DOKUMENTATION

des Projektes „Modularisierte Studiengänge“, kurz ModStud

Inhaltsverzeichnis

1 Einführung

Aufgabenstellung des Projektes ModStud war es, auf Grundlage einer Datenbasis von Vorlesungsmodulen im RTF-Format eine Datenbank zu erstellen, die einen Zugriff über verschiedene Schnittstellen erlaubt, worüber die Mitarbeiter und Studenten der Universität Leipzig die Moduldaten nutzen können. Wichtigste Schnittstelle dabei sollte ein Webinterface sein, das den Zugriff über Navigation und leistungsfähige Suchfunktionen gestattet.

Wesentliche Richtlinien bei der Konzeption waren

- Plattformneutralität: die Lösung sollte sowohl unter Unix als auch unter Windows lauffähig sein,
- Verwendung standardisierter Datenformate und Schnittstellen, um die Bindung an ein Softwareprodukt zu minimieren,
- einfache Wartbarkeit und Erweiterbarkeit,
- soweit wie möglich Einsatz von Open-Source und frei verfügbarer Software mit einer großen Entwickler- und Nutzerbasis, die eine fortgesetzte Pflege der Software sicher stellt.

Zur Gewährleistung der Plattformneutralität wurde entschieden, die Lösung in Java zu entwickeln. Mit XML und darauf aufbauenden Standards wie XSLT, XQuery und WSDL wurde der Forderung nach standardisierten Formaten und Schnittstellen entsprochen. Durch Parametrisierung, Modularisierung und Trennung von Logik und Layout ist für eine einfache Wartung und Weiterentwicklung gesorgt. Die aufgrund der Richtlinie ausgewählte Software, die zur Realisierung verwendet wurde, ist in Kapitel 3.1 aufgeführt.

Die Datenbasis wurde nach einer modifizierten DTD eines Partnerprojektes an der Universität Gießen (*mkintern.dtd*) in das XML-Format überführt und in der freien, nativen XML-Datenbank eXist gespeichert. Zum automatisierten Einlesen der Daten wurde ein Kommandozeilen-Tool entwickelt. Auf die Moduldaten kann entweder über ein Webinterface oder über einen Web-Service zurückgegriffen werden. Innerhalb des Webinterfaces ist es möglich, die Module in den Formaten HTML, XML, PDF und PostScript auszugeben oder auch über ein Formular zu editieren.
Diese Funktionen sollen nachfolgend vorgestellt und dokumentiert werden.

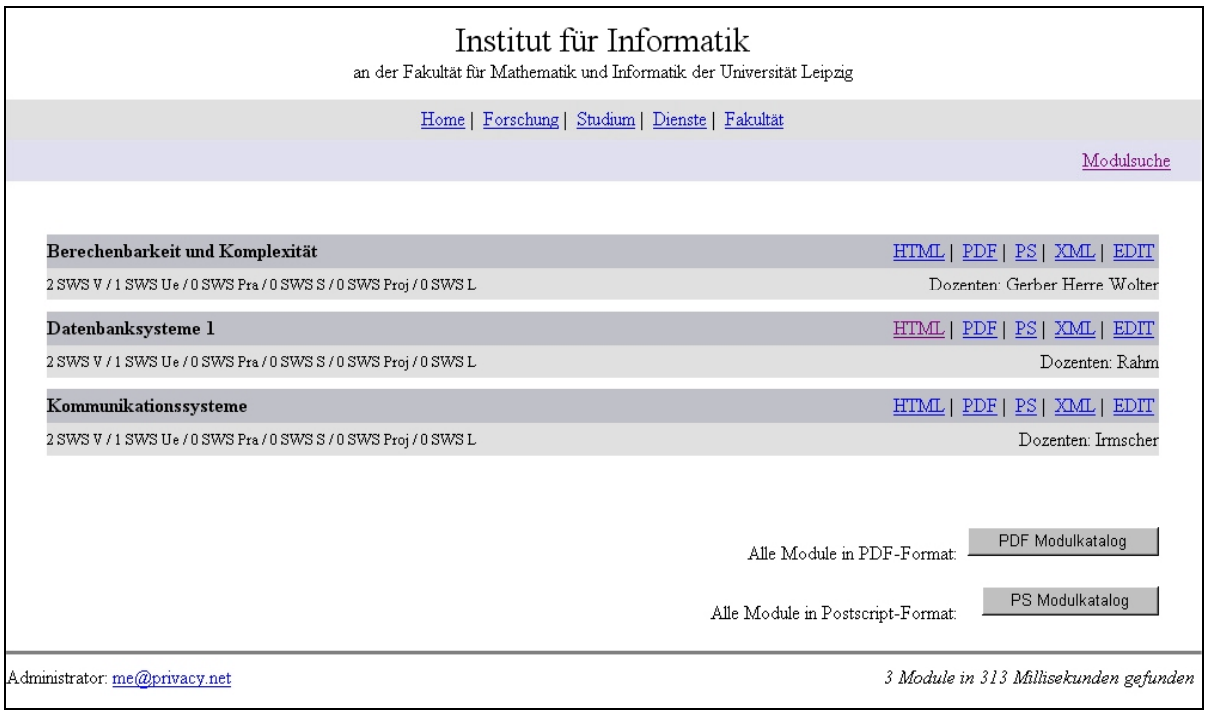
2 Projektfunktionen

2.1 Die Datenbank

Die Datenbank eXist verwaltet die Moduldaten in XML-Form innerhalb von Kollektionen. Kollektionen entsprechen dem Konzept der Verzeichnisse in einem Dateisystem und können bei XML-Anfragen in XPath der Pfadangabe vorangestellt werden. Die Struktur der Kollektionen (*/modstud/Universitätsname/Studiengang/*) ist wie das gesamte Projekt für die Speicherung von mehreren Studiengängen verschiedener Universitäten ausgelegt.

2.2 Das Webinterface

Nach dem Starten des Projektes erscheint bei Aufruf der Seite http://server:port/exist/ die Startseite. Sie bietet standardmäßig eine Übersicht über alle in der Datenbank befindlichen Module (des Studienganges Informatik der Universität Leipzig), geordnet nach Modulnamen.

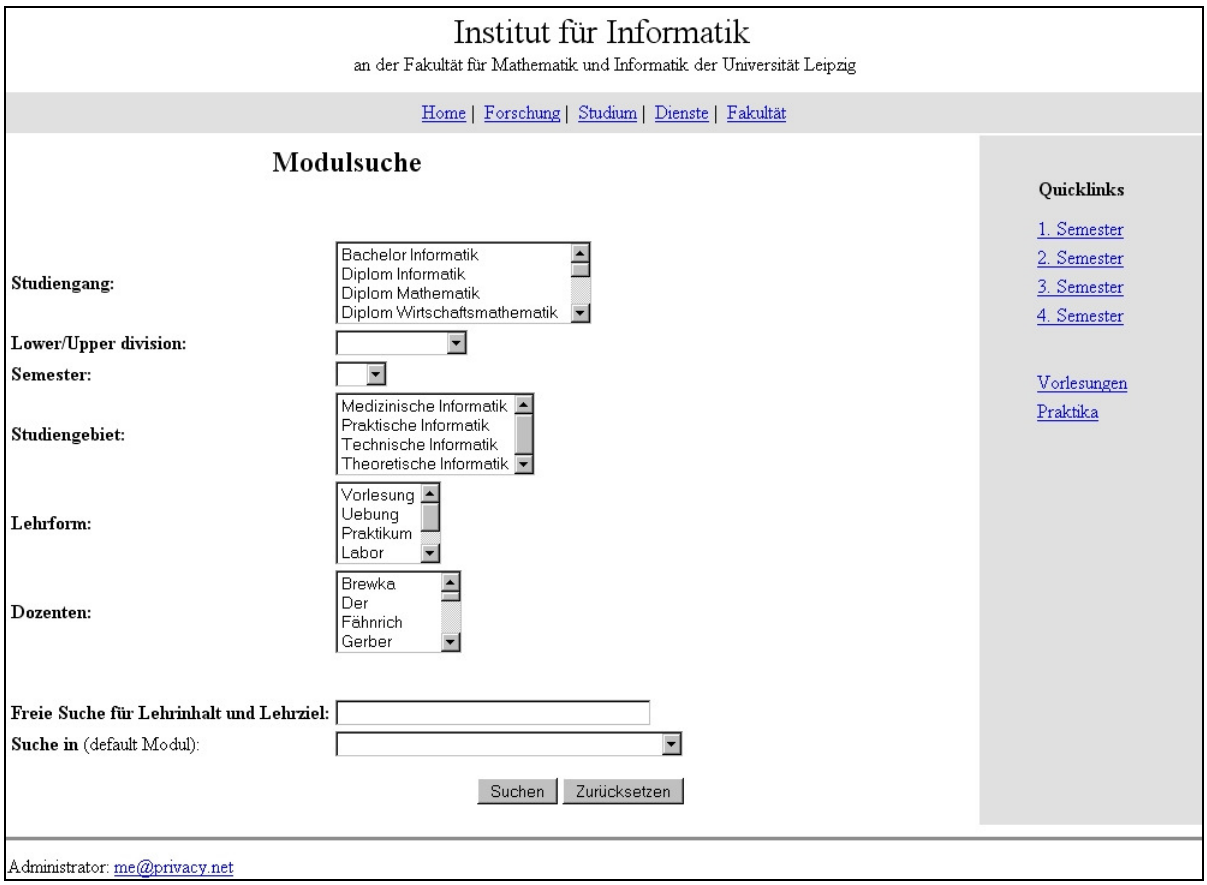


Zu jedem Modul werden der Modulname und in einer weiteren Zeile die Lesenden und der Aufwand pro Lernform in SWS angegeben. Über die Verknüpfungen rechts des Modulnamens kann die Darstellung des jeweiligen Moduls (HTML, PDF oder PS, XML) gewählt werden oder dieses in einem HTML-Formular editiert werden. Alle auf dieser Seite angezeigten Module können auch zusammen in einem Modulkatalog als PDF- oder PS-Dokument gespeichert werden, indem der Benutzer auf die Formularbuttons unten rechts auf der Startseite drückt.

Über den Link „Modulsuche“ oben rechts gelangt der Benutzer auf die Suchseite des Projektes. Dort ist es zum Einen möglich, über die Verknüpfungen rechts (Quicklinks) zu Übersichten über die Module einzelner Semester zu gelangen oder sich nur alle Vorlesungen oder Praktika anzeigen zu lassen.

Die Hauptfunktion dieser Seite ist jedoch die individuelle Suche nach Modulen. Dazu kann man bei den Suchparametern für Studiengang, Division, Semester, Studiengebiet, Lehrform und Dozenten unter allen in der DB vorhandenen Einträgen beliebig viele auswählen, die in dem gesuchten Modul vorkommen sollen.

Über den Parametern Lehrinhalt und Lehrziel ist sogar eine freie Volltextsuche möglich. Die eingegebenen Suchwörtern werden dabei mit dem logischen ODER verknüpft und automatisch durch Sternoperatoren am Beginn und am Ende jedes Wortes erweitert, so dass auch Teilausdrücke berücksichtigt werden. Die Mindestlänge der zu suchenden Wörter beträgt in der Voreinstellung 3 Zeichen.

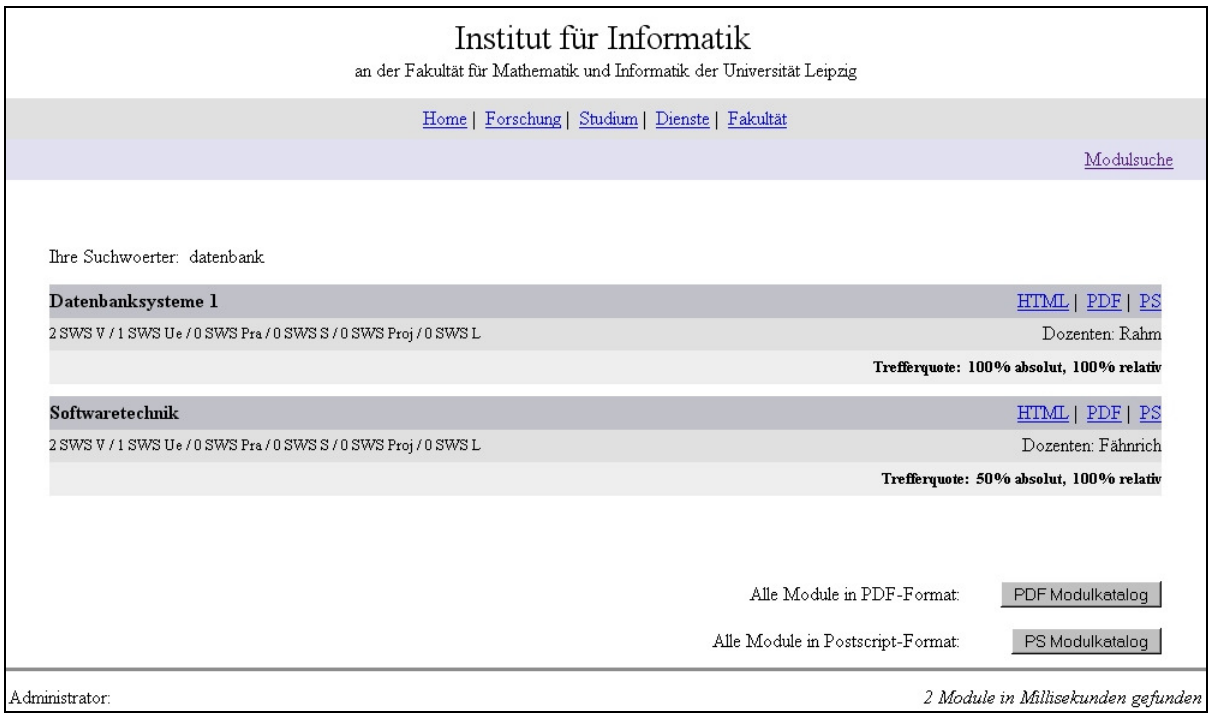


Die Suche liefert nach dem Abschicken des Formulars schließlich eine Ergebnisliste mit allen Modulen, auf die die Suchparameter zutreffen.
Diese wird geordnet nach einer absoluten und einer relativen Trefferquote der Suchbegriffe. Die absolute Trefferquote gibt die Anzahl der absolut gefundenen Suchwörter für ein Modul in Prozent der größten gefundenen Anzahl an, die relative Trefferquote gibt die Anzahl gefundener disjunkter Suchwörter für ein Modul in Prozent an, normiert auf die größte Anzahl gefundener disjunkter Suchwörter innerhalb eines Moduls.

Bei einer Suche nach 2 unterschiedlichen Wörtern erreicht also das Modul eine 100%ige absolute Trefferquote, bei dem die maximale Anzahl an Vorkommen der Suchwörter festgestellt wurde. Die relative Quote kann dagegen nur dann 100% betragen, wenn das Modul beide Suchbegriffe enthält.

Über die Verknüpfungen zu jedem Modul lassen sich die unterschiedlichen Darstellungsoptionen für die Module wählen (siehe Abbildungen am Dokumentende). Die HTML-Darstellung lehnt sich wie die Ausgabe im PDF- oder PS-Format weitgehend an die Vorgabe der RTF-Beschreibungen an.

Zur Weiterverarbeitung der Moduldaten kann sich der Nutzer einerseits das Modul als XML-Datei zurückgeben lassen, um es dann lokal auf dem Client zu bearbeiten, oder gleich das HTML-Editier-Formular benutzen. Dieses bietet für alle änderbaren Parameter der Module (in englisch und deutsch) Felder an, in denen die Änderungen vorgenommen werden können. Listen wie die der Lehrenden, der Literatur oder der Studiengänge können über einen entsprechenden Button erweitert werden. Ist das Modul fertig editiert, werden die Änderungen durch Abschicken des Formulars (Drücken des Xupdate-Buttons) in die Datenbank übertragen.



2.3 Der Webservice

Für die Einbindung der Moduldatenbank in externe Softwareumgebungen steht ein WebService zur Verfügung. Nach einer Authentifizierung am Server können zu einer Modulnummer entweder nur die zugehörigen Creditpoints oder Creditpoints, Niveaustufe und Workload zusammen angefordert werden. Es ist ebenfalls möglich, das gesamte Modul in XML-Form als Antwort zu liefern, um es eventuell an der Senke weiter zu verarbeiten.

2.4 Das Ladeprogramm

Um die Datenbank um neue Module zu erweitern ist es nötig, diese entweder einzeln über den eXist-Client (nächster Abschnitt) einzufügen oder das automatische Ladeprogramm zu nutzen. Diese kann komplette Verzeichnis- und Zugriffsrechtestrukturen von Modulen automatisiert in die Kollektionen der Datenbank übertragen und umgekehrt auch Backups vornehmen bzw. ein Verzeichnis mit der Datenbank abgleichen.

2.5 Der eXist-Client

Die XML-Datenbank eXist bringt bereits einen Client mit, der es ermöglicht, per GUI die DB zu verwalten. Es lassen sich Kollektionen und Dokumente einfügen und entfernen, Rechte setzen und sogar einzelne Dokumente anzeigen und bearbeiten.

3 Technische Dokumentation

3.1 Verwendete Software

Das Paket Modstud besteht ausschließlich aus Komponenten, die in der unabhängigen Programmiersprache Java geschrieben wurden. Dadurch lässt es sich problemlos auf jedes andere System portieren, für das es ein Java Runtime Environment (Version 1.4.x) gibt.

Zentrales Bindeglied ist der kostenlos verfügbare Servlet-Server Jetty, in dessen Containern die restliche Software läuft. Das Paket mit eXist und Cocoon (inklusive FOP als FO-Prozessor) wurde extra für die Erfordernisse des Projektes angepasst und kompiliert.

Während die native XML-Datenbank eXist für die Speicherung und Veränderung der Module zuständig ist, wird Cocoon für die Weiterverarbeitung der Rohdaten verwendet. Dabei wird grundsätzlich über ein XSP- oder ein XQ-Skript eine Anfrage an die Datenbank gestellt, deren Rückgabe über verschiedene Transformationsskripte in der Cocoon-Pipeline schließlich zu einem Ausgabedokument verarbeitet wird.

XSP-Skripte sind dabei XML-Dokumente, in denen Code einer Programmiersprache eingebettet und zur Aufrufzeit ausgeführt werden kann (z.B. Anfragen an die Datenbank und Verarbeitung der Rückgaben). eXist bietet dafür ein sogenanntes Logicsheet an, über das es eine Schnittstelle zur Datenbank über spezielle XML-Tags realisiert. XQ-Dokumente enthalten dagegen nur XQuery-Anweisungen, die direkt der Datenbank übergeben werden. In der Cocoon-Pipeline werden danach die aus den XML-Rückgaben erzeugten SAX-Events durch XSL-Skripte in die Ausgabeformate (HTML, XML, XSL-FO) konvertiert und schließlich zu Ausgabe-Dokumenten serialisiert (z.B. PDF/PS). eXist bietet darüber hinaus weitere Schnittstellen, von denen u.a. der WebService und XML-RPC für das Projekt genutzt wurden und in den jeweiligen Abschnitten erläutert werden.

Damit die Bearbeitung ohne Probleme auf verschiedensprachigen Systemen läuft, wurde die Pipeline komplett auf die XML-Standard-Kodierung UTF-8 umgestellt. Dazu wurden Änderungen an der zentralen Cocoon-Konfigurationsdatei *sitemap.xmap* (`<encoding>UTF-8</encoding>`) und der Serverkonfiguration *web.xml* (`form-encoding` und `container-encoding` auf UTF-8) vorgenommen.
Nachfolgend finden sich die technischen Beschreibungen hinter den einzelen Funktionen, angefangen mit den Modulen, welche die Cocoon-Pipeline verwenden.

3.2 Cocoon-Module

3.2.1 Modulübersicht (modstud_db.xsp, modstud_db.xsl)

Eine Übersichtsdarstellung einer Menge von Modulen erfolgt durch das XSP-Skript *modstud_db.xsp*, welches unter der URL http://server:port/exist/modstud/modstud_db.xsp erreichbar ist.
Dieses stellt in Kurzform den Titel, die SWS der einzelnen Lehreinheiten und die Namen der Dozenten eines Moduls dar. Zu jedem Modul existieren Verknüpfungen zu einer Detaildarstellung in HTML, PDF, Postscript, XML sowie zum Editieren des Moduls (siehe folgende Abschnitte). Die Module sind per Voreinstellung nach Modulnamen sortiert. Ausnahme sind Suchergebnisse mit freier Textsuche, vgl. dazu *Modulsuche*. Weiterhin existieren zwei Verküpfungen, die einen Modulkatalog aller in der momentanen Darstellung angegebenen Module in PDF-Format oder in Postscript-Format generieren.
Die HTML-Darstellung von *modstud_db.xsp* erfolgt im Allgemeinen über das XSL-Skript *modstud_db.xsl*. Bei der Darstellung von Suchergebnissen mit freier Textsuche wird stattdessen das XSL-Skript *modstud_db_freesearch.xsl* benutzt, wobei die Suchergebnisse vorher durch *modstud_sort_searchresults.xsl* sortiert werden.
Für die Auswahl der darzustellenden Module werden die Requestparameter verwendet. Die Bedeutung der einzelnen Parameter ist in *modstud.ModulPropertiesGlobal* festgelegt. Die Parameter werden in *modstud.ModulDBOperationsGlobal.getComplexSearchRequestString* geparst und zu einer XQuery-Anfrage zusammengesetzt. Als legale Zeichen für Parameterwerte werden dabei nur Zahlen und Buchstaben sowie eine in *modstud_db.xsp* angegebene zusätzliche Menge von Sonderzeichen akzeptiert. Für Requestparameterwerte für die freie

Suche in Modulen (vgl. *modstud_search.xsp*) werden XQuery-Funktionen aus der Datei *xqueries_global.xq* verwendet, die beim Initialisieren von *modstud.ModulGlobalProperties* eingelesen wird. Die DB-Anfrage erfolgt über die XML:DB-Schnittstelle von eXist. Für Benutzernamen, Passwort, Kollektion und URI werden die Werte der entsprechenden get-Methoden von *modstud.ModulPropertiesGlobal* verwendet, die Werte können in der globalen Properties-Datei *modstud.properties* festgelegt werden.

Properties von modstud_db.xsp

rooturi	Defaultwert:	xmldb:exist:///db
	get-Methode:	getRootURI_web()
	Beschreibung:	gibt die URI zum DB-Zugriff an
root_collection	Defaultwert:	/db/modstud
	get-Methode:	getRootCollection_web()
	Beschreibung:	Rootpfad für die Kollektionen, oberhalb der nicht auf DB-Daten durch den Client zugegriffen werden kann
default_collection	Defaultwert:	kein Wert
	Get-Methode:	getDefaultCollection_web()
	Beschreibung:	Pfad zur Kollektion, relativ zu *root_collection*
modulkatalog_gueltigkeit	Defaultwert:	w2002
	Get-Methode:	getGueltigkeitKatalog_web()
	Beschreibung:	<gueltig>-Tag aus DTD
user_readonly	Defaultwert:	guest
	Get-Methode:	getUserReadonly_web()
	Beschreibung:	Benutzername für lesenden Zugriff
password_readonly	Defaultwert:	guest
	Get-Methode:	getPasswordReadonly_web()
	Beschreibung:	Passwort für user_readonly
admin_mailaddress	Defaultwert:	kein Wert
	Get-Methode:	getAdminMailAddress()
	Beschreibung:	E_Mail-Adresse des Administrators
Defaultlanguage	Defaultwert:	de
	Get-Methode:	getDefaultLanguage_web()
	Beschreibung:	Default-Sprache für Werteauswahl aus Moduldaten
	Anmerkung:	die Unterstützung vom Sprach-Tag bei Moduldaten ist zur Zeit nicht vollständig implementiert

Weitere Properties für Requestparameter, Datenbanktreiber sind möglich.

3.2.2 Modulsuche (modstud_search.xsp, modstud_search.xsl)

Suchmöglichkeiten in den Modulen existieren über ein XSP-Skript, das über die URL

http://server:port/exist/modstud/modstud_search.xsp erreichbar ist. Die Suchparameter lassen sich in zwei Gruppen unterteilen:

a) Vorgebene Werte: Als Suchparameter mit vorgegebenen Werten existieren zur Zeit Studiengang, Niveaustufe (lower/upper division), Semester, Studiengebiet, Lehrform und Dozenten. Die möglichen Werte ergeben sich dabei entweder aus der DTD (Lehrform) oder aus den Moduldaten. In Studiengang, Studiengebiet, Lehrform und Dozenten ist eine Suche über mehrere Werte möglich.
b) Freie Suche: Eine freie Textsuche ist in Lehrinhalt und Lehrziel eines Moduls und in Lehrinhalt und Lehrziel der von einem Modul vorausgesetzten Module möglich. Die Suchwörtern sind OR-verknüpft und werden durch Sternoperatoren am Beginn und am Ende jedes Wortes erweitert, so dass auch Teilausdrücke berücksichtigt werden. Eine Mindestlänge der Wörter ist festgelegt (in der Voreinstellung 3 Zeichen).

Die HTML-Darstellung von *modstud_search.xsp* erfolgt über das XSL-Skript *modstud_search.xsl*. Die Suchergebnisse werden durch das XSP-Skript *modstud_db.xsp* verarbeitet und dargestellt. Bei einer freien Textsuche wird bei den gefundenen Modulen eine absolute und eine relative Trefferquote der Suchbegriffe angegeben. Die absolute Trefferquote gibt die Anzahl der absolut gefundenen Suchwörter für ein Modul in Prozent an, normiert auf die größte Anzahl absolut gefundener Suchwörter eines Moduls. Die relative Trefferquote gibt die Anzahl gefundener disjunkter Suchwörter für ein Modul in Prozent an, normiert auf die größte Anzahl gefundener disjunkter Suchwörter innerhalb eines Moduls. Die Darstellung in *modstud_db.xsp* ist absteigend sortiert nach der absoluten Trefferquote.
Zugriff und die Festlegung der Werte für Benutzername, Passwort, URI und Kollektion erfolgen wie bei *modstud_db.xsp*. Für die Ermittlung der Werte der einzelnen Suchparameter werden XQuery-Funktionen aus der Datei *xqueries_global.xq* verwendet. Zu den einzelnen Properties siehe *modstud_db.xsp*.

3.2.3 Modulausgabe (Modulausgabe.xsp, Modulausgabe.xsl)

Die Modulausgabe gibt ein Modul aus der Datenbank als XML-Datei zurück. Erwartet werden als URL-Parameter (GET) Werte für das Modul (nr) und das Semester (semester), also z.B. *Modulausgabe.xsp?nr=INF1101&semester=w2002*.
In der XSP-Datei wird über das XMLDB-Logicsheet von eXist eine XPATH-Anfrage nach dem Dokument gestellt, welches dieser Nummer entspricht und das gefragte Semester aufweist. Dieses Dokument wird als XML zurückgegeben und das Stylesheet entfernt nur den obersten document-Tag aus der XSP-Datei.
Damit das auszugebende XML-Dokument auch wieder einen Hinweis auf die verwendete DTD besitzt, muss die Cocoon-Deklaration für den XML-Serializer angepasst werden:

```
<map:serializer mime-type="text/xml" name="modul_xml"
src="org.apache.cocoon.serialization.XMLSerializer">
      <doctype-system>mkintern.dtd</doctype-system>
      <indent>yes</indent>
      <encoding>UTF-8</encoding>
</map:serializer>
```

3.2.4 Modulanzeige (Modulanzeige.xsp, Modulanzeige.xsl)

Die Modulanzeige zeigt ein Modul aus der Datenbank als HTML-Seite an. Erwartet werden als URL-Parameter (GET) Werte für das Modul (nr), das Semester (semester) und die Ausgabesprache (`lang ::= en | de`), also z.B.
Modulanzeige.xsp?lang=en&nr=INF1101&semester=w2002.
In der XSP-Datei wird über das XMLDB-Logicsheet von eXist eine XPATH-Anfrage nach

dem Dokument gestellt, welches dieser Nummer entspricht und das gefragte Semester aufweist. Außerdem werden bei mehrsprachigen Angaben im Moduldokument die der übergebenen Sprache entsprechenden Einträge ausgewählt. Ist eine Angabe nur in einer Sprache verfügbar, so wird diese auf jeden Fall angezeigt. Die so gewonnenen SAX-Events verarbeitet das Stylesheet zu einer HTML-Repräsentation des Modules.

3.2.5 Moduledit (Moduledit.xsp, Moduledit.xsl)

Dieses Modul erzeugt ein HTML-Formular mit den Daten aus dem übergebenen Modul, so dass dieses editiert werden kann. Nach dem Abschicken des Formulars durch Klicken des XUpdate-Buttons ganz unten auf der Seite wird das Modul auf dem Server mit XUpdate aktualisiert. Erwartet werden als URL-Parameter (GET) Wert für das Modul (nr) und das Semester (semester), also z.B. *Moduledit.xsp?nr=INF1101&semester=w2000*. Die mehrsprachigen Einträge des Moduls können dabei separat voneinander eingegeben werden.
Um ein neues Modul aus einem alten zu erschaffen, muss ein weiterer Parameter übergeben werden, nämlich "neu=true". Dann kann man einen neuen Namen für das Modul eingeben, unter dem es dann in der Datenbank gespeichert wird.
Die Felder für die Angaben von Studiengängen, Literaturangaben, Lesenden und Vorkenntnissen können durch Klicken auf den darunter liegenden Button erweitert werden, so dass weitere Angaben möglich sind.
Die XSP-Datei erfüllt 3 verschiedene Aufgaben:

a) beim ersten Aufruf der Seite wird über das XMLDB-Logicsheet von eXist eine XPATH-Anfrage nach dem Dokument gestellt, welches dieser Nummer entspricht und das gefragte Semester aufweist. Dieses wird als XML an das Stylesheet weitergereicht.
b) wurde die Seite bereits aufgerufen und ein Erweitern-Button angeklickt, so wird eine XML-Repräsentation des Moduls angelegt, in dem die bereits editierten Angaben übernommen werden. Zusätzich werden neue Felder dort angelegt, wo sie durch den Erweitern-Button angefordert werden. Danach wird dieses neue, temporäre Dokument an das Stylesheet übergeben.
c) wurde schließlich der XUpdate-Button angeklickt, so wird aus den Formulardaten ein XUpdate-Dokument erstellt. In dem Fall, dass ein neues Dokument erstellt werden soll, wird die Eingabe für den neuen Namen geprüft und dann vor dem Updaten dieses als Kopie des alten Dokumentes angelegt (per XML-RPC). Das Updaten erfolgt in diesem Fall auf dem neuen Dokument. Am Ende wird dann schließlich das XUpdate-Dokument an eine XML-RPC-Methode übergeben.

In den ersten beiden Fällen erzeugt das Stylesheet aus dem übergebenen XML-Dokument das Formular; im zweiten Fall nur die Meldung ob das Updaten erfolgreich war oder nicht.

3.3 PDF/Postscript-Darstellung

Für die Erzeugung von Modulbeschreibungen im PDF- oder Postscript-Format existiert ein Skript *modstud_fo_pdf.xsl* im xsl:fo-Format, welches eine oder mehrere in XML vorliegende Modulbeschreibung(en) transformiert. Verküpfungen zu PDF/PS-Darstellungen von www-Seiten aus existieren zur Zeit nur von *modstud_db.xsp*, das Skript kann aber auch unabhängig von der Cocoon/eXist-Umgebung mit einem FO-Prozessor benutzt werden. Als Renderer wird FOP vom Apache-Projekt verwendet. Zur Verwendung innerhalb der eXist/Cocoon-Umgebung ist eine neu kompilierte Cocoon-Version mit den entsprechenden FO-Schnittstellen notwendig.

3.4 SOAP/WSDL-Schnittstelle („Webservice“)

3.4.1 Beschreibung

Für den lesenden Zugriff auf die Moduldaten steht eine SOAP/WSDL-Schnittstelle zur Verfügung. In der Defaultkonfiguration ist die Schnittstelle unter der URL http://server:port/exist/services/ModulQuery erreichbar, die WSDL-Beschreibung kann unter der URL http://server:port/exist/services/ModulQuery?WSDL erreicht werden. Es sind drei Methoden für die Abfrage der Moduldaten implementiert sowie eine zusätzliche Methode für die E-Mail-Adresse des Serveradministrators. Die Methoden für die Moduldaten liefern:

a) Creditpoints für ein Modul
b) Creditpoints, Niveaustufe, Workload für ein Modul
c) XML-Darstellung eines Moduls

Die prinzipielle Architektur für den Zugriff basiert auf einem einfachen Session-Konzept. Dafür existiert eine Methode zur Authentifizierung und Etablierung einer Session und eine Methode zur Beendigung einer Session. Der Ablauf eines Zugriffs hat folgende Form:

1.*connect (user, password)*	Authentifizierung und Etablierung der Session, Rückgabe einer Session-ID an Client
2.*Zugriff auf Moduldaten*	Authentifizierung durch Session-ID
3.*disconnect (sessionid)*	Beendigung der Session

3.4.2 Methoden

3.4.2.1 connect

Die Methode authentifiziert den Client gegenüber eXist und etabliert eine Session.

Spezifikation:

Parameter 1:	*String username*	Benutzername	
Parameter 2:	*String password*	Passwort des Benutzers	
Rückgabe:	*ModulConnect*		
Rückgabeparameter:	*int errorvalue*	0	erfolgreiche Authentifizierung, Session etabliert
		1	kein gültiger Benutzername
		2	kein gültiges Passwort
		3	nicht spezifizierter Fehler
		4	maximale Anzahl Sessions etabliert
	String errorvalue	enthält bei Fehler Fehlermeldung	
	String sessionid	enthält bei Erfolg Session-ID	

Der Client identifiziert sich durch Benutzernamen und Passwort. Die Daten werden durch die eXist-DB überprüft und bei Gültigkeit eine Session etabliert. Jede Session hat eine durch den Administrator festgelegte Gültigkeitsdauer, die bei jedem späteren Zugriff erneuert wird.

Eine Session wird also erst dann ungültig, wenn eine vorgegebene Zeitspanne lang nicht auf den Server zugegriffen wurde. Die Anzahl der gleichzeitig möglichen Sessions ist durch den Administrator begrenzt. Die Session-ID ergibt sich aus dem Hashwert des Sessionobjekts (*Object.hashCode()*).

3.4.2.2 disconnect

Die Methode beendet eine Session.

Spezifikation:

Parameter 1:	*String sessionid*	Session-ID von zu beendender Session
Rückgabe:	*void*	

3.4.2.3 getCreditPoints

Die Methode liefert die Leistungspunkte (ECTS-Creditpoints) für ein Modul.

Spezifikation:

Parameter 1:	*String modulid*	Modul-ID des gesuchten Moduls
Parameter 2:	*String collection*	Relativer Pfad zur Kollektion, innerhalb der gesucht werden soll; optionaler Parameter
Parameter 3:	*String sessionid*	Session-ID
Parameter 4:	*String gueltigkeit*	Gültigkeit des Modulkataloges (vgl. <gueltig>-Tag in DTD); optionaler Parameter
Rückgabe:	*ModulCreditPoints*	
Rückgabeparameter:	*int errorvalue*	0 Anfrage erfolgreich 1 *modulid*enthält illegale Zeichen oder *gueltigkeit* ist nicht valide 2 nicht näher spezifizierter Fehler 3 inkonsistente Daten (Creditpoints kein int-Wert) 4 *modulid* nicht eindeutig 5 kein Modul mit *modulid* gefunden 6 Fehler bei DB-Anfrage 7 keine/ungültige *sessionid*
	String errorvalue	enthält bei Fehler Fehlermeldung
	int value	bei Erfolg Creditpoints

Anmerkung zum *collection*-Parameter: Der Gesamtpfad zur Kollektion setzt sich aus einem durch den Administrator vorgebenen Rootpfad und dem (optionalen) Parameter *collection* zusammen. Dadurch kann der Administrator Kollektionenbereiche in der DB vor dem Zugriff von außen schützen. Falls der Parameter *collection* nicht angegeben wird, setzt sich der Gesamtpfad aus dem Rootpfad und einem Defaultpfad zusammen.

3.4.2.4 getModulValues

Die Methode liefert folgende Werte für ein Modul: Creditpoints, Leistungsnachweis, Niveaustufe, Semester, Workload (Lehre, Eigenarbeit, Gesamt).

Spezifikation:

Parameter 1:	*String modulid*	Modul-ID des gesuchten Moduls
Parameter 2:	*String collection*	Relativer Pfad zur Kollektion, innerhalb der gesucht werden soll; optionaler Parameter

Parameter 3:	*String sessionid*	Session-ID	
Parameter 4:	*String gueltigkeit*	Gültigkeit des Modulkataloges (vgl. <gueltig>-Tag in DTD); optionaler Parameter	
Rückgabe:	*ModulValues*		
Rückgabeparameter:	i*nt errorvalue*	0	Anfrage erfolgreich
			modulid enthält illegale Zeichen oder *gueltigkeit* ist nicht valide
		2	nicht näher spezifizierter Fehler
		3	inkonsistente Daten
		4	*modulid* nicht eindeutig
		5	kein Modul mit *modulid* gefunden
		6	Fehler bei DB-Anfrage
		7	keine/ungültige *sessionid*
	String errorvalue	enthält bei Fehler Fehlermeldung	
	int creditpoints	Creditpoints	
	String leistungsnachweis	Leistungsnachweis	
	float workloadGesamt	Workload Gesamt	
	float workloadEigenarbeit	Workload Eigenarbeit	
	float workloadLehre	Workload Lehre	
	String niveaustufe	Niveaustufe	
	int niveaustufeSemester	Niveaustufe Semester	

3.4.2.5 getModulAsXMLString

Die Methode liefert ein Modul in XML-Format.

Spezifikation:

Parameter 1:	*String modulid*	Modul-ID des gesuchten Moduls	
Parameter 2:	*String collection*	Relativer Pfad zur Kollektion, innerhalb der gesucht werden soll; optionaler Parameter	
Parameter 3:	*String sessionid*	Session-ID	
Parameter 4:	*String gueltigkeit*	Gültigkeit des Modulkataloges (vgl. <gueltig>-Tag in DTD); optionaler Parameter	
Rückgabe:	*ModulAsXMLString*		
Rückgabeparameter:	i*nt errorvalue*	0	Anfrage erfolgreich
		1	*modulid*enthält illegale Zeichen oder *gueltigkeit* ist nicht valide
		2	nicht näher spezifizierter Fehler
		4	*modulid* nicht eindeutig
		5	kein Modul mit *modulid* gefunden
		6	Fehler bei DB-Anfrage
		7	keine/ungültige *sessionid*
	String errorvalue	enthält bei Fehler Fehlermeldung	
	String value	enthält Modul in XML-Format	

3.4.2.6 getDBAdminMailAddress

Die Methode liefert die E-Mail-Adresse des Administrators. Sie kann ohne Session-ID benutzt werden.

Spezifikation:

Parameter:	*void*	
Rückgabe:	*String*	bei Erfolg Email-Adresse, sonst leeren String

3.4.3 Clients

Zum einfachen Testen der SOAP-Schnittstelle sind zwei Clients implementiert, die untermodstud/ws/clients verfügbar sind.

- Java-Client: Der Client wurde mit der Apache Axis-Bibliothek realisiert. Parameter wie Username, Passwort, Modulid werden aus einer Properties-Datei *webserviceclient.properties* gelesen.
- Python-Client: Dieser Client wurde mit Hilfe der SOAPpy-Bibliothek realisiert. Parameter sind im Quelltext zu ändern. Für die Ausführung ist ein Python-Interpreter und die SOAPpy-Bibliothek notwendig.

3.4.4 Administration

Die Klassendateien der SOAP-Schnittstelle werden zusammen mit den Klassendateien für die allgemeine Konfiguration (modstud.*) in einem JAR-Archiv unter der Verzeichnisstruktur *webapp/WEB-INF/lib/modstud.jar*in die Laufzeitumgebung des Servers eingebunden. Der Webservice ist in der Axis-Konfigurationsdatei *webapp/WEB-INF/server-config.wsdd* eingetragen. Für das Logging von Zugriffen auf die SOAP-Schnittstelle wird die Logging-Datei von eXist *log4j.xml* verwendet.

3.4.5 Properties

Variable Parameter der SOAP-Schnittstelle werden in der globalen Modstud-Propertiesdatei *modstud.properties*angegeben und beim Starten des Servers eingelesen. Defaultwerte sind in der Klasse *modstud.ModstudPropertiesGlobal* festgelegt. Der Zugriff auf die Properties von der SOAP-Schnittstelle aus erfolgt über get-Methoden in *modstud.ModstudPropertiesGlobal*, benutzte Properties von der SOAP-Schnittstelle sind also nicht vom Propertienamen abhängig, sondern vom zurückgelieferten Wert der entsprechenden get-Methode.

Die Properties im einzelnen:

maximum_sessions	Defaultwert:	600
	get-Methode:	getMaxSessions_ws()
	Beschreibung:	gibt die maximale Anzahl gleichzeitig möglicher Sessions an
rooturi_ws	Defaultwert:	xmldb:exist://localhost:8080/exist/xmlrpc
	get-Methode:	getRootURI_ws()
	Beschreibung:	gibt die URI zum DB-Zugriff an
root_collection	Defaultwert:	/db/modstud
	get-Methode:	getRootCollection_ws()
	Beschreibung:	Rootpfad für die Kollektion, unterhalb der nicht auf DB-Daten durch den Client zugegriffen werden kann

	Anmerkung:	zur Zeit die gleiche Propertie wie beim http-Zugriff
default_collection	Defaultwert:	kein Wert
	Get-Methode:	getDefaultCollection_ws()
	Beschreibung:	Pfad zur Kollektion, relativ zu *root_collection*
modulkatalog_gueltigkeit	Defaultwert:	w2002
	Get-Methode:	getGueltigkeitKatalog_ws()
	Beschreibung:	<gueltig>-Tag aus DTD
admin_mailaddress	Defaultwert:	kein Wert
	Get-Methode:	getAdminMailAddress()
	Beschreibung:	E-Mail-Adresse des Administrators

3.5 Das Ladeprogramm

3.5.1 Beschreibung

Zum Austausch von Moduldateien zwischen der Datenbank und dem Dateisystem steht ein Ladeprogramm zur Verfügung. Das Programm verfügt über drei grundsätzliche Funktionen:

1. *Upload* von Dateien aus dem Dateisystem in die Datenbank
2. *Backup* von Modulen aus der Datenbank in das Dateisystem
3. *Synchronisation* zwischen Datenbank und Dateisystem

Neben dem Modulaustausch können auch die Zugriffsberechtigungen (Besitzer, Gruppe, Berechtigungen) auf die Module in der Datenbank geladen bzw. gespeichert werden.

3.5.2 Funktionen

3.5.2.1 Upload

Beim Laden von Modulen werden eine Verzeichnisstruktur und die darin enthaltenen Moduldateien aus dem Dateisystem auf eine Kollektionenstruktur in der DB abgebildet. Dateien aus dieser Verzeichnisstruktur werden zuerst nach Dateiendung ausgewählt (im Normalfall „.xml“) und dann mit der DTD der Moduldateien auf Gültigkeit überprüft. Die Zugriffsberechtigungen auf das Modul in der DB werden, falls vorhanden, aus einer Datei zur Verwaltung dieser Berechtigungen gelesen und in der DB gesetzt. Alternativ werden Defaultwerte für den Zugriff gesetzt.

3.5.2.2 Backup

Beim Backup wird eine Kollektionenstruktur aus der DB auf eine Verzeichnisstruktur im Dateisystem abgebildet. Die Zugriffsberechtigungen der Module und Kollektionen werden ebenfalls in einer Datei gespeichert, um sie bei einem eventuellen späteren Laden wieder in die DB übertragen zu können.

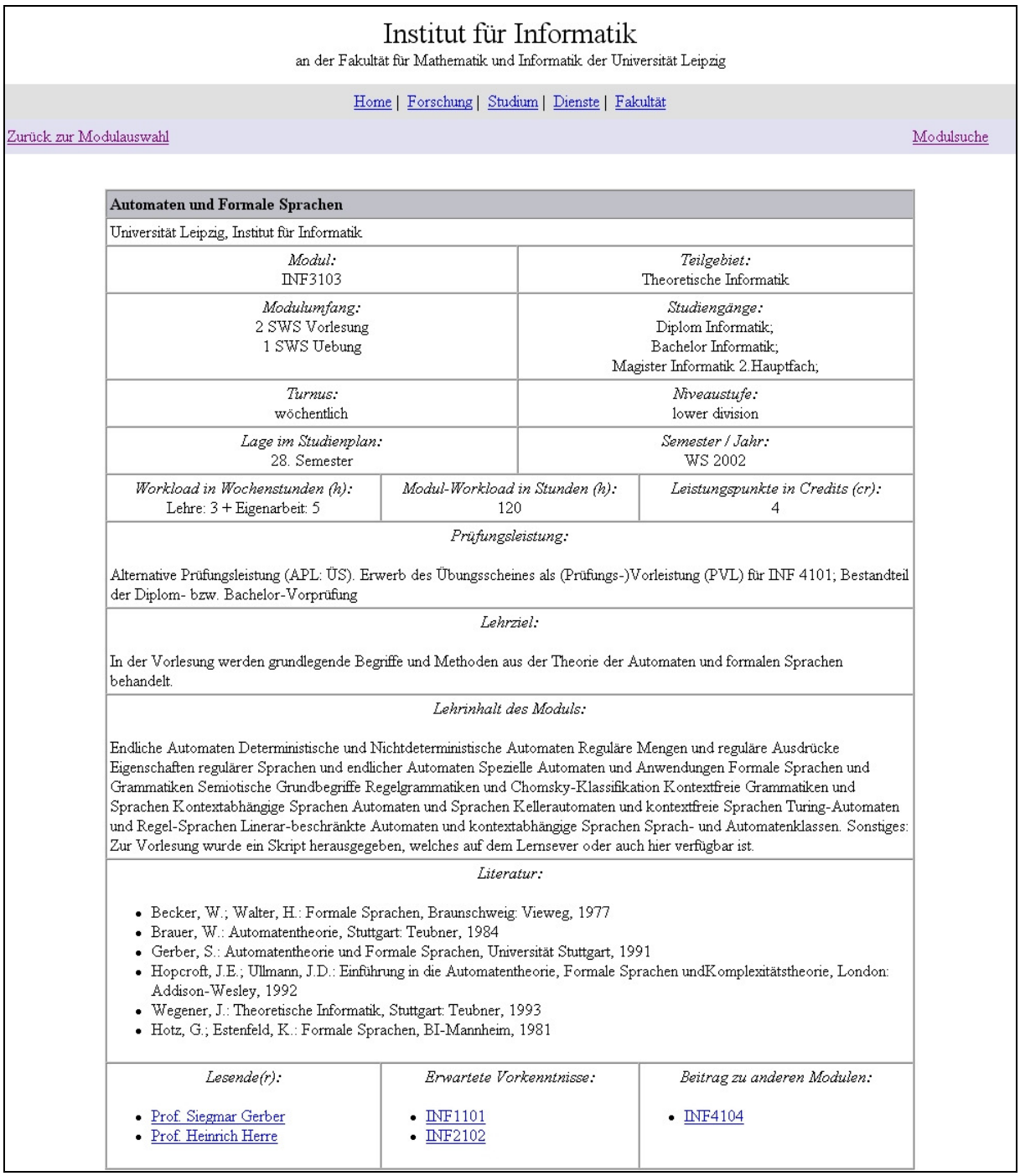
3.5.2.3 Synchronisation

Diese Funktion vereint die Funktionen *Backup* und *Upload* und bietet zusätzlich eine Überprüfung von Änderungen an Modulen, d.h. es wird eine Verzeichnisstruktur im Dateisystem und eine Kollektionenstruktur in der DB in einem Zeitintervall synchronisiert. Neue Module in der DB oder im Dateisystem werden ausgetauscht. Weiterhin werden die

letzten Änderungszeitpunkte der Module sowohl im Dateisystem als auch in der DB überprüft. Falls ein Modul seit dem letzten Überprüfungszeitpunkt geändert wurde, wird es ebenfalls ausgetauscht. Wie beim Backup bzw. Upload werden die Zugriffsberechtigungen für die Module in der DB gesetzt bzw. in eine Datei gespeichert.

3.5.2.4 Administration

Die Administration erfolgt in erster Linie über eine Propertiesdatei *modstudloader.properties*, in der die Parameter für DB-URI, Zugriffsberechtigungen, Verzeichnis im Dateisystem, Zeitintervall für Synchronisation etc. angegeben werden können. Die Parameter Verzeichnis, Zugriffsberechtigungen und Zeitintervall können auch als Programmargumente übergeben werden. Das Logging des Programms erfolgt über LOG4J.



Institut für Informatik

an der Fakultät für Mathematik und Informatik der Universität Leipzig

Home | Forschung | Studium | Dienste | Fakultät

Zurück zur Modulauswahl Modulsuche

Automaten und Formale Sprachen

Universität Leipzig, Institut für Informatik

Modul: INF3103	*Teilgebiet:* Theoretische Informatik
Modulumfang: 2 SWS Vorlesung 1 SWS Uebung	*Studiengänge:* Diplom Informatik; Bachelor Informatik; Magister Informatik 2.Hauptfach;
Turnus: wöchentlich	*Niveaustufe:* lower division
Lage im Studienplan: 28. Semester	*Semester / Jahr:* WS 2002

Workload in Wochenstunden (h):	*Modul-Workload in Stunden (h):*	*Leistungspunkte in Credits (cr):*
Lehre: 3 + Eigenarbeit: 5	120	4

Prüfungsleistung:

Alternative Prüfungsleistung (APL: ÜS). Erwerb des Übungsscheines als (Prüfungs-)Vorleistung (PVL) für INF 4101; Bestandteil der Diplom- bzw. Bachelor-Vorprüfung

Lehrziel:

In der Vorlesung werden grundlegende Begriffe und Methoden aus der Theorie der Automaten und formalen Sprachen behandelt.

Lehrinhalt des Moduls:

Endliche Automaten Deterministische und Nichtdeterministische Automaten Reguläre Mengen und reguläre Ausdrücke Eigenschaften regulärer Sprachen und endlicher Automaten Spezielle Automaten und Anwendungen Formale Sprachen und Grammatiken Semiotische Grundbegriffe Regelgrammatiken und Chomsky-Klassifikation Kontextfreie Grammatiken und Sprachen Kontextabhängige Sprachen Automaten und Sprachen Kellerautomaten und kontextfreie Sprachen Turing-Automaten und Regel-Sprachen Linerar-beschränkte Automaten und kontextabhängige Sprachen Sprach- und Automatenklassen. Sonstiges: Zur Vorlesung wurde ein Skript herausgegeben, welches auf dem Lernsever oder auch hier verfügbar ist.

Literatur:

- Becker, W.; Walter, H.: Formale Sprachen, Braunschweig: Vieweg, 1977
- Brauer, W.: Automatentheorie, Stuttgart: Teubner, 1984
- Gerber, S.: Automatentheorie und Formale Sprachen, Universität Stuttgart, 1991
- Hopcroft, J.E.; Ullmann, J.D.: Einführung in die Automatentheorie, Formale Sprachen undKomplexitätstheorie, London: Addison-Wesley, 1992
- Wegener, J.: Theoretische Informatik, Stuttgart: Teubner, 1993
- Hotz, G.; Estenfeld, K.: Formale Sprachen, BI-Mannheim, 1981

Lesende(r):	*Erwartete Vorkenntnisse:*	*Beitrag zu anderen Modulen:*
• Prof. Siegmar Gerber • Prof. Heinrich Herre	• INF1101 • INF2102	• INF4104

Abbildung 1 - Darstellung eines Moduls als HTML-Dokument

Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik
http://www.informatik.uni-leipzig.de

Modulbeschreibungen für Lehrveranstaltungen

Automaten und Formale Sprachen

Niveaustufe:	lower division	**Modulnummer:**	INF3103
Semester:	28	**Creditpoints (ECTS):**	4
Teilgebiet:	Theoretische Informatik	**Semesterturnus:**	Wintersemester
Turnus:	wöchentlich	**Dauer:**	1 Semester
Lehrumfang:	2 SWS Vorlesung + 1 SWS Übung		

Lernziel

In der Vorlesung werden grundlegende Begriffe und Methoden aus der Theorie der Automaten und formalen Sprachen behandelt.

Lerninhalt

Endliche Automaten Deterministische und Nichtdeterministische Automaten Reguläre Mengen und reguläre Ausdrücke Eigenschaften regulärer Sprachen und endlicher Automaten Spezielle Automaten und Anwendungen Formale Sprachen und Grammatiken Semiotische Grundbegriffe Regelgrammatiken und Chomsky-Klassifikation Kontextfreie Grammatiken und Sprachen Kontextabhängige Sprachen Automaten und Sprachen Kellerautomaten und kontextfreie Sprachen Turing-Automaten und Regel-Sprachen Linerar-beschränkte Automaten und kontextabhängige Sprachen Sprach- und Automatenklassen. Sonstiges: Zur Vorlesung wurde ein Skript herausgegeben, welches auf dem Lernsever oder auch hier verfügbar ist.

Literatur

Becker, W.; Walter, H.: Formale Sprachen. Braunschweig: Vieweg, 1977
Brauer, W.: Automatentheorie. Stuttgart: Teubner, 1984
Gerber, S.: Automatentheorie und Formale Sprachen. Universität Stuttgart, 1991
Hopcroft, J.E.; Ullmann, J.D.: Einführung in die Automatentheorie, Formale Sprachen undKomplexitätstheorie. London: Addison-Wesley, 1992
Wegener, J.: Theoretische Informatik. Stuttgart: Teubner, 1993
Hotz, G.; Estenfeld, K.: Formale Sprachen. BI-Mannheim, 1981

Leistungsnachweis

Alternative Prüfungsleistung (APL: ÜS). Erwerb des Übungsscheines als (Prüfungs-)Vorleistung (PVL) für INF 4101; Bestandteil der Diplom- bzw. Bachelor-Vorprüfung

Erwartete Vorkenntnisse

INF1101
INF2102

Workload

pro Woche (h):	3 h Lehre + 5 h Eigenarbeit	Gesamt (h):	120 h

Dozenten

Prof. Siegmar Gerber, email: gerber@informatik.uni-leipzig.de

Abbildung 2 - Modul als PDF-Dokument

```
- <module>
  - <mk>
    - <docinfo>
        <gueltig>w2002</gueltig>
        <institution xml:lang="de">Universität Leipzig, Institut für Informatik</institution>
      </docinfo>
      <vorwort xml:lang="de"/>
      <beschreibung xml:lang="de"/>
    - <modulliste>
      - <modulgruppe>
          <gruppentitel xml:lang="de">Theoretische Informatik</gruppentitel>
        - <modul nr="INF3103" lp="4">
          - <modulinfo>
            - <dozent>
                <titel>Prof.</titel>
                <vorname>Siegmar</vorname>
                <name>Gerber</name>
                <email>gerber@informatik.uni-leipzig.de</email>
              </dozent>
            - <dozent>
                <titel>Prof.</titel>
                <vorname>Heinrich</vorname>
                <name>Herre</name>
                <email>herre@informatik.uni-leipzig.de</email>
              </dozent>
            </modulinfo>
          - <modulinhalt>
              <modultitel xml:lang="de">Automaten und Formale Sprachen</modultitel>
              <sprache>de</sprache>
              <studiengang>Diplom Informatik</studiengang>
              <studiengang>Bachelor Informatik</studiengang>
              <studiengang>Magister Informatik 2.Hauptfach</studiengang>
              <angebot turnus="w" dauer="1"/>
            - <lernform turnus="wöchentlich">
                <eigenarbeit-w>5 SWS</eigenarbeit-w>
                <workload lehre="3" eigenarbeit="5" gesamt="120"/>
                <vorlesung>2 SWS</vorlesung>
                <uebung>1 SWS</uebung>
              </lernform>
              <niveaustufe semester="28">lower division</niveaustufe>
            - <leistungsnachweis xml:lang="de">
                Alternative Prüfungsleistung (APL: ÜS). Erwerb des Übungsscheines als (Prüfungs-)Vorleistung (PVL) für INF 4101; Bestandteil der Diplom- bzw. Bachelor-Vorprüfung
              </leistungsnachweis>
            - <lernziel xml:lang="de">
                In der Vorlesung werden grundlegende Begriffe und Methoden aus der Theorie der Automaten und formalen Sprachen behandelt.
              </lernziel>
            - <lerninhalt xml:lang="de">
                Endliche Automaten Deterministische und Nichtdeterministische Automaten Reguläre Mengen und reguläre Ausdrücke Eigenschaften regulärer Sprachen und endlicher Automaten Spezielle Automaten und Anwendungen Formale Sprachen und Grammatiken Semiotische Grundbegriffe Regelgrammatiken und Chomsky-Klassifikation Kontextfreie Grammatiken und Sprachen Kontextabhängige Sprachen Automaten und Sprachen Kellerautomaten und kontextfreie Sprachen Turing-Automaten und Regel-Sprachen Linerar-beschränkte Automaten und kontextabhängige Sprachen Sprach- und Automatenklassen. Sonstiges: Zur Vorlesung wurde ein Skript herausgegeben, welches auf dem Lernsever oder auch hier verfügbar ist.
              </lerninhalt>
            - <litverz>
              - <lit>
                  <litautor>Becker, W.; Walter, H.</litautor>
                  <littitel>Formale Sprachen</littitel>
                  <verlag>Braunschweig: Vieweg, 1977</verlag>
                </lit>
```

Abbildung 3 - Modul im XML-Format (Auszug)

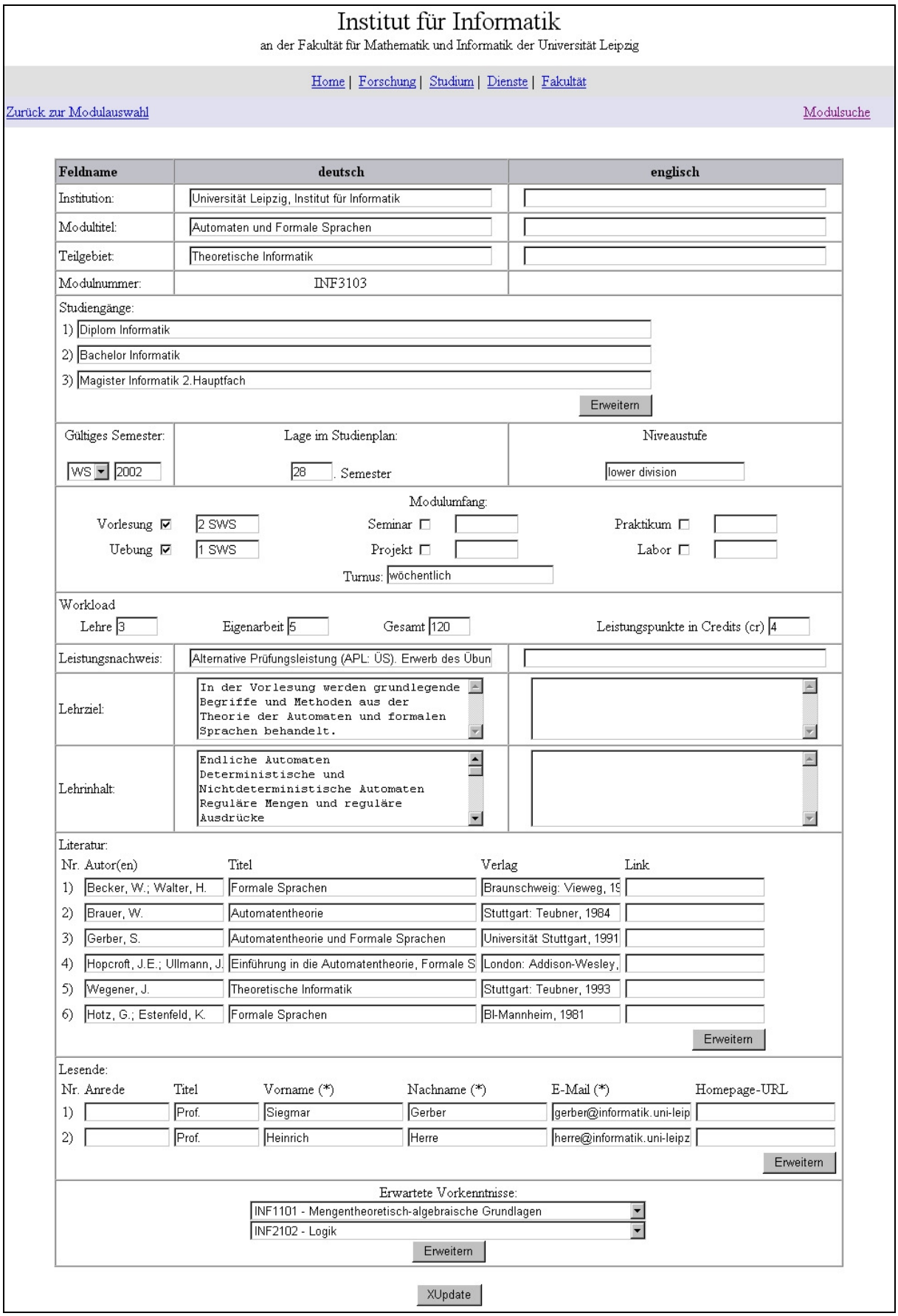


Abbildung 4 - HTML-Formular für das Editieren eines Moduls