränkt.

4.9 Das Perl Skript

Das Skript verfügt über die selbe Funktionalität wie das Webinterface. Die Syntax des Komanndozeilenargumente ist auf der entsprechenden MAN-Page dokumentiert (sie-

he Anhang).

Eine beispielhafte Suche würde wie folgt ausgeführt werden. Der Aufruf
`perl BibSeachCLI.pl author=Fischer#Strauß`
findet alle Datensätze mit den Authoren „Fischer“ oder „Strauß“. Ob weitere Authoren an dem Werk beteiligt sind bleibt unbeachtet. Auch werden Werke gefunden an denen sowohl „Fischer“ als auch „Strauß“ als Author aufgeführt werden. Groß- und Kleinschreibung wird bei dieser Anfrage nicht geprüft.

Der Aufruf
`perl BibSeachCLI.pl -i author=Fischer&Strauß year<1999`
findet alle Datensätze bei denen „Fischer“ und „Strauß“ als Authoren angegeben sind, und bei denen das angebene Jahr vor 1999 liegt. Ob weitere Authoren an dem Werk beteiligt sind bleibt unbeachtet, Groß- und Kleinschreibung wird jedoch geprüft.

Eine ausschließende Abfrage (beispielsweise alle Datensätze mit dem Author „Fischer“ jedoch ohne den Author „Strauß“) ist zum Zeitpunkt der Fertigstellung dieser Arbeit noch nicht implementiert. Diese Funktionalität muß serverseitig ergänzt werden, so daß auch das Webinterface davon betroffen ist.

4.10 Der Export ins BibTeX-Format

Das Tranformations Tool exportiert den Datenbestand der XML Datei in das BibTeX-Format. Hierbei wird die die Daten enthaltende XML Datei eingelesen und daraus in ein Java DOM (Document Object Model)[11] document erstellt. Dieses stellt Methoden zur Transformierung mittels XSLT zur Verfügung. Nach der Transformierung wird das Ergebnis in einer (vorher angegebenen) Datei gespeichert.

Die Transformierung mittels XSLT findet dabei wie folgt statt.

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
  >
  <xsl:output method="text" />
  <xsl:strip-space elements="bibitem" />
```

Zunächst wird die Ausgabe als text festgelegt und alle überflüssigen Leerzeichen entfernt.

```
  <xsl:template match="/">
     <xsl:apply-templates />
  </xsl:template>

  <xsl:template match="/biblist/bibitem">
    <xsl:text>@</xsl:text><xsl:value-of select="@type" />
    <xsl:text>{</xsl:text><xsl:value-of select="@key" />
```

```
    <xsl:apply-templates />
    <xsl:text>
}

</xsl:text>
  </xsl:template>
```

Alle BibItem Elemente werden abgearbeitet und für jedes neue BibItem wird in den Ausgabestrom `@type{key<zeilenumbruch>` geschrieben, wobei `type` den Wert des type-Attributes des aktuellen BibItems hat und `key` den entsprechenden key-Attribut Wert.

```
  <xsl:template match="/biblist/bibitem/*">
      <xsl:text>,
      </xsl:text><xsl:value-of select="name()" />
      <xsl:text> = {</xsl:text><xsl:value-of select="." />
      <xsl:text>}</xsl:text>
      <xsl:apply-templates select="address|annote|author|
                                   booktitle|chapter|crossref|
                                   descriptor|edition|editor|
                                   howpublished|institution|
                                   isbn|journal|key|month|note|
                                   number|organization|pages
                                   publisher|title|type|series|
                                   school|year|volume|url|"
      />
  </xsl:template>

  <xsl:template match="text()">
    <xsl:value-of select="normalize-space()" />
  </xsl:template>

</xsl:stylesheet>
```

Für jedes im aktuellen BibItem enthaltene Kindelement wird in den Ausgabestrom `<attributname> = {<attributwert>}<zeilenumbruch>` geschrieben. So wird aus der XML Notation

```
<bibitem key="Bur95" type="Article">
   <pages>309-337</pages>
   <year>1995</year>
   <author>M. Burgess</author>
   <number>3</number>
   <volume>8</volume>
   <journal>Computing Systems</journal>
   <title>A Site Configuration Engine</title>
</bibitem>
```

die BibTeX Notation

```
@Article{Bur95,
      pages = {309-337},
      year = {1995},
      author = {M. Burgess},
      number = {3},
      volume = {8},
      journal = {Computing Systems},
      title = {A Site Configuration Engine}
}
```

gemacht. Weiterhin wandelt das Programm alle Sonderzeichen in LaTeX konforme Zeichenketten um.

Aufgerufen wird XML2BibTeX mit zwei Kommandozeilenparametern. Zum einen den Namen der XML Datei und zum anderen den Namen der auszugebenden Datei. Beispielsweise liest das Kommando
`java XML2BibTeX ibr.xml ibr.bib` die Datei `ibr.xml` ein und schreibt die daraus gewonnenen BibTeX-Daten in die Datei `ibr.bib`.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Die im Zuge dieser Arbeit erstellte Anwendung löst die Aufgabe der Bibliotheksverwaltung mittels Java Servlets im Zusammenspiel mit Servlet Container und Web Server. Weiterhin wurde ein CLI Client zur Suche in den Bibliotheksdaten über die Kommandozeile implementiert. Der Export der XML Daten in das BibTeX-Format wurde mittels eigenständigem Java Programm gelöst.

Die XML hat sich als äußerst flexibel und hervorragend für diese Aufgabe geeignet erwiesen. Die Java 2 Plattform bietet alle notwendigen Werkzeuge sowie vieles darüber hinaus. Der Jakarta Tomcat Server bietet sehr viele Einstellungsmöglichkeiten, ist dadurch für Laien aber eher schwer zu konfigurieren.

Die Implementierung verlief größtenteils problemlos. Als eine besondere Schwierigkeit erwiesen sich lediglich die verschiedenen Notationen der Sonderzeichen in HTML, BibTeX und Klartext sowie die zuerst eher mäßige Performance der Servlets, welche jedoch durch Optimierung des Programmcodes gesteigert werden konnte.

5.2 Ausblick

Die erstellte Anwendung könnte auf vielerlei Arten erweitert bzw. angepasst werden. Denkbar sind z.B. RMI Schnittstellen oder das hinzufügen weiterer Parameter des HTTP Requests wie z.B. die Wahl einer anderen Datenquelle. Auch die Art der Datenquelle könnte, bei stark zunehmenden Datenbestand oder stark zunehmender Nutzung, in eine leistungsfähigere Datenbank geändert werden. Denkbar ist sicherlich auch eine Anpassung der Anwendung an andere Institutswebseiten und Datenbestände.

A Abkürzungsverzeichnis

CLI	Command Line Interface
CORBA	Common Request Broker Architecture
DBS	Datenbankensystem
DOM	Document Object Model
DTD	Document Type Definition
EIS	Enterprise Information System
EJB	Enterprise Java Bean
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IIOP	Internet Inter-ORB Protocol
J2EE	Java 2 Enterprise Edition
JDBC	Java Data Base Connectivity
JSP	Java Server Pages
SGML	Standard Generalized Markup Language
UML	Unified Modeling Language
URL	Unified Ressource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

B Anhang

B.1 XML Schemata

B.1.1 biblist.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
>
    <xs:element name="address" type="xs:string"/>
    <xs:element name="annote" type="xs:string"/>
    <xs:element name="author" type="xs:string"/>
    <xs:complexType name="bibitemType">
        <xs:all>
            <xs:element ref="address" minOccurs="0"/>
            <xs:element ref="annote" minOccurs="0"/>
            <xs:element ref="author" minOccurs="0"/>
            <xs:element ref="booktitle" minOccurs="0"/>
            <xs:element ref="chapter" minOccurs="0"/>
            <xs:element ref="crossref" minOccurs="0"/>
            <xs:element ref="descriptor" minOccurs="0"/>
            <xs:element ref="edition" minOccurs="0"/>
            <xs:element ref="editor" minOccurs="0"/>
            <xs:element ref="howpublished" minOccurs="0"/>
            <xs:element ref="institution" minOccurs="0"/>
            <xs:element ref="isbn" minOccurs="0"/>
            <xs:element ref="journal" minOccurs="0"/>
            <xs:element ref="key" minOccurs="0"/>
            <xs:element ref="month" minOccurs="0"/>
            <xs:element ref="note" minOccurs="0"/>
            <xs:element ref="number" minOccurs="0"/>
            <xs:element ref="organization" minOccurs="0"/>
            <xs:element ref="pages" minOccurs="0"/>
            <xs:element ref="publisher" minOccurs="0"/>
            <xs:element ref="school" minOccurs="0"/>
            <xs:element ref="series" minOccurs="0"/>
            <xs:element ref="title" minOccurs="0"/>
            <xs:element ref="type" minOccurs="0"/>
            <xs:element ref="url" minOccurs="0"/>
            <xs:element ref="volume" minOccurs="0"/>
```

```
            <xs:element ref="year" minOccurs="0"/>
        </xs:all>
        <xs:attribute name="key"
                      type="xs:string"
                      use="required"
        />
        <xs:attribute name="type" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="Article"/>
                    <xs:enumeration value="Book"/>
                    <xs:enumeration value="Booklet"/>
                    <xs:enumeration value="InBook"/>
                    <xs:enumeration value="InCollection"/>
                    <xs:enumeration value="InProceedings"/>
                    <xs:enumeration value="Manual"/>
                    <xs:enumeration value="MastersThesis"/>
                    <xs:enumeration value="Misc"/>
                    <xs:enumeration value="PhdThesis"/>
                    <xs:enumeration value="Proceedings"/>
                    <xs:enumeration value="TechReport"/>
                    <xs:enumeration value="Thesis"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="biblist">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="bibitem"
                            type="bibitemType"
                            maxOccurs="unbounded"
                />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="booktitle" type="xs:string"/>
    <xs:element name="chapter" type="xs:string"/>
    <xs:element name="crossref" type="xs:string"/>
    <xs:element name="descriptor" type="xs:string"/>
    <xs:element name="edition" type="xs:string"/>
    <xs:element name="editor" type="xs:string"/>
    <xs:element name="howpublished" type="xs:string"/>
    <xs:element name="institution" type="xs:string"/>
    <xs:element name="isbn" type="xs:string"/>
    <xs:element name="journal" type="xs:string"/>
    <xs:element name="key" type="xs:string"/>
```

```
    <xs:element name="month" type="xs:string"/>
    <xs:element name="note" type="xs:string"/>
    <xs:element name="number" type="xs:string"/>
    <xs:element name="organization" type="xs:string"/>
    <xs:element name="pages" type="xs:string"/>
    <xs:element name="publisher" type="xs:string"/>
    <xs:element name="school" type="xs:string"/>
    <xs:element name="series" type="xs:string"/>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="type" type="xs:string"/>
    <xs:element name="url" type="xs:string"/>
    <xs:element name="volume" type="xs:string"/>
    <xs:element name="year" type="xs:short"/>
</xs:schema>
```

B.1.2 lendlist.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
>
    <xs:element name="key" type="xs:string"/>
    <xs:element name="lenddate" type="xs:string"/>
    <xs:complexType name="lenditemType">
        <xs:all minOccurs="1">
            <xs:element ref="key"/>
            <xs:element ref="lentbymatno"/>
            <xs:element ref="lentbyname"/>
            <xs:element ref="lenddate"/>
        </xs:all>
    </xs:complexType>
    <xs:element name="lendlist">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="lenditem"
                            type="lenditemType"
                            minOccurs="0"
                            maxOccurs="unbounded"
                />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="lentbymatno" type="xs:string"/>
    <xs:element name="lentbyname" type="xs:string"/>
</xs:schema>
```

Literaturverzeichnis

[1] D. FLANAGAN, J. FARLEY, W. CRAWFORD: *Java Enterprise in a Nutshell*. O'Reilly, 2002.

[2] RAY, E. T.: *Learning XML*. O'Reilly, 2001.

[3] R. L. SCHWARTZ, T. PHOENIX: *Learning Perl*. O'Reilly, 2001.

[4] P. WAINWRIGHT, A. AHMAD, M. LINK: *Professional Apache 2.0*. Wrox Press Inc., 2002.

[5] GOODWILL, J.: *Apache Jakarta-Tomcat*. APress, 2001.

[6] *http://www.w3.org/TR/xslt (abgerufen am 01.10.2002)*.

[7] *http://www.w3.org/TR/REC-xml (abgerufen am 01.10.2002)*.

[8] *http://www.w3.org/TR1998/REC-xml-19980210 (abgerufen am 01.07.2002)*.

[9] H. KOPKA, P. DALY: *A Guide to LaTeX: Document Preparation for Beginners and Advanced Users*. Addison-Wesley Pub Co., 1999.

[10] *http://java.sun.com/products/servlet/ (abgerufen am 01.10.2002)*.

[11] *http://www.w3.org/DOM/ (abgerufen am 01.10.2002)*.