OPUS+ Anwender- und Referenzhandbuch

Alexander Wagner

Revision: 1.12 - Date: 2006/09/06 22:05:04

Inhaltsverzeichnis

Teil I

Handbuch

Kapitel 1

Einführung

OPUS ist ein von der Universität Stuttgart entwickelts System zur elektronischen Erfassung von Publikation und Prüfnungsschriften an Universitätsbibliotheken. Die primäre Anwendung von OPUS ist hierbei die Erfassung von Prüfungsschriften wie Dissertationen, Habilitationen und ähnlichem.

An der Universitätsbibliothek Würzburg bestand nun der Wunsch, auch für andere wissenschaftliche Publikationen (Zeitschriftenaufsätze u. ä.) ein elektronisches Publikationssystem zu haben um dieses als *Institutional Repository* einsetzen zu können, wozu in OPUS 3.0 noch einige Erweiterungen nötig sind:

- Dokumentzuordnung: Hier bietet OPUS lediglich Fakultäten und Institute an. Es ist aber notwendig diese Zugehörigkeit bis zur Arbietsgruppenebene darstellen zu können. Weiterhin fehlt in OPUS die Möglichkeit sonstige Einrichtungen wie z. B. einen Sonderforschungsbereich abzubilden.

- Personalisierte Funktionen: Da OPUS keine Anmeldung erfordert ist das erzeugen von Listen auf Autorenbasis zunächst nicht eindeutig möglich. Für die neue Anwendung war dies aber gewünscht wobei allerdings eine Anmeldung des Benutzers am System weiterhin vermieden werden sollte, da derzeit keine zentralen Authentifizierungssysteme zur Verfügung stehen. Neben der Erstellung autorenbezogener Listen sollte es auch möglich sein, diese auf jeder Organisationsebene zu erzeugen, also z. B. die Liste aller Publikationen eines Lehrstuhls oder einer Arbeitsgruppe.

- Da weiterhin das neue System als Archiv dient, dürfen Benutzer zwar Datensätze anlegen, aber nicht löschen oder bearbeiten können. So soll es z. B. möglich sein, eine Arbeitsgruppe zu definieren, für die konsi-

stente Archivierung soll diese aber nicht mehr vom Benutzer geändert werden können.

Lediglich die direkten Benutzerdaten (Name etc.) kann der Benutzer (bis auf seine ID) ändern. Da diese Daten beim Speichern eines Dokuments auf die Datenbank als Dokumentmerkmal übertragen werden ist hier auch bei nachträglichem Ändern der Benutzerdaten eine konsistente Archivierung gegeben.

Beispiel: Herr Mayer ändert (z. B. durch Heirat) seinen Namen in Müller. Alle Papiere die er als Mayer eingestellt hat werden auch weiterhin unter diesem Namen erscheinen. Diejenigen, die er zukünftig einstellt werden jedoch mit dem Namen Müller versehen. Unabhängig davon kann aber Herr Müller eine Liste aller seiner Veröffentlichungen (unabhängig von seinem Namen) erhalten, da diese Zuordnung über die Autorenkennung erfolgt, welche beim Einstellen des Dokuments mit diesem abgelegt wird. [1]

- Bibliographische Daten der Publikation: Da sich OPUS auf Prüfungsarbeiten konzentriert werden keine Zeitschriftenreferenzen erfaßt. Für ein Institutional Repository sind diese aber notwendig. Weiterhin soll über die DOI eine Verknüpfung zum publizierten Volltext möglich sein, insbesondere da einige Verlage eine solche Verknüpfung über ihre Lizenzbestimmungen verlangen.

- Importschnittstellen: Da es für die Benutzer möglichst einfach sein soll, die Daten zu erfassen ist eine Anbindung an bereits bestehende Datenbanken (arXiv, PubMed...) notwendig. Dies vermeidet, daß alle bereits vorhandenen Metadaten nocheinmal erfaßt werden müssen.

- Exportschnittstellen: Um Benutzern die Möglichkeit zu geben auch die erfaßten Metadaten weiterzuverabeiten sollten verscheiedene Exportschnittstellen geschaffen werden.

- Statistikschlüssel: Für die Universitätsverwaltung war es weiterhin erwünscht, die einschlägigen Statistikschlüssel für die jeweiligen Organisationseinheiten auf der Datenbank zu hinterlegen.

Alle diese Funktionen sind in OPUS+ realisiert worden. Hierbei wurde besonderer Wert drauf gelegt ein weitgehend autarkes Subsystem zu realisieren,

[1] Natürlich wäre die umfassende Lösung dieses Problems nur mit Hilfe einer normierten Autorendatenbank möglich. Es ist aber einerseits zu bedenken, daß die Erfassung für den Benutzer noch verständlich und einfach sein muß und daß andererseits eine solche universelle Datenbank derzeit nicht zur Verfügung steht.

das möglichst wenig in die internen Strukturen von OPUS eingreift und das sich so vergleichweise einfach in ein bestehendes OPUS integrieren respektive sich genauso einfach auch wieder entfernen läßt, ohne dabei die Dateninteggrität zu zerstören. Besonderer Wert wurde weiterhin darauf gelegt, die Erfassung der Daten durch den Benutzer so einfach wie möglich zu gestalten und dem Benutzer für ihn Hilfreiche Funktionen zur Verfügung zu stellen um eine möglichst hohe Akzeptanz des Systems sicherzustellen.

Kapitel 2

OPUS+ Installation

Die Installation von OPUS ist im Detail im zugehörigen technischen Handbuch beschrieben. Da sie aber eine Vielzahl von Handgriffen erfordert die zu Änderungen an vielen verschiedenen Stellen im Original-OPUS-Code führt wurde ein Script entwickelt, daß dies übernimmt. Dieses setzt dabei auf einem original-OPUS auf.

2.1 Funktionsprinzip

Das *Perl*-Script InstallOPUS+ benutzt als Ausgangsbasis das original tar-Archiv der OPUS-Distribution, derzeit in Version 3.0.4. Mittels patch werden hier zunächst alle für die Installation spezifischen Änderungen in einzelne tags umgewandelt[1], die dann später einfach ersetzt werden können.

D.h. InstallOPUS+ erzeugt aus der ursprünglichen Distribution zunächst eine einfach konfigurierbare Distribution, welche im Prinzip als Ausgangsbasis für weitere Installationen dienen kann. Enthielte die original-OPUS-Distribution bereits die nötigen tags wäre dieser erste Schritt überflüssig.

In diese angepaßte Installation werden sodann die Programmteile von OPUS+ eingefügt. Hierzu werden einerseits wiederum Patches verwendet um die neuen Modulaufrufe innerhalb von OPUS zu verankern, andererseits werden die neuen Programmodule selbst in den OPUS-Baum integriert.

Diese angepaßte Distribution wird dann im nächsten Schritt in ihr Zielverzeichnis installiert, wobei alle zuvor eingefügten tags durch den installationspezifischen Wert ersetzt werden. Die Installationsspezifischen Werte sind dabei in InstallOPUS+ selbst als Variablen definiert, so daß die gesamte Konfiguration in einer Datei zusammengefaßt wird.

[1] Der hierfür benötigte Patch ist natürlich von der jeweiligen OPUS-Version abhängig und muß für diese jeweils einmalig erzeugt werden.

2.2 Konfiguration

Zunächst müssen in InstallOPUS+ einige globale Variablen den lokalen Gegebenheiten angepaßt werden. Hier ist zunächst einmal anzugeben, wo das Original-OPUS-Archiv liegt und wie es heißt (`$opussourcetgz`), in welchem Verzeichnis die nötigen Patches untergebracht (`$patchdir`) sind und wo das eigentliche OPUS+ zu finden ist (`$newfilesdir`), d.h. die Dateien die zusätzlich in den OPUS-Baum integriert werden müssen.[2]

Sodann ist in absoluten Pfaden zu definieren von wo (`$srcdir`) nach wo (`$target`) OPUS+ installiert werden soll.

Hinweis In `$srcdir` muß berücksichtigt werden, daß das Auspacken des Originalarchivs (`$opussourcetgz`) u. U. ein neues Unterverzeichnis erzeugt. Dieser Pfad muß hier mit angegeben werden!

Schlußendlich werden die OPUS-Scripte definiert die einerseits die Datenbank anlegen (`$installopus`) und andererseits für das Backup verwendet werden (`$backupscript`).

Sind alle diese Einstellungen vorgenommen folgt ein großes globales Hash (%tags) welches die eigentlichen OPUS+-Konfigurationsparameter enthält. Diese setzen sich teilweise wiederum aus Einträgen dieses Hashes selbst zusammen, da der OPUS-Code hier nicht systematisch ist und/oder bisweilen kleine sprachliche Anpassungen nötig sind.

Hinweis Jeder Hashkey entspricht in den gepatchten Sourcen einem gleichnamigen Metatag, der sich durch vorangestelltes %%% sowie ein nachfolgendes %%% auszeichnet. D. H. findet das Installationsprogramm in den zu installierenden Quellen z. B. das tag %%%mysqluser%%% wird dieses durch `$tags{'mysqluser'}` ersetzt werden. Durch diesen Mechnismuß ist es möglich auch für zukünftige Erweiterungen das Installationsprogramm zu nutzen.

Wichtig Aufgrund des Pattern matching das hier zum Einsatz kommt ist es notwendig die Tags korrekt einzugeben, d. h. mit genau 3 %-Zeichen zu öffnen und wieder zu schließen. Andernfalls wrid das Installationsprogramm bei einem nicht korrekt beendeten Tag endlos nach dessen Ende suchen!

Jeder Hash-Key gibt Auskunft darüber wofür die Variable später verwendet wird. Sollte es in OPUS selbst in einem der Konfigurationsfiles eine Variable geben wird als Key deren Name benutzt. Da das gesamte Hash vorbelegt ist sind für jede Variable bereits Beispiele vorhanden, so daß klar sein

[2] Hier werden keine spezifischen Dateien gesucht, sondern alles unterahalb von `$newfilesdir` in den OPUS-Baum kopiert. Hier können also beliebige Dateien liegen, lediglich die OPUS-eigene Verzeichnisstruktur ist einzuhalten.

sollte was sie jeweils bedeuten.

2.3 Interna

Vor jedem Schritt gibt das Hauptprogramm eine kurze Meldung aus, was es als nächstes tun wird. Jede dieser Funktionen ist dabei in einer eigenen Prozedur verpackt. Wärend des einbauens der nötigen Konfigurationsparameter erfolgt weiterhin für jede prozessierte Datei die Ausgabe eines Punktes so daß man erkennen kann, daß das Programm noch arbeitet.

2.3.1 Decompress

Packt durch einen system-Aufruf das Original-OPUS-Archiv aus.

2.3.2 ApplyPatches

Wendet die in `$patchdir` vorliegenden Patches der Reihe nach auf die Ausgepackte OPUS-Distribution an. Am Ende dieser Routine werden die Dateien aus `$newfilesdir` zur OPUS-Distribution kopiert.

Hinweis Für die Reihenfolge in der die Patches angewendet werden ist der Dateiname wichtig. Um eine Reihenfolge zu garantieren sollten die Dateien deswegen 000-Patch0, 001-Patch1 (oder ähnlich) usw. heißen.

2.3.3 ConfigApache

Gibt die nötigen Zeilen für die apache.conf aus. Diese werden jedoch nicht in diese Eingetragen!

2.3.4 ReplaceTags

Ersetzt alle Tags in der übergebenen Zeile durch ihren entsprechenden Wert.

2.3.5 ProcessFile()

Bearbeitet alle Dateien rekursiv. Die Dateien werden mit File::Find gesucht, für jedes bearbeitete Datei wird ein "." auf dem Bildschrim ausgegeben. Weiterhin wird der Zielpfad erzeugt und die neue Datei dorthin geschrieben.

Hinweis Für den neuen Dateinamen wird hierbei lediglich `$srcdir`

durch `$target` ersetzt, weswegen beide Pfade absolute Pfade sein müssen.

Hinweis Dateien mit den Endungen png und gif werden nicht bearbeitet. 

2.3.6 CreateDatabase

Diese Routine legt die Datenbank an und erteilt dem `$tags{'mysqluser'}` die nötigen Rechte zum Zugriff.

Hinweis Diese Routine fragt interaktiv das Root-Passwörd für die Datenbank ab. 

Wichtig Das Password für den mysql-User ist im Script abgelegt! Da dies aber sowieso in den Dateien der endgültigen `OPUS`-Installation im Klartext abgelegt wird ist dies hier nicht kritischer als im `OPUS`-Baum selbst. 

2.3.7 CreateTabels()

Erzeugt mit Hilfe der `OPUS`-Eigenen Scripten die nötigen Tabellen.

Kapitel 3

Module von OPUS+

3.1 Allgemeines

OPUS+ ist eine Erweiterung von OPUS, d. h. es benutzt alles was in OPUS bereits vorhanden ist und ergänzt dieses um einige neue Funktionen. Als echte Erweiterung ist es so angelegt, daß es das original-OPUS so wenig wie möglich verändert und sich so weit wie möglich an die in OPUS verwendeten Konventionen und Vorgehensweisen anpaßt.

3.1.1 Globale Variablen

OPUS selbst legt fast alle Werte in globalen Variaben ab. Da OPUS+ als Erweiterung hierzu gedacht ist, sollte hier nicht der Versuch unternommen werden das gesamte Codelayout zu ändern. Allerdings werden alle globalen Variabeln in variables.inc zusammengefaßt. Diese Datei liest auch alle neuen Textstrings ein und setzt die zugehörigen Variablen enstprechend. Es wurde versucht möglichst eindeutige sprechende Namen zu verwenden ohne diese sinnlos lang zum machen.

docvariables.inc enthält die nötigen global Deklarationen für die Dokumenteigenschaften. Ein include beider Dateien führt also dazu, daß alle Variablen mit vernünftigen Werten belegt zur Verfügung stehen.

Weiterhin wird im gesamten Code darauf verzichtet für Schleifenzähler komplizierte Variablennamen einzuführen. Die Variablen `$i`, `$j`, `$k` ... dienen hierzu und haben im sonstigen Code keine Bedeutung als als Index für ein Feld. Allerdings wurde wo immer möglich auf die Verwendung numerischer Indicees verzichtet, da PHP intern alle Felder als assoziative Arrays (Hashes) behandelt und man hier besser eine Konstruktion der Form

```
foreach ($feld as $key=>$value) {}
```

verwendet. Sind zwei Variablen über den `$key` verknüpft, kann so das gleiche Erreicht werden wie bei einem numerischen Schleifenzähler.

3.1.2 Front- und Backends

Die Meisten neuen Formulare ermöglichen es dem Benutzer zusätzliche Felder anzufordern oder Daten aus anderen Quellen (andere interne Tabellen, externe Datenbanken) nachzuschlagen. Aus diesem Grund enthalten die verwendeten Formulare oft mehrere Buttons die der Benutzer anwählen kann. Diese sind als submit-Buttons ausgeführt mit entsprechenden Einträgen in den name und value Feldern.

Um die Behandlung der so erzeugten Ereignisse übersichtlicher zu gestalten wird die Behandlung der Ereignisse in eine eigene Datei verschoben. Diese heißen wie das eigentliche Formular mit angehängtem "backend". Z. B. heißt das Backend für das NewWorkgroup-Formular NewWorkgroupbackend.php. D. h. alle Ergeinisse die in NewWorkgroup.php generiert werden, werden von NewWorkgrupbackend.php behandelt. Verzweigt eine Aktion nicht auf eine externe Seite führt das Backend ein erneutes Laden des eigentlichen Dialogs aus, sobald die nötigen Änderungen der Variablen vorgenommen wurden. Hierzu wird die Funktion RefreshForm() benutzt. Dieses Prinzip wird in Abschnitt 3.4.3 beispielhaft erläutert.

Hinweis Da das Formular zum Einstellen von neuen Dokumenten vergleichswiese komplex ist besteht es aus den Dateien submitform.php (1. Seite) submitform2.php (2. Seite) und submitbackend.php.

3.1.3 Konventionen im Code

Kommentare Alle neuen Funktionen haben einen Kommentarheader, der die übergebenen Argumente kurz erklärt und beschreibt was eine Funktion tun soll. Weiterhin wurde versucht im Sourcecode so viele Kommentare anzubringen, daß es möglich ist, dem Programmablauf zu folgen. Kommentare, die zur Dokumentation dienen werden mit // eingeleitet. Temporär auskommentierter Debug-Code hingegen mit //–//, nicht benötigte Teile bis zu ihrer engültigen Entfernung mit # bzw. ##, ### etc. um verschiedene Ebenen deutlich zu machen.

Namen, Variablen Es wurde Versucht die Dateinamen so zu wählen, daß die Funktion des jeweiligen Moduls daraus hervorgeht. Gleiches gilt auch für Variablennamen. Auf die Verwendung von Unterstrichen wurde weitestgehend verzichtet. Alle Variablen werden deklariert und vorbelegt.

PHP Der gesamte Code wurde zwar mit PHP5 entwickelt, sollte aber auch mit PHP4 lauffähig sein, da auf die Verwendung neuer PHP5-Funktionen bewußt verzichtet wurde.

Schleifenzähler Einbuchstabige Variablen ab `$i` dienen nur aus Schleifenzähler, bzw. Indicees für Felder in Schleifen.

3.2 Übernahme aus anderen Datenbanken

Um das Einstellen von Dokumenten zu vereinfachen steht eine Schnittstelle zu anderen OAI2-kompatiblen Repositorien und zu PubMed zur Verfügung.

3.2.1 OAI2-Repositorie-Repositorienn

arXiv.php repräsentiert die Import-Schnittstelle zu arXiv.org und den anderen OAI-Repositorien. Auch wenn arXiv über sein eigenes XML-Format deutlich umfangreichere Informationen zur Verfügung stellen kann wird hier die normale OAI2-Schnittstelle verwendet und lediglich ein Dublin-Core-kompatibler Metadatensatz angefragt. Der Grund für diese Entscheidung war, daß derzeit der arXiv-eigene XML-Record noch nicht als endgültiges Format vorliegt und so noch weitere Änderungen zu erwarten sind.

Auf der durch arXiv.php erzeugten Seite kann der Benutzer die jeweilige Record-ID eingeben, für die dann die entsprechenden Metadaten angefordert werden. OAI2 unterstützt derzeit keine Suche nach Metadaten, so daß die Ermittlung der Record-ID dem Benutzer überlassen bleibt. In den Teilgebieten der Wissenschaft, in denen arXiv.org als fester Bestandteil integriert ist (z. B. Physik, Mathematik, Informatik etc.) stellt dies kein Problem dar, da der Wissenschaftler mit diesem System vertraut ist und in weiten Teilen diese IDs in Literaturverzeichnissen Verwendung finden.

Da das Anfordern der Daten von arXiv.org über deren OAI2-Schnittstelle geschieht kann man natürlich andere, OAI2-konforme Repositorien auf die gleiche Weise abfragen. Will man ein weiteres Repositorium zum derzeitigen Satz (arXiv, Pubmed Central, Biomed Central und RePEc) hinzufügen, muß lediglich ein entsprechender Eintrag im Feld `$oaiurls` der Datei include/oaiservers.inc angelegt werden. Hierzu sind die Werte aus Tabelle 3.1 anzugeben.

Index	Eintrag	Beispiel
0	URL der OAI-Schnittstelle	http://arxiv.org/oai2
1	OAI-Prefix	oai:arXiv.org:
2	Formatanforderung	oai_dc
3	Name des Repositoriums	arXiv

Tabelle 3.1: Definition einer OAI2-Ressource

Wichtig

- Der Name des Repositoriums muß aus technischen Gründen mit dem Hashkey des Repositoriums in `$oaiurls` übereinstimmen!

- Derzeit wird nur das Dublin-Core-Format (oai_dc) korrekt behandelt. Allerdings sind Erweiterungen sehr einfach möglich, wie die Unterstützung von NCBI PubMed zeigt.

Das erzeugte Formular schließlich enthält eine Drop-Down-Liste mit den Namen der unterstützen Server. Daneben ein Eingabefeld für die entsprechende ID.

Der Button [START] schließlich startet die Abfrage. Dies ist in der Funktion QueryOAI2 implementiert, die einen enstprechenden XML-String zurückliefert, der von der Funktion ParseOAIResult in die Variable `$dublincore` geschrieben wird. Hierbei wird ein eventuell nötiges recoding der Zeichen mit Hilfe des PHP-Modul iconv vorgenommen, so daß alle Strings in ISO-8859-1 vorliegen.

Das Ergebnis zur Übernahme in OPUS+ wird in einem HTML-Formular dargestellt. Hier kann der Benutzer noch Änderungen vornehmen, oder er wählt [WEITER MIT AUSWAHL], was die Daten an das OPUS+-System weitergibt und das entsprechende Formular vorausfüllt.

Da das System derzeit die Kenntnis der eindeutigen Dokument-ID voraussetzt ist diese Funktion in Fachbereichen, in denen die entsprechenden Dokument-IDs derzeit nur wenig Beachtung finden primär von geringerem Nutzen. Allerdings kann man diese Identifikationsnummern i.d.R. auf den Webseiten der jeweiligen Repositorien, die dann auch über weitreichendere Suchfunktionen verfügen, in Erfahrung bringen.

Um hier eine komfortablere Nutzung zu ermöglichen wäre natürlich eine Suche nach Metadaten hilfreich. Ideal wäre hierzu wohl eine Anbindung an *BASE*, *OAIster* und *citebase.org*, allerdings steht derzeit nur bei citebase.org ein xml-Format zur Verfügung, welches sich derzeit noch in experimentellem

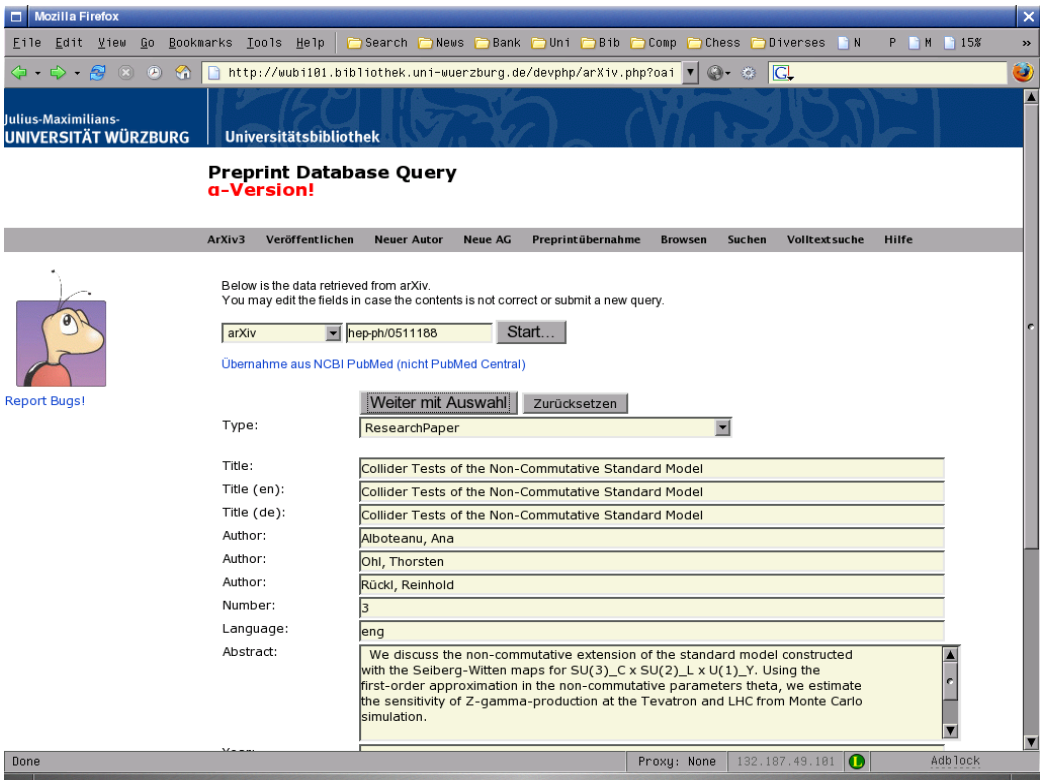

Abbildung 3.1: Das Preprint-Interface von OPUS+

Stadium befindet. Aus diesem Grund ist eine solche Anbindung bisher nicht realisiert.

Die Umstellung auf einen erweiterten XML-Datensatz ist sehr einfach möglich, in dem man das Feld `$dublincore` in include/oaiservers.inc entsprechend ergänzt. Der Eingesetzte XML-Parser liefert alle in `$dublincore` angegebenen XML-Elemente in einem Hash mit den Elementnamen als Schlüssel zurück. Aus diesem Grund ist dieser nicht auf die Verwendung des Dublin-Core beschränkt.

3.2.2 NCBI PubMed

NCBI PubMed stellt seine Daten nur in einem eigenen XML-ähnlichen Format zur Verfügung und benutzt hierzu nicht eine OAI2-Konforme Schnittstelle sondern ein eigenes Toolset mit Namen entrez.

pubmed.php implementiert hier das für das Abfragen des Repositoriums nötige Formular sowie die nötigen XML-Parser. Da PubMed außerdem keine Dublin-Core-kompatiblen Datensätze zurückliefert muß hier ein eigenes

Set ausgelesen werden. Außerdem enthält dieses pseudo-XML-Format keine Zeichensatzinformationen. Es scheint allerdings zumindest derzeit immer ISO-8859-1 zu sein, so daß eine Übernahme in OPUS+ direkt erfolgt.

Da PubMed u.a. auch die Journalreferenzen mit zurückliefert könnte man diese im Prinzip zur Weiterverarbeitung heranziehen. Da aber normierte Namen für die Zeitschriften gewünscht sind werden diese Daten derzeit ignoriert und aus dem Ergebnisrecord von PubMed lediglich der Dublin-Core-Subset weitergereicht. Der Parser selbst speichert diese Zusätzlichen Informationen aber bereits ab, da hier nichts anderes zu machen ist, als die Variable `$xmltags` entsprechend zu ergänzen. Über diese Variable sind weitere Felder problemlos auszulesen, ohne daß dies größere Änderungen am Code erfordert. Der XML-Parser liefert die Ausgelesenen Felder dann in einem Hash `$dcresult` mit den in `$xmltags` eingetragenen Schlüsseln zurück.

3.3 Interne Autoren

Um personalisierte Listen anbieten zu können ist in irgendeiner Form eine Identifikation des Autors erforderlich. In OPUS+ sollte hierbei auf eine explizite Anmeldung am System mit Benutzerkennung und Password verzichtet werden. Trotzdem wird zwischen internen und externen Autoren unterschieden.

Interne Autoren sind solche, die dem System bereits bekannt sind und von denen das System weiß, welchen Organisatioseinheiten diese angehören. Hiermit ist bereits festgelegt, daß interne Autoren diejenigen Autoren sind, die an der Universität selbst arbeiten. Dieses Konzept ist neu in OPUS+. Es wurde nötig, da eine Möglichkeit geschaffen werden sollte einerseits die eingestellten Dokumente den verschiedenen Organisationseinheiten der Universität zuordnen zu können und andererseits personalisierte Dienste anzubieten (z. B. eine eindeutige Liste aller Papiere eines Autors).

3.3.1 Anmelden eines Autors

NewAuthor.php ist die Schnittstelle zum Anmelden eines neuen Autors am System. Hierbei sollte bewußt keine echte Authentifizierung erforderlich sein, sondern es sollte sich lediglich der Autor am System bekannt machen können, seine Kontaktdaten und Zuordnungen zu den einzelnen Organisationseinheiten hinterlegen können. Hierbei wird eine minimale Anzahl von Daten erhoben:

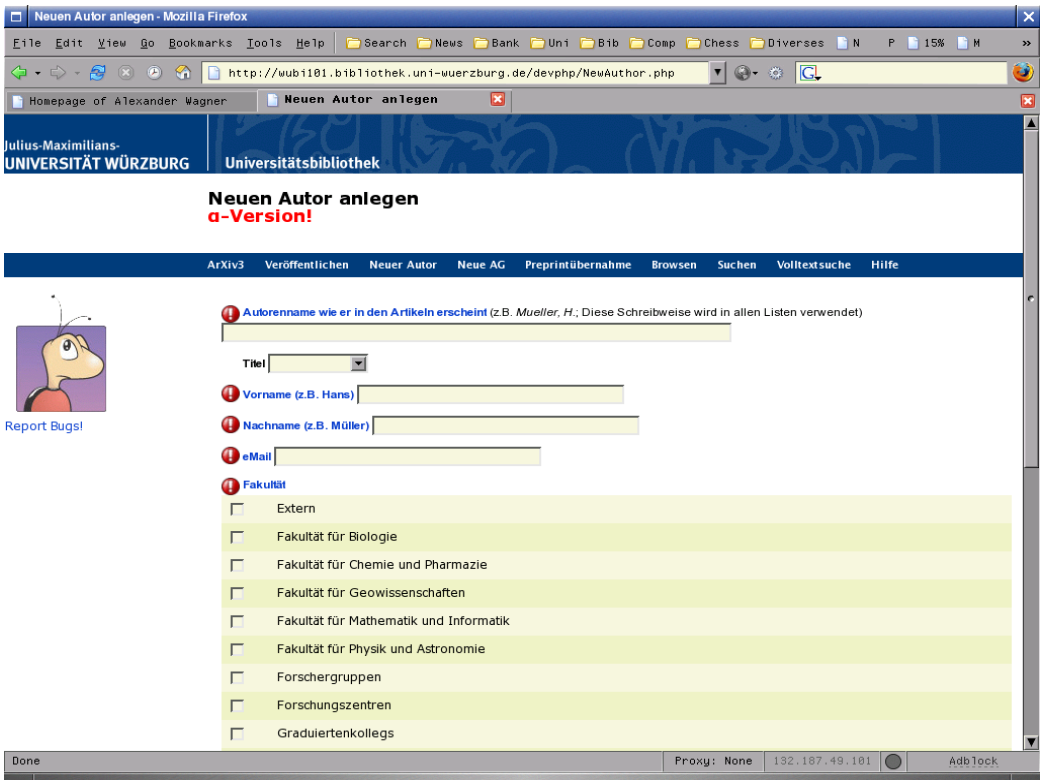

Abbildung 3.2: Anmelden eines Autors in OPUS+

Feld	Inhalt	Beispiel
CREATOR_NAME	Name auf der Veröffentlichung	Mueller, H.
TITEL	Akad. Titel	Prof. Dr.
VORNAME	Vorname	Hans
NACHNAME	Nachname	Müller
EMAIL	eMail-Adresse	h.mueller@...

Tabelle 3.2: Daten der internen Autoren

Das Feld mit dem `$creator_name` hat in sofern eine Sonderrolle, als die dort angegebene Schreibweise überall in OPUS verwendet wird. Diese Schreibweise wird immer dann abgefragt, wenn es um einen Autorennamen geht. Dieses Feld enstpricht in seiner Bedeutung dem internen OPUS-Feld `$creator_name` und ist eingeführt worden, da viele Autoren z. B. Umlaute in ihrem Namen nicht literal schreiben, sondern "USA-kompatibel" umschreiben. Auch ist es so möglich zumindest halbautomatisch eine Liste von Autoren mit den internen Daten abzugleichen um interne und externe Autoren zu unterscheiden.

Beim Anlegen eines neuen autors erfolgt die Titelauswahl erfolgt über eine Drop-Down-Liste, welche alle möglichen Einträge aus der Datenbank abfragt. Hierbei ist eine landessprachliche Variante möglich. Die Tabelle TITLE die hierbei abgefragt wird muß vom Systembetreuer passend gefüllt werden. Der Autor hat keine Möglichkeit hier Einträge zu ergänzen.

Weiterhin enthält das Anmeldeformular eine Liste mit allen Fakultäten der Universität. Der Autor kann hier über Checkboxen alle diejenigen auswählen, denen er angehört. Letzters ist so zu verstehen, daß alle die Fakultäten ausgewählt werden sollen, zu denen sich der Autor zugehörig "fühlt" oder mit denen der Autor regelmäßig zu tun hat. Z. B. würde ein Physiker, der mit dem Klinikum zusammenarbeitet hier sowohl die Physik als auch die Klinik angeben können, auch wenn er nicht in der Klinik angestellt ist. Ein klick auf den [WEITER]-Knopf blendet, dann, sofern mindestens eine Fakultät gewählt ist, alle anderen aus und zeigt die Liste der zu den gewählten Fakultäten gehörigen Institute an. Hier kann wiederum ausgewählt werden, und man gelangt zu den Lehrstühlen und von dort zu den Arbeitsgruppen.

Diese Sequenz von Listen wird von der Routine SelectGroupWhizard erzeugt. Die Daten bis zur Lehrstuhlebene werden hierbei direkt von der Datenbank zur Verfügung gestellt. Bei den Arbeitsgruppen wird eine Eingabe durch die Benutzer gewünscht, so daß diese Liste zunächst immer leer ist. Dafür erhält der Benutzer am Ende des Dialogs eine Möglichkeit eine neue Arbeitsgruppe anzulegen, die dann später in diesem Dialog auftauchen wird. (S.a. 3.3.3). Hat ein Autor alle Felder gültig ausgefüllt werden die Daten in die Datenbank übernommen. Außerdem wird hierbei eine eindeutige numerische Autoren-ID vergeben.

Wichtig Die Autorendaten werden direkt in die engültigen Tabellen übernommen, sie stehen dem Benutzer unmittelbar zur Verfügung und werde nicht zunächst von einem Bibliothekar geprüft.

3.3.2 Editieren von Autorendaten

Hier wird der Benutzer zunächst nach seinem `$creator_name` gefragt. Auf Grund seiner Eingabe bekommt er, sofern mehr als ein Treffer vorliegt, eine Liste zur Auswahl des "passenden" Eintrags. Auch der selbe Autor wird durchaus mehrmals auftauchen, wenn er mehreren Einheiten angehört. In diesem Fall ist es intern gleichgültig welchen Eintrag man auswählt, da ab hier nur noch die interne numerische Autorennummer verwendet wird.

Nach Ausawahl des richtigen Autors werden alle derzeit erfaßten Daten ausgegeben. Die Zuordnungen erscheinen dabei als Tabelle mit einem Symbol daneben um diese zu editieren. Ein Anklicken des Symbols z. B. neben

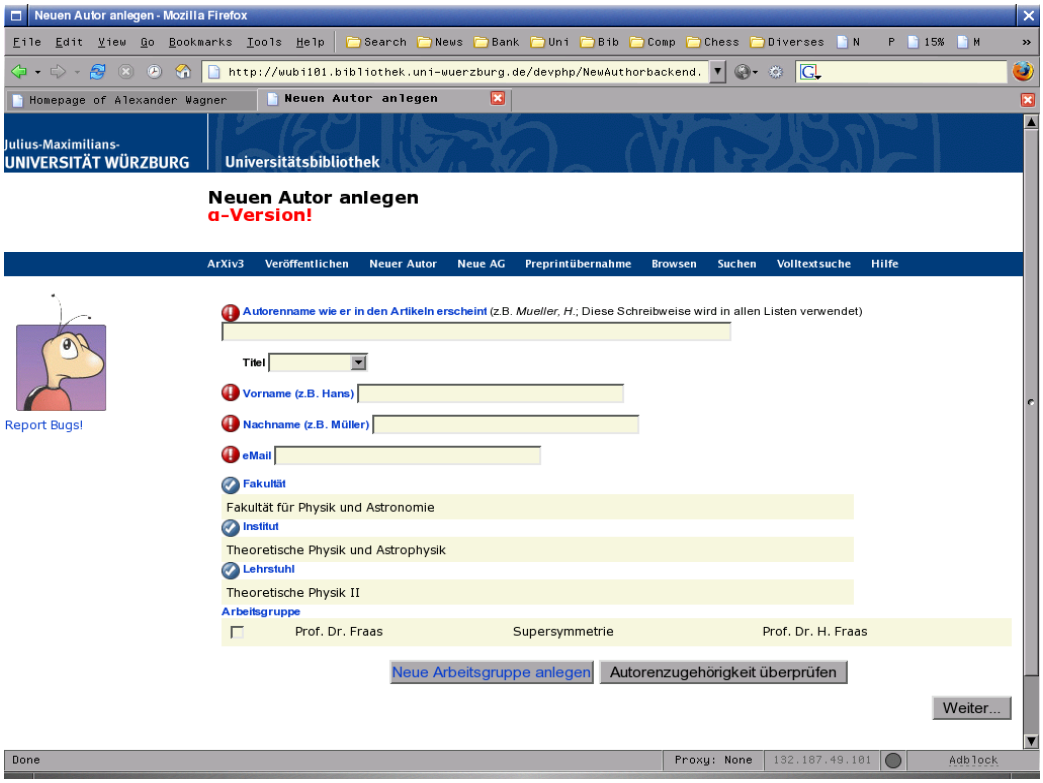

Abbildung 3.3: Anmelden eines Autors in OPUS+ nach Auswahl aller Gliederungesebenen.

der Institutsliste führt dann dazu, daß alle zu den derzeit ausgewählten Fakultäten gehörigen Institute als Checkboxliste angezeigt werden, in denen der Autor dann zusätzliche Institute auswählen oder vorhandene abwählen kann. Ein klick auf den Knopf [WEITER] klappt diese Liste dann wieder zusammen und zeigt sie mit den neuen Zuordnungen an wie zuvor.

Dieser gesamte Auswahlmechanismus wird über die Funktion EditableAssociationList() realisiert. Damit diese Funktion weiß, welcher Teil der Liste gerade editiert werden soll und welcher statisch angezeigt wird, werde in in `$_REQUEST` eine der Variablen `$EFaculty`, `$EInst`, `$EChair` und `$EGroup` gesetzt, um den zugehörigen Teil editierbar zu machen. Das setzen dieser Variablen geschieht nach anklicken der entsprechenden Editiersymbole in EditAuthorbackend.php, wobei das wählen eines der Symbole jeweils ein Button-Event erzeugt. Aufgrund dieses Events wird dann die entsprechende Variable in `$_REQUEST` gesetzt und sodann mit RefreshForm das Formular EditAuthor.php neu geladen.

Hat der Benutzer alle Änderungen durchgeführt kann er durch anklicken des günen Hakens am Ende der Seite ein Eintragen der Daten auf der Datenbank veranlassen. Die nötigen Aufrufe hierzu sind ebenfalls in EditAuthorbackend.php für das Event Enter definiert. Die Personendaten werden dabei durch UPDATE aufrufe auf die AUTHOR-Tabelle realisiert.

Um das Eintragen der Zugehörigkeiten möglichst einfach zu gestalten werden zunächst alle Zugehörigkeiten des Autors über seine interne numerische ID gelöscht. Sodann wird in einer Schleife jede Fakultät mit jedem Institut, Lehrstuhl und Arbeitsgruppe verknüpft eingetragen. Aufgrund der Idexierung der Tabelle MEMBERSHIP filtert hierbei dann die Datenbank eventuelle Doubletten heraus. Da Aufgrund der erzeugten Indexfehler der Aufruf von mysql_query einen Fehler zurückliefert, was in den `OPUS`-Klassen einen Programmabbruch auslöst, muß an dieser Stelle direkt mysql_query aufgerufen werden und es darf nicht $opus->query() benutzt werden!

3.3.3 Arbeitsgruppen

Die Arbeitsgruppen der Universität sollen von den jeweiligen Benutzern individuell angelegt werden. Da hierbei eineindeutige Zuordnugnen von Dokumenten zu Gruppen gewünscht sind, wird ein Editieren der Arbeitsgruppendaten aber nicht angeboten. Wenn sich eine Arbeitsgruppe ändert, so soll vom Benutzer eine neue Arbeitsgruppe eingerichtet werden.

Der Dialog NewWorkgroup.php erzeugt zunächst eine Liste aller Fakultäten, die bekannt sind. Da eine Arbeitsgruppe eine logische Einheit unterhalb eines Lehrstuhls darstellt gehört sie nur zu einer Fakultät, so daß dem Benutzer eine Liste von Radioknöpfen präsentiert wird, aus denen er die passende

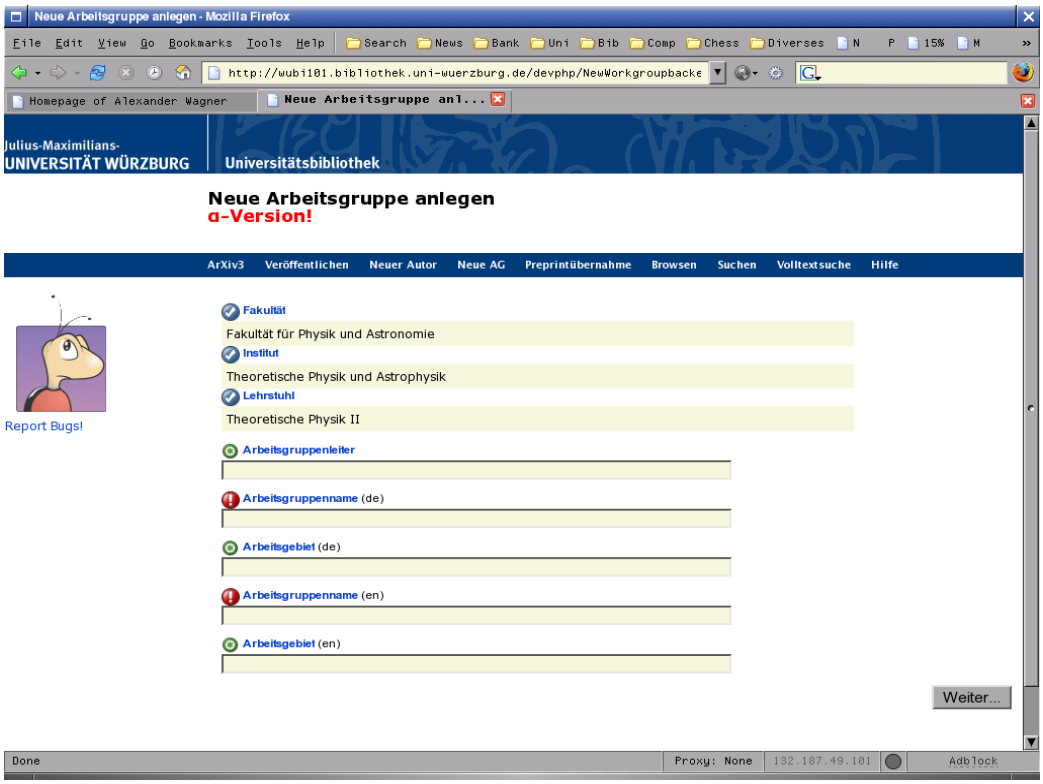

Abbildung 3.4: Anleden einer Arbeitsgruppe

auswählen kann. Die Auswahl von [WEITER] übernimmt die Fakulät und präsentiert eine Lister aller zu dieser gehörigen Institute. Nach Auswahl und erneutem [WEITER] bekommt man die Lehrstühle entsprechend präsentiert. Wurde der Lehrstuhl gewählt kann kann der Benutzer in normalen Eingabefeldern den Arbeitsgruppenleiter das Arbeitsgruppengebiet und den Arbeitsgruppennamen eingeben. Diese Felder werden in allen unterstützten Landessprachen angeboten. Wurden alle Pflichtfelder ausgefüllt, erscheint nach auswahl des Knopfes [WEITER] die gesamte eingabe erneut und ein großser grüner Hanken durch dessen auswahl der Benutzer die Übernahme der Daten in die Datenbank startet. Die ID-Nummer der Arbeitsgruppe wird hierbei automatisch erzeugt und fortlaufend vergeben.

3.4 Einstellen neuer Papiere

Diese ist natürlich die zentrale Funktion von OPUS. In OPUS+ wird diese weitgehend übernommen, und um einige zusätzliche, v.a. für Zeitschriftenaufsätze relevante Metainformationen ergänzt. Da sich die nötigen Änderungen hier

als etwas umfangreicher erweisen wird von OPUS+ der gesamte Submitdialog ersetzt. U. a. werden hierbei einige Felder umsortiert und eine zweite Seite eingeführt. Die erste Seite des Dialogs submitform.php enthählt hierbei die Felder die der Autor auf jeden Fall ausfüllen soll oder muß. Auf der zweiten Seite submitform2.php werden dann zusätzliche Felder angeboten, die vorhanden sein können aber nicht müssen. Z. B. findet sich hier die Eingabe der Journalreferenz, aber auch die Vergabe normierter Schalgwörter wurde von der ersten Seite als Pflichtfeld hierher verschoben und ist fakultativ.

3.4.1 Allgemeine Metadaten

Die allermeisten dieser Felder werden bereits vorausgefüllt, wenn eine Übernahme der Metadaten aus einer externen Quelle (z. B. *arXiv.org*, *PubMed*) möglich ist. Weiterhin sind fast alle Felder auf dieser Seite Pflichtfelder. Der Dialog selbst ist in submitform.php realisiert. Das zugehörige Backend ist submitbackend.php.

Autoreneingabe

Da eindeutige Zuordnugnen der Papiere zu den Autoren gewünscht werden unterscheidet sich die Auswahl der Autoren in OPUS+ von der in OPUS. Im Eingabefeld *Verfassername* wird zunächst, wie in OPUS, der interne $creator_name erfasst. Dieser Name muss für interne Autoren mit dem in der Autorenanmeldung benutzen übereinstimmen, bzw. OPUS+ erzwingt diese Übereinstimmung nach einem Nachschlagen des Autors.

Ist der eingegebene Verfasser Mitglied der Universität, so muß dies der Anwendung mitgeteilt werden. Es könnte sich ja auch um einen Namensgleichen externen Autor handeln. Aus diesem Grund muß nach der Eingabe des $creator_name oder eines Teils davon, der Knopf [Autor aus Liste wählen] angewählt werden. Anhand des eingegebenen Namens(teils) sucht OPUS+ alle Autoren der Universität, die so oder so ähnlich heißen. (Die Datenbank benutzt hierbei ein rlike, matched also nach Regular Expressions.)

Hierbei ist es möglich, daß ein Autor mehrfach erscheint. Dies geschieht, sobald er zu mehreren Fachbereichen gehört. Das Verhalten hier ist unumgänglich, da es sich ja auch um eine Namensgleichheit innerhalb der Universität handeln könnte, bzw. es für einen Autor verwirrend ist, wenn lediglich eine seiner Zuordnungen auftaucht. (Es wäre nicht (einfach) zu steuern, daß dies seine "Hauptzugehörigkeit" ist, so daß ein Biophysiker z. B. eine Zugehörigkeit zur Medizinischen Fakultät angezeigt bekäme, aber keine zur Physik. Aus diesem Grund stellt das doppelte Aufführen des gleichen Autors

hier das kleinere Übel dar.)

Wichtig Gibt man nichts ein, so findet die Datenbank derzeit alle internen Autoren, was eine relativ umfangreiche Liste ergeben dürfte. Ein Einschränken der Auswahl über den Nachnamen ist also höchst sinnvoll. Alternativ kann man an dieser Stelle die für leere Strings verwendete regexp von .* in etwas einschränkenderes ändern.

Wurden bei der Übernahme der Daten aus einer externen Datenbank mehrere Autoren übertragen, so kann man durch Auswahl des Knopfes [AUTORENZUGEHÖRIGKEIT ÜBERPRÜFEN] die Anwendung veranlassen, bei jedem eingegebenen Autorennamen nach einem passenden internen `$creator_name` zu suchen und den Autor entsprechend zuzuordnen. Hierbei wird auch die Liste der Fakultäten, Institute, Lehrstühle und Arbeitsgruppen entsprechend angepaßt (s. a. GetAffiliations()). Ist bei dieser Auflösung ein Autor als "Extern" erkannt, so folgt zwingend die Zuordnung des Papiers zur Einheit "Extern".

Die beiden Buttons [AUTORENFELD HINZUFÜGEN] und [AUTORENFELD ENTFEREN] fügen jeweils eine zusätzliches Autorenfeld hinzu oder entfernen das letzte Feld.

Sind alle Autoren eingegeben kann man mit dem Knopf [WEITER] die Zurodnungsliste des Papiers eintragen. Hierbei werden für die Fakultäten usw. jeweils die Zugehörigkeiten aller Autoren vorgeschlagen. Gehört ein Autor z. B. zur Fakultät für Physik und zur Fakultät für Medizin und ist ein zweiter Autor extern, so wird im Fakultätsfeld fol1gerichtig Extern, Physik und Medizin angeboten. Nach Auswahl von [WEITER] werden dann jedoch alle Institute die zu den selektierten Fakultäten gehören, unabhänig davon ob einer der Autoren dort Mitglied ist angeboten. Es werden allerdings die Institute zu denen ein Autor gehört vorselektiert. (Ein Autor aus einer Fakultät könnte z. B. einen Vortrag im Rahmen einer Fakultätsweiten Veranstaltung einstellen, der allen Instituten zugeordnet sein soll.) Die Vorauswahl bei den Lehrstühlen und Arbeitsgruppen erfolgt entsprechend.

Hinweis Aufgrund der Vorselektion sollte nach dem Erkennen der internen Autoren in fast allen Fällen mehrfaches auswählen des Knopfes [WEITER] eine korrekte Zuordnungsliste des Papiers erhalten.

Die Eingabefelder nebst den zugehörigen Könpfen werden mit der Funktion AuthorInputFields() erzeugt, welche lediglich die abstraktere Routine CreatorInputFields() mit den für Autoren passenden Parametern aufruft.

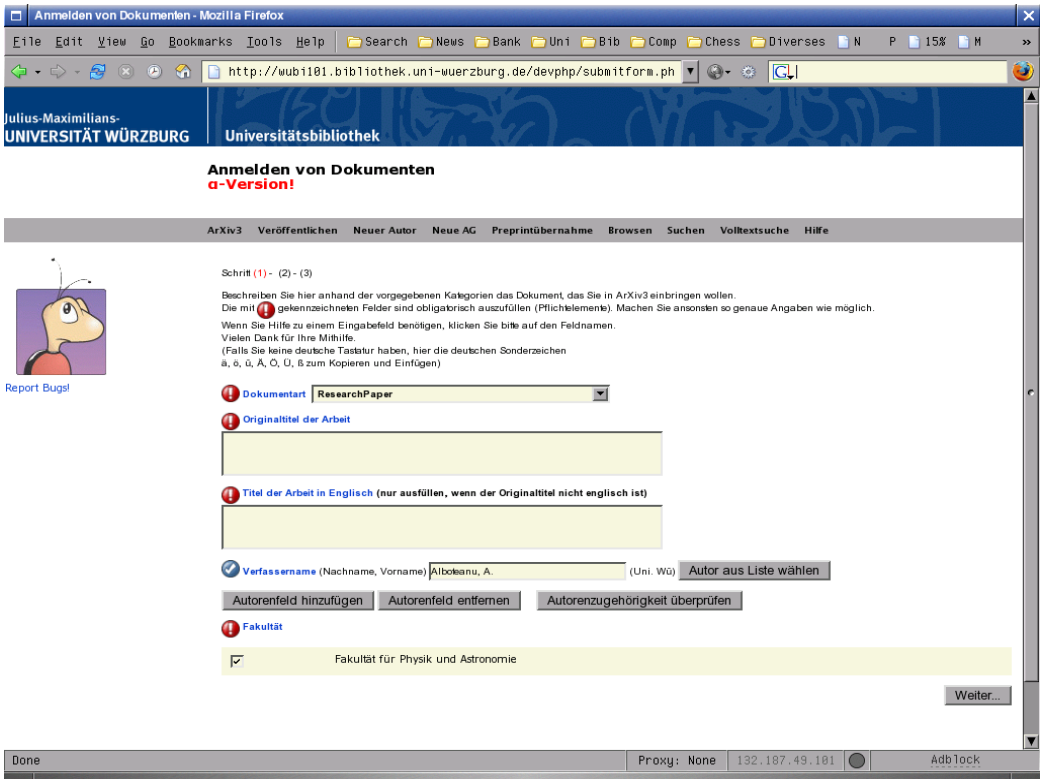

Abbildung 3.5: Teil des Formulars zum einstellen von Dokumenten.

"Contributors"

Da in einigen Fachbereichen Herausgeber etc. eine Ähnliche Bedeutung haben wie in anderen Fachbereichen die Autoren, werden diese in OPUS+ analog behandelt. Ihre Eingabe erfolgt genauso wie bei Autoren getrennt nach internen und externen Herausgebern, in einer zusätzichen, zu OPUS_AUTOR anlaogen Tabelle OPUS_CONTRIBUTOR. Auch die Behandlung der temp-Tabellen ist analog.

Die Eingabefelder nebst den zugehörigen Könpfen werden mit der Funktion ContributorInputFields() erzeugt, welche lediglich die abstraktere Routine CreatorInputFields() mit den für Autoren passenden Parametern aufruft.

Andere Pflichtdaten

Nachdem die Zuordnung des Papiers zu den Organisationseinheiten abgeschlossen ist erscheinen alle weiteren Eingabefelder für Metadaten wie z. B. der Abstract. Bei der Übernahme von einer externen Datenbank ist auch hier fast alles vorausgefüllt. Andernfalls wird lediglich die mail-Adresse des Absenders mit der des (internen) Erstautors vorbelegt. Die Auswahl des Knopfes [WEITER] am Ende des Fomulars führt dann zur Eingabe fakultativer Metadaten

3.4.2 Zustätzliche Metadaten

In submitform2.php werden vom Autor zusätzliche Metadaten abgefragt. Standardmäßig erscheint hier nur eine Liste mit Knöpfen, deren Anwahl entsprechende Eingabefelder "ausklappt", die dann gefüllt werden müssen.

Angaben zur Zeitschrift

Handelt es sich um einen Aufsatz, der in einer Zeitschrift erschienen ist, so können die nötigen bibliographischen Angaben hiermit eingegeben werden.

Zunächst wird hierzu der Name der Zeitschrift erforderlich. Da normierte Namen für die weitere Verarbeitung erforderlich sind kann der Autor hier den Namen oder einen Namensbestandteil eingeben. Alternativ besteht die möglichkeit zur Angabe der Indexnummer (derzeit die *ISSN*), welche durchsucht wird, wenn keine Zeitschrift mit passendem Namen gefunden wird. Schlußendlich kann auch die internationale Abkürzung eingegeben werden, so daß nach dieser gesucht wird. Der Knopf [ZEITSCHRIFT NACHSCHLAGEN] lädt dann eine Liste aller Zeitschriften, die auf das eingegebene Kriterium passen. Analog zur Auswahl eines internen Autors kann sodann die passende

Zeitschrift gewählt werden. Ist sie nicht in der Liste enthalten besteht die Möglichkeit die Zeitschrift am System anzumelden (s. a. Abschnittt 3.7).

Nach Auswahl der Zeitschrift wird Anhand der lokal vorhandenen Daten zu dieser Publikation zunächst die Sherpa/RoMEO-Datenbank nach den bekannten Autorenrechten abgefragt und das Ergebnis dem Autor in Kurzform angezeigt. Im erscheinenden, den RoMEO-Farben enstsprechend farbig hinterlegten, Feld werden die nötigen Links zum Verlag, zur Sherpa/RoMEO-Datenbank selbst, sowie wenn diese dort bekannt ist, zur Autorenlizenz nebst einer Kurzfassung der Rechte angegeben. Ist eine Zeitschrift noch nicht in Sherpa/RoMEO erfaßt, so erhält der Autor einen Link zu dieser Datenbank, über den er, falls bekannt, die nötigen Informationen zur Verfügung stellen kann.

Hinweis U. u. schlägt das automatische durchsuchen von Sherpa/RoMEO fehl, da verschiedene Schreibweisen für die Zeitschriftennamen in Gebrauch sind z. B. *Physical Review*. OPUS+ listet die Zeitschrift dann zunächst als *Sherpa Red*, es kann aber hilfreich sein, manuell die Sherpa/RoMEO-Datenbank unter der angegebenen URL zu konsultieren.

Ist die Zeitschrift gewählt sind das Jahr, Volume, Ausgabe und Seiten anzugeben. Ist der Digital Object Identifier (DOI) bekannt, so kann dieser eingebeben werden um einen Link auf die publizierte Ausgabe des Artikels zu ermöglchen.

Hinweis Einige Verlage verlangen einen Link zur publizierten Version. Dies ist nur möglich, wenn ein DOI angegeben wird!

Preprint Nummern

In einigen Fachbereichen ist es üblich, jedes erschienene Papier mit einer institutseigenen Nummer zu versehen. Hierfür sind die beiden Felder für interne und externe Preprintnummern vorgesehen.

Sonstige

Die anderen Felder entsprechen denen in OPUS bereits bekannten.

3.4.3 Das Backend

Das Backend für den Sumitdialog ist der Prototyp für alle anderen Backend-Scripte und gleichzeitig auch eines der komplexesten, da hier zahlreiche Events behandelt werden müssen: Einerseits lösen alle Buttons in submitform.php

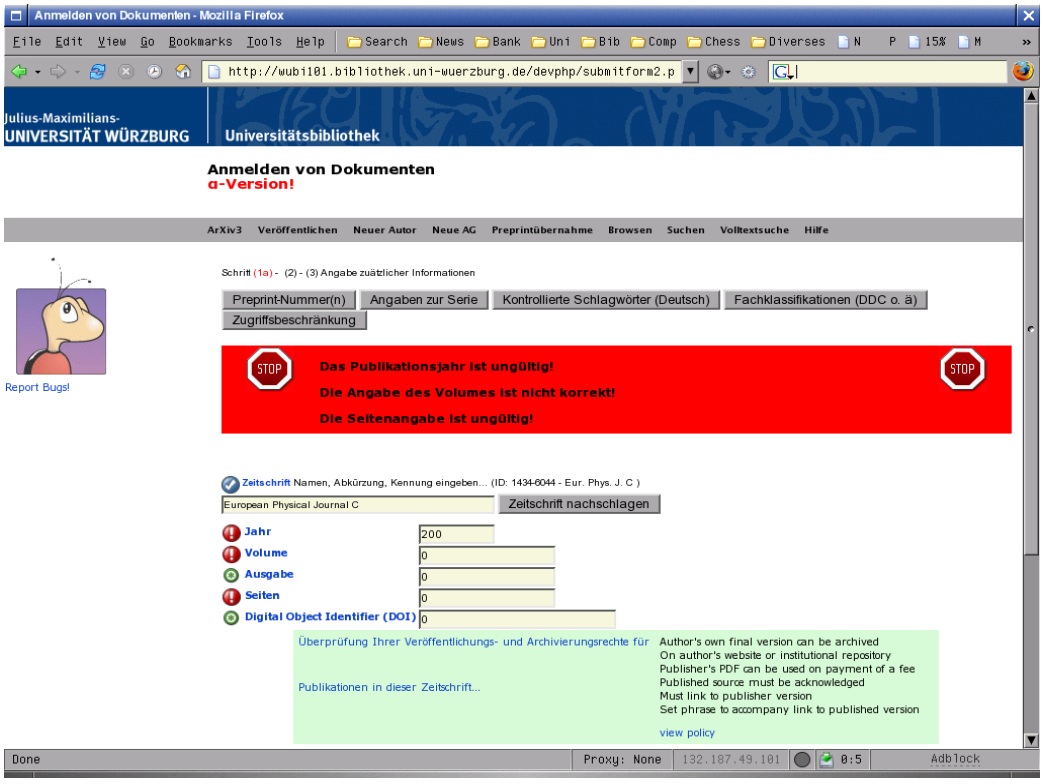

Abbildung 3.6: Angaben zur Zeitschrift mit Antwort von Sherpa/RoMEO.

eigenständige Events aus, andererseits werden hier auch die aus submitform2.php resultierenden Events behandelt. Schlußendlich ersetzt dieses Backend noch die Funktion von meta.php aus OPUS.

Im wesentlichen bestehen die Backends in OPUS+ aus einer reinen if/elseif Anweisungen, die jeweils überprüfen ob im *superglobalen Array* $_REQUEST entsprechend den ausgelösten Events Variablen gesetzt sind.

Wichtig Je nachdem ob graphische oder textuelle Buttons verwendet werden erzeugen verschiedene Browser unterschiedliche Events. Mozilla-artige Browser erzeugen zwar auch bei graphischen Buttons immer ein Event mit dem Buttonnamen, IE jedeoch übergibt dann nur die angeklickte x- und y- Koordinate. Da diese Events jedoch bei Textknöpfen nicht erzeugt werden muß man immer auf beide Möglichkeiten prüfen! Die typische Codesequenz sieht folgendermaßen aus:

```
if ( isset ($_REQUEST['Refresh_x']) ||
  isset ($_REQUEST['Refresh'])
) {
unset($_REQUEST['Refresh_x']);
unset($_REQUEST['Refresh']);
print(RefreshForm($host, $form, $referer));
}
```

Weiterhin müssen beim Verarbeiten des Events die entsprechenden superglobalen Variablen zurückgesetzt werden! Eine Ausnahme bilden hiervon lediglich die Reload-Events zum neu einlesen der Autorenzugehörigkeit. Dieses Event wird im opusinterface.php in der Subroutine GetSubmitParameters() abgearbeitet und dort auch gelöscht.

Bei Fehlern in der Eingabe wird die Variable $okstr auf einen entsprechenden Ausdruck gesetzt, so daß diese Fehler in den Modulteilen submitform.php und submitform2.php am Anfang mit PrintError passend formatiert ausgegeben werden können.

Lookup/LookupContributor submitbackend.php überprüft zunächst, ob $_REQUEST['Lookup'] gesetzt ist. Aus dieser Variable muß die Nummer des nachzuschlagenden Autors extrahiert werden. Weiterhin ist zu unterscheiden ob ein Autor oder ein sonstiger Beteiligter nachgeschlagen werden soll. Die Events hierzu sind Lookup für Autoren und LookupContributor für sonstige Personen, wobei jeweils hinter dem Event selbst der Index zurückgeliefert wird, der zu extrahieren ist (z. B. Lookup5 für den 5. Autor).

Wird ein Lookup-Event erkannt, so wird zunächst $fillstep gelöscht und in den entsprechenden Lookup-Dialog (L-Author.php) verzweigt. $fillstep

muß zurückgesetzt werden, da so die Zuordnungsliste des Papiers zurückgesetzt wird, und die Fakultätsebene anhand der neuen Autorenliste so neu generiert wird. Die Funktion LookupAuthor, die in L-Author.php definiert ruft nach Beendigung automatisch wiederum submitform.php auf, wobei nun via $_REQUEST die neuen Autorendaten übergeben werden, und von GetSubmitParameters() eingelesen werden können.

AddAuthor/AddContributor, RemoveAuthor/RemoveContributor Liegt kein Lookup-Request vor wird geprüft, ob ein Autorenfeld addiert (AddAuthor oder AddContributor) oder entfernt (RemoveAuthor oder RemoveContributor) werden soll. Hierfür wird jeweils die Rückgabe der Funktionen AddFormElement() bzw. RmFormElement() via print ausgegeben. Die jeweiligen Routinen addieren oder entfernen einen Eintrag in der Autorenliste, in dem sie schlicht das Feld um einen Index erweitern oder kürzen.

Reload Dieses Event führt lediglich zum zurücksetzen von Fillstep. Es wird selbst nicht gelöscht! Auf diese Weise wird innerhalb von GetSumbitParameters() das automatische Auflösen der Namen nach internen/externen angestoßen.

Refresh Lädt lediglich das Formular neu. Dies führt z. B. auch zur Gültigkeitsüberprüfung der eingegebenen Werte. Dieses Event wird weiterhin vom "Zuordnungswizard" generiert, da der vorhandene HTML-Code entsprechende Checkboxen plaziert, die die nötigen Zordnungsfelder nach einem Refresh korrekt setzen.

Published Dieses Event wird ausgelöst, wenn der Autor die Felder für eine Zeitschriftenreferenz anfordert. Es wird lediglich $publishedpaper auf true gesetzt und wie üblich das Formular neu geladen.

Journal Ruft LookupJournal() auf um normierte Zeitschriftendaten zu übernehmen. Nach Auswahl der Zeitschrift kommt der Benutzer auf die zweite Seite des Submitdialogs zurück.

Series, ISBN, OtherURL, Validity, Access, Preprint, Class Diese Events sind analog zu Published, setzen die zugehörige Bool-Variable auf true und laden dann das Formaular neu, so daß die entsprechenden Felder in submitform2.php angezeigt werden.

SWD OPUS öffnet hier eine eigene Seite die dem Benutzer kurz erklärt, wie der normierte Schlagwortkatalog zu benutzen ist. In OPUS+ erzeugt submitbackend.php diese Seite und verzweigt dann auf den Schlagwortkatalog. Alle eingegebenen Parameter werden hierbei übergeben, so daß diese nach dem Rücksprung aus dem SWD wieder zur Verfügung stehen.

Page2 Ist die erste Seite ausgefüllt wird über dieses Event die zweite Seite angefordert. Hierbei werden alle bisherigen Eingaben auf Gültigkeit überprüft, ggf. `$okstr` gesetzt und die erste Seite wiederum mit einer Fehlermeldung angezeigt. Sind alle Werte gültig werden die externen Autoren extrahiert und auf die zweite Seite umgeschaltet.

SUBMIT Ist die zweite Seite komplett ausgefüllt erfolgt hier zunächst die Überprüfung auf Gültigkeit aller Parameter. Ggf. wird erzeut die zweite Seite zur Korrektur angezeigt. Sind alle Eingaben gültig werden sie dem Benutzer erneut in einer Tabelle zur Kontrolle mittels PrintOPUSPaperdata angezeigt. Dies entspricht der Funktion von meta.php aus OPUS. Das erzeugte Formular verzweigt dann zu ./uploadform.php, das den Fileupload übernimmt.

3.5 Exportschnittstellen

Ein zentrales Element für den Benutzer ist der Zugriff auf die Daten von OPUS+. OPUS selbst bietet hier nur den Zugriff auf den Volltext, es können jedoch keine bibliographischen Daten in eigene Bibliographieprogramme wie z. B. *BibTEX* oder *EndNote* übernommen werden. Diese Schnittstellen stehen in OPUS+ zur Verfügung. Sie sind in lib/exporters.php und dazugehörig in bibdownload.php realisiert.

3.5.1 Allgemeines

Jede Exportschnittstelle ist so realisiert, daß sie stets die Daten eines Papieres im entsprechenden Format zur Verfügung stellt. Die Funktion GetPaperData() ist hierbei die Schnittstelle zur Datenbank und füllt die üblichen OPUS-globalen Variablen mit den Werten eines über `$source_opus` spezifizierten Dokuments. Mit den Funktionen in lib/exporters.php werden diese dann entsprechend aufbreitet und entweder als String zurück gegeben (alle Funktionen die mit "As" anfangen, also z. B. AsBibTeXPaper()) oder driekt Ausgegeben (alle Funktionen ohne "As" am Anfang, also z. B. BibTeXPaper()).

Wichtig Werden die "As"-Funktionen verwendet ist darauf zu ach-

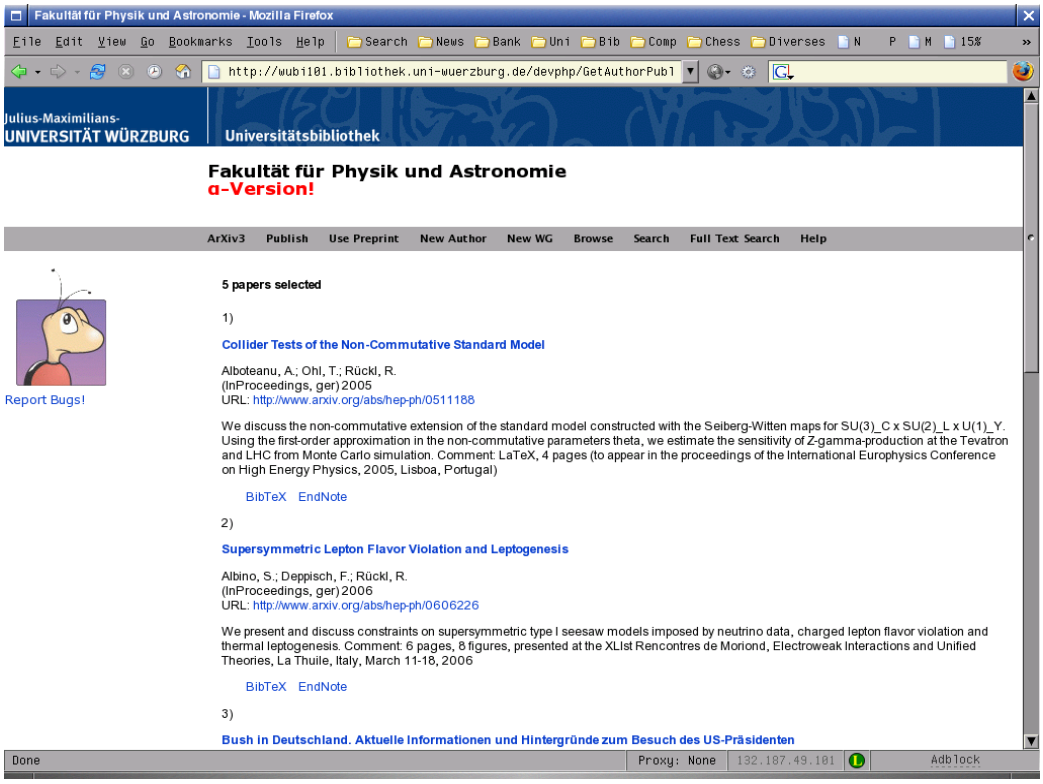

Abbildung 3.7: Publikationsliste als HTML: automatische Erzeugung über Parameter.

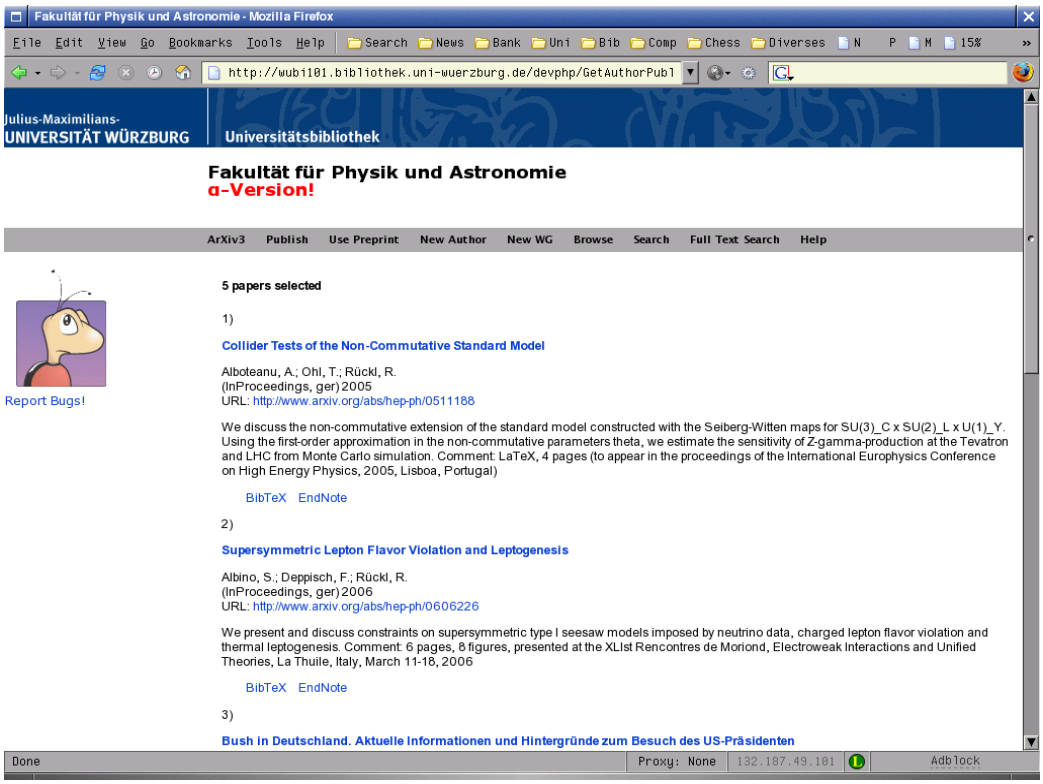

Abbildung 3.8: Publikationsliste als HTML: Interaktive Selektion

ten, daß zuvor die OPUS-üblichen globalen Variablen mit ResetSubmitVariables() gelöscht werden!

Manche Exportformate (z. B. BibTEX) benötigen für jede Referenz einen eindeutigen Identifier. Dieser wird folgendermaßen gebildet:

```
$projekt-$date_year:$source_opus
```

Dies führt zwar nicht zu schönen Identifiern aber zu definitiv eindeutigen. Sie können auf einfache Weise vom Benutzer selbst angepaßt werden.

3.5.2 Download

Um dem Benutzer die selektierten Listen oder Einträge zum Download anbieten zu können soll keine temporäre Datei erzeugt werden müssen. Aus diesem Grund werden die Liste zunächst in einer Variablen erzeugt. Dies geschieht in bibdownload.php welches seine Parameter aus dem Superglobalen Feld `$_REQUEST` ausliest. Hier muß einerseits `$_REQUEST['format']` gesetzt werden (mögliche Werte sind bibtex, arxiv oder endnote) sowie das angefragte

Format	Mime-Type
bibtex	text/x-bibtex
xml	application/xml
endnote	text/plain

Tabelle 3.3: Mime-Typen für den Bibliographiedownload

Dokument via `$_REQUEST['source_opus']`. Hierbei kann, muß aber nicht, `$source_opus` ein Feld sein. In diesem Fall wird die gesammte Liste exportiert, ansonsten die Daten für ein einzelnes Dokument. Da die Routine `$_REQUEST` ausliest kann man sie auch problemlos direkt als URL der Form

```
bibdownload?format=bibtex&source_opus=5
```

ansprechen. Dies mag für fortgeschrittene Benutzer von Interesse sein.

Nach dem erzeugen der Liste im enstsprechenden Format wird in der Routine Download() aus lib/network.php ein HTML-Header erzeugt der dem empfangenden Webbrowser einen Download vorspielt. Hierbei werden die in Tabelle 3.3 angegebenen Mime-Types benutzt, die es dem Benutzer erlauben automatisch ein passendes Anwendungsprogramm zu starten.

3.5.3 HTML

Dies ist im engeren Sinne kein Exportformat. Es dient vielmehr zur Darstellung von Literaturlisten innerhalb des OPUS-Systems selbst. Allerdings kann ein Autor über diese Funktion einen einfachen Link auf seine eigene Homepage setzen, der seine gesamten im `OPUS+` erfassten Publikationen stets aktuell auflistet.

Um die Daten eines eingestellten Dokuments in HTML auszugeben dient die Funktion HTMLPaper(). Hierzu gehört die Funktion AsHTML().

3.5.4 BibTEX

Das in den Naturwissenschaften weit verbreitete Satzsystem TEX/LATEXwird über dieses Exportformat bedient. Es werden hierbei keine bibitems erzeugt, sondern echte BibTEX-Records, die dann wie üblich mit Hilfe von bibtex in passende bibitems konvertiert werden können. Dies entspricht der in LATEXgebräuchlichen Form der Literaturverwaltung. Ein zu EndNote vergleichbares Verwaltungsprogramm hierzu wären z. B. `JabRef`oder `pybliographer`.

Um die Daten eines eingestellten Dokuments in BibTEXauszugeben dient die Funktion BibTeXPaper(). Hierzu gehört die Funktion AsBibTeX().

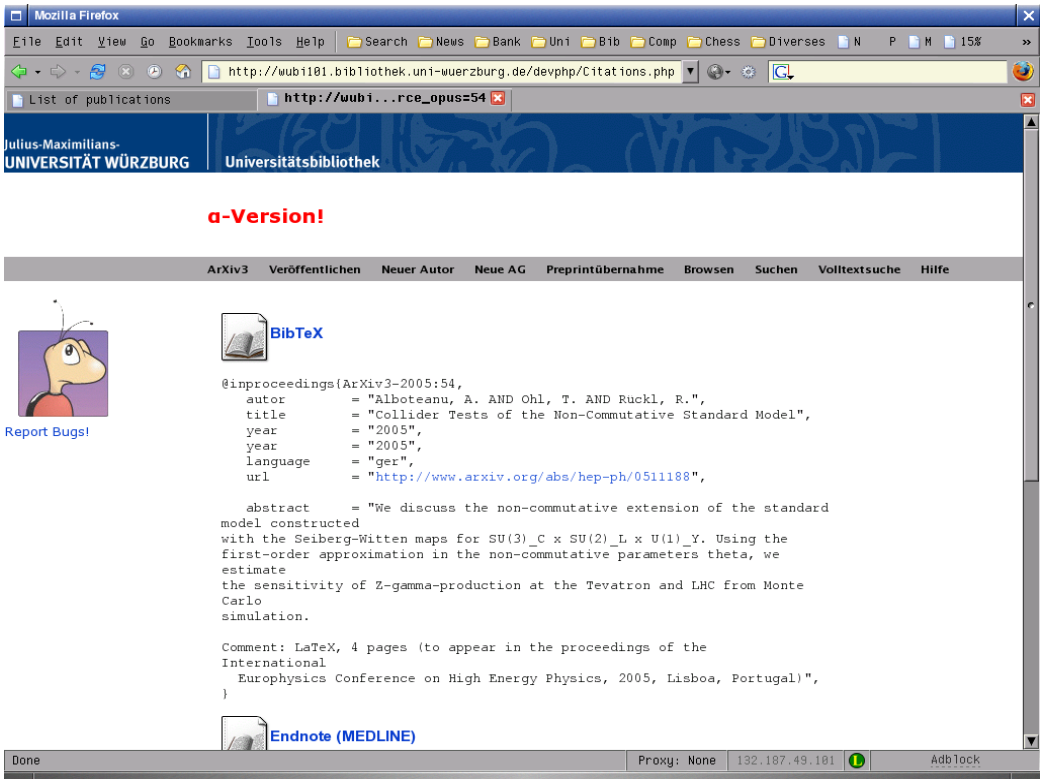

Abbildung 3.9: Bibliographiedownload

3.5.5 EndNote

Für den Export in EndNote wurde das Format von *NCBI PubMed* (Medline) realisiert, da EndNote hierfür im Lieferumfang bereits fertige Importfunktionen zur Verfügung stellt und es ein standardisiertes "EndNote-Format" nicht gibt.

Hinweis Das EndNote-Format kann auch von vielen andern Bibliographieprogrammen eingelesen werden.

Um die Daten eines eingestellten Dokuments Endnote-kompatibel auszugeben dient die Funktion EndnotePaper(). Hierzu gehört die Funktion AsMedline().

3.5.6 XML

Für Fortgeschrittene Benutzer steht dieser Export zur Verfügung. Das gewählte XML-Format ist im wesentlichen mit dem arXiv-Record identisch, allerdings werden die Journalreferenzen in Einzeltags ausgegeben, so daß ein leichteres Parsen der nämlichen möglich ist.

Um die Daten eines eingestellten Dokuments Endnote-kompatibel auszugeben dient die Funktion arXivPaper(). Hierzu gehört die Funktion AsArXivXML(), welche wiederum für die üblichen XML-Header die Funktionen XMLPageHeader() und XMLPageFooter() benutzt.

3.6 Kontakt zu den Autoren

Um mit den Autoren eines eingestellten Dokuments in Kontakt treten zu können sollen interne Autoren ihre e-Mail-Adresse in ihren Daten hinterlegen. Möchte ein Nutzer in Kontakt mit den Autoren treten, bekommt er einen entsprechenden e-Mail-Link angeboten, der ein einfaches Eingabeformular öffnet, in dem er seinen Namen und seine Adresse angeben kann, sowie die Nachricht and die Autoren. Diese Nachricht wird an alle Autoren, die eine Adresse hinterlegt haben versandt, wobei ihre mail-Adresse an den Benutzer nicht ausgegeben wird. Die Autoren erhalten neben dem Text des Benutzers auch den Abstract ihres Dokuments und einen Link zu diesem zugeschickt, so daß sie die Anfrage einfach zuordnen können. Die generierte Mail ist hierbei als reiner Text gehalten, so daß sie mit jedem Mailprogramm problemlos verarbeitet werden kann. Sie sieht beispielsweise so aus:

```
From    : "J. User" <j.user@somewhere.edu>
Subject : [ArXiv3]: Question about Your Publication
```

```
---------------- Sent via ArXiv3 Webpage ----------------

Nachfrage zum Dokument...

---------------- Sent via ArXiv3 Webpage ----------------



------------------------------------------------------------

This message refers to your publication in ArXiv3:
   (http://nbn-resolving.de/urn:nbn:de:bsz:nn-opus-543)
------------------------------------------------------------

Title   : <title>

Abstract:
<abstract>
------------------------------------------------------------
```

Bereits das Subject der Mail ist so gestaltet, daß ein einfaches Erkennen und auch ggf. Filtern möglich ist. Bei erfolgreichem Versandt wird diese Nachricht auch dem Benutzer exakt so angezeigt.

Zum Mailversandt wird das Modul phpmailer benutzt, das es erlaubt, einen externen *SMTP-Server* zu verwenden. Weiterhin sind in diesem Paket nahezu alle Möglichkeiten implementiert um e-Mails zu verschicken (z. B. Attachements, HTML etc.) so daß diese Lösung auch an anderer Stelle verwendet werden könnte.

Hinweis Da nicht festzustellen ist, welche Landessprache der jeweilige Autor vorzieht und ob alle Autoren die gleichen Sprachen sprechen werden alle Texte nur in englisch ausgebeben. Derzeit ist eine Sprachanpassung dieser Texte nicht vorgesehen, wäre aber natürilch einfach zu realisieren.

3.7 Neue Zeitschriften anmelden

Da keine einfache normierte Zeitschriftenliste zugänglich ist, werden Zeitschriften in einer internen Tabelle vorgehalten. Aus diesem Grunde ist natürlich keine Vollständigkeit dieser Liste gewährleistet. Publiziert ein Autor bei einer

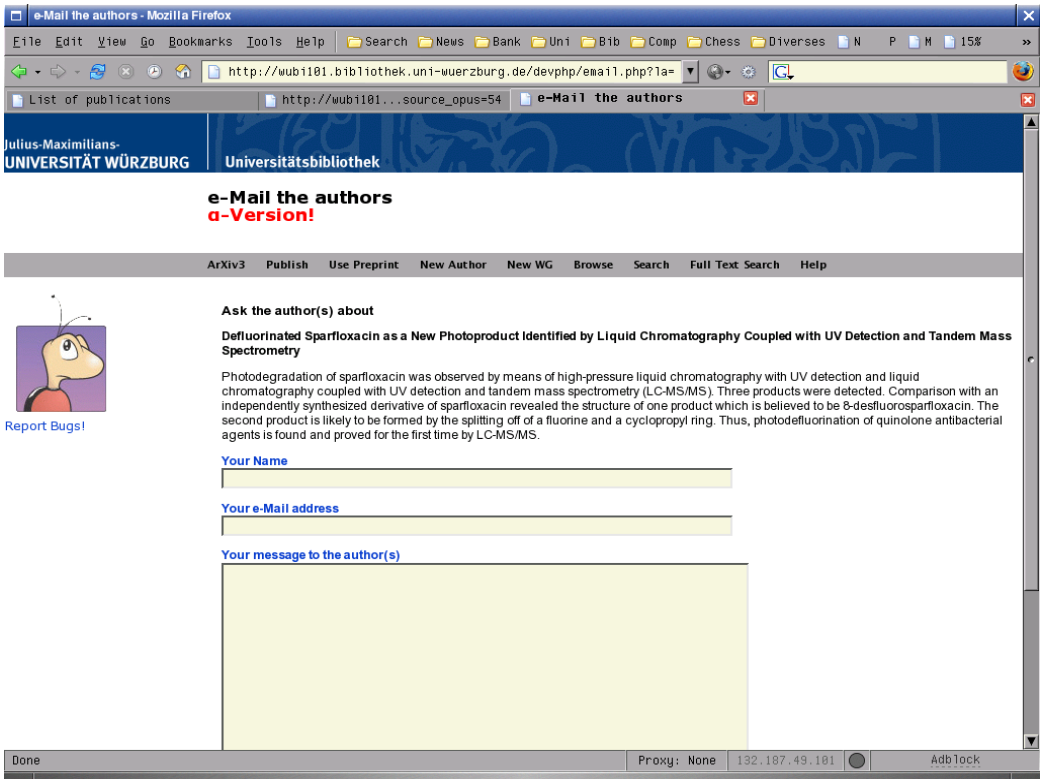

Abbildung 3.10: Kontakt zu den Autoren

Zeitschrift, die nicht vorhanden ist, os resultiert seine Suche in einer entweder leeren Liste oder einer Liste von Zeitschriften in der die gewünschte nicht gefunden werden kann.

In diesem Fall kann der Benutzer die Zeitschrift am System anmelden, ein entsprechender Button am Ende der Nachschlagen-Seite verzweigt in das Modul "NewJournal". In der folgenden Maske muß der Benutzer die entsprechenden Felder ausfüllen und die Zeitschirft wird sodann sofort in die Liste der vorhandenen Zeitschriften übernommen und steht somit den Benutzern sofort zur Verfügung.

Damit eine Kontrolle durch den Bibliothekar und ein entsprechendes Anpassen des Eintrags trotzdem möglich ist wird nach Anlage des neuen Datensatzes eine e-Mail an `$project@$projectdomain` geschickt, die die vom Benutzer eingegebenen Daten enthält.

3.8 Publikationslisten

Publikationslisten können auf zwei Arten erzeugt werden:

1. Interaktiv: GetGroupPublications.php
2. Automatisch: GetAuthorPublications.php mit entsprechenden Parametern.

Jede dieser Listen wird als HTML ausgegeben und im Browser dargestellt. Für jedes Dokument sowie für die gesamte Liste werden außerdem Downloadbuttons in den unterstützten Export-Formaten zur Verfügung gestellt.

3.8.1 Interaktive Benutzung

Diese ist hauptsächlich gedacht um per Mausklick Listen für verschiedene Organisationsbereiche zusammenzustellen. Derzeit können keine logischen Verknüpfungen realisiert werden, aber es ist auf einfache weise möglich von der Fakultätsliste zur Institutsliste, von dort zur Lehrstuhl- oder Arbeitsgruppenliste zu gelangen.

3.8.2 Automatische Erzeugung

Diese Funktion ist v.a. interessant wenn man für die eigenen Seiten automatisch eine Liste erzeugen möchte. Die Einbinung kann dabei über einen normalen Link erfolgen, der direkt auf die OPUS+-Datenbank zeigt. Es ist für

den Benutzer nicht notwendig irgendwelche Programmte zu schreiben oder direkt auf die Datenbank zuzugreifen, er erhält aber trotzdem eine stets akutelle Liste.

Zur Erzeugung der Liste stehen verschiedene Schlüsselwörter zur Verfügung:

- au: Selektiert per `$creator_name`
- id: Selektiert per numerischer Autorenkennung
- faculty: Selektiert eine Fakultät
- institute: Selektiert ein Institut
- chair: Selektiert einen Lehrstuhl
- group: Selektiert eine Arbeitsgruppe

Mit Hilfe dieser Schlüsselwörter wird dann eine URL der Form

```
.../GetAuthorPublications.php?la=de&<schl"usselwort>=<wert>
```

erzeugt. Der Aufruf dieser Adresse generiert dann aus der Datenbank eine Liste mit den entsprechenden Einträgen.

au Dies ist für den Benutzer am einfachsten, da er hier einfach seinen Namen angeben kann, so wie er auf dem Paper erscheint (creator name). Da dieser aber nicht eindeutig sein muß können durchaus auch fremde Dokumente in der Erzeugten Liste landen, da alle Autoren gültig sind, deren Name so "klingt wie". Intern wird hier ein SOUNDEX-Match eingesetzt, so daß man z.B: sowohl *Müller* als auch *Mueller* schreiben kann um das gleiche Ergebnis zu erziehlen.

Wichtig Gibt ein Autor als `$creator_name` *Müller, H.* an, so muß für den Wert von au auch *Müller,H.* angegeben werden! Die Leerzeichen sind nicht notwendig, jedoch das Intial, da sonst der Name "falsch klingt".

id Erzeugt eindeutige Listen für einen Autor, da hier die eindeutige numerische ID als Selektionskriterium benutzt wird. Diese ID wird dem Autor z. B. angezeigt, wenn er seine Daten editiert.

faculty Selektiert eine Faklutät anhand hierer Fakultätsnummer. Beispiel: faculty=11

institute Selektiert ein Institut anhand seiner Instituts-ID. Diese setzt sich aus der Faklultätsnummer und der Institutsnummer zusammen: Instituts-ID = ffii wobei ff die Fakultätsnummer und ii die Institutsnummer darstellt. Beispiel: faculty=1102

chair Selektiert einen Lehrstuhl anhand seiner ID. Diese setzt sich analog der Instituts-ID zusammen: ffiicc wobei cc die Lehrstuhlnummer darstellt. Beispiel: chair=110202

group Selektiert eine Arbeitsgruppe anhand ihrer ID. Diese kann man aus der Lehrstuhlnummer und der Arbeitsgruppennummer analog erzeugen: ffiiccgg Beispiel: chair=11020304

3.9 Jahresbibliographie

Eine immer wiederkehrende Anforderung ist eine Liste aller Publikationen innerhalb eines Jahres zur Erstellung der Jahresbibliographie der Universität. Eine einfache Ausgabe einer solchen Liste läßt sich in OPUS+ mit Hilfe von YearSummary.php realisieren. Um ein einfaches Einbinden in andere Webseiten per Link zu ermöglichen wird das gewünschte Jahr als Parameter übergeben. Zur Selektion der Papiere für die Bibliographie dient hierbei das Feld date_year in der Tabelle OPUS. Durch Verknüpufung mit der Tabelle DOCASSOC kann man nun auf einfache Weise eine Bibliographe erstellen. Der Aufruf der Bibliographie geschieht dabei

```
.../YearSummary..php?la=de&year=<jahr>
```

Es wird eine HTML-Liste aller Veröffentlichungen dieses Jahres geordnet nach Fakultäten erstellt. Sind mehrere Fakultäten beteiligt wird die Veröffentlichung bei jeder Fakultät aufgeführt, d. h. es entstehen hier (gewollt) Doubletten.

Teil II

Referenz

Kapitel 4

Tabellen für OPUS+

Neben den standardmäßigen OPUS Tabellen werden die hier beschriebenen Tabellen zusätzlich benötigt. Diese enthalten einerseits zusätzliche Dokumentinformationen und Zuordnungen, andererseits die detaillierteren Autorendaten für interne Verfasser. Zusätzlich müssen die weiteren Gliederungsebenen (Lehrstühle und Arbeitsgruppen) abgelegt werden.

4.1 Allgemeines

4.1.1 Landessprachunterstützung aka NLS aka i18n

Um mehrere Sprachen zu unterstützen führt OPUS mehrere Tabellen nahezu identischen Inhalts ein, die lediglich übersetzte Textfelder enthalten. Ein Beispiel ist die FACULTY-Tabelle, die als FACULTY_DE und FACULTY_EN vorliegt. Im OPUS-Code wird diese Tabelle sodann mit Hilfe von faculty_$la Anweisungen referenziert. Diese Aufteilung in mehrere Tabellen schafft eine gewisse Redundanz von Information neben einer Fehlerquelle beim Übersetzen, da der zugehörige Index in jeder Tabelle geführt werden muß. OPUS+ benutzt daher für alle Landessprachen nur eine Tabelle mit entsprechenden Spalten. Zwar werden die original-OPUS-Tabellen unverändert übernommen, aber z. B. die neue Tabelle CHAIRNAME enthält hier spalten wie NAME_DE, NAME_EN etc. Diese werden im PHP-Code analog referenziert, lediglich sind so alle Landessprachunterstützungen in einer Tabelle vereinigt.

Da im PHP-Code auf die jeweilige Sprachversion immer durch Stringverküpfung mit der Variablen $la zugegriffen wird sind beide Ansätze offen für eine beliebige Anzahl von Landessprachen.

4.1.2 Indicees

OPUS selbst verwendet in den internen Tabellen z. T. verschiedene Namen für die selben Indexwerte. Dieses wurde in OPUS+ bewußt verieden, da man so von der SQL-Anweisung USING profitieren kann. Auch wurden für Spaltennamen möglichst aussagekräftige Bezeichnungen verwendet, die einheitlichen Schemata folgen. Z. B. wird der (deutsche) Name einer Einrichtung immer NAME_DE genannt. Welche Name dies ist ergibt sich aus der Tabellenzugehörigkeit. CHAIRNAMES.NAME_DE sollte eindeutig genug sein.

4.2 Dokumentinformationen

Da zusätzlich zu den Dublin Core einträgen Metadaten benötigt werden müssen diese in weiteren Tabellen abgelegt werden. Dies geschieht so, dass diese Tabellen neben den "normalen" OPUS Tabellen existieren und diese nicht stören.

Lediglich zwei der Originaltabellen von OPUS werden ergänzt: OPUS_AUTOR und deren temporäres Äquivalent TEMP_AUTOR. Zu diesen Tabellen wird jeweils am Ende die Spalte ID hinzugefügt. Diese Spalte enthält die numerische Autoren-ID, sofern der $creator_name ein intern bekannter Autor ist. Sonst ist die Spalte $NULL.

4.2.1 docinfo / temp_docinfo

Primary Key	source_opus	
SOURCE_OPUS	int(11) NOT NULL	OPUS Dokument-ID
EXTAUTHORS	TEXT	Liste aller als exteren Autoren
JOURNALNO	varchar(25)	ID-Nummer des Journals
JOURNALYEAR	int	Jahr (erscheinen in Zeitschrift)
JOURNALVOL	int	Volume
JOURNALISSUE	varchar(25)	Issue
JOURNALPAGE	varchar(25)	Seiten (von-bis)
DOI	varchar(50)	Digital Object Identifier
MD5	varchar(40)	Checksumme
LOCALPREPRINT	varchar(25)	Lokale Preprintnummern
OTHERPREPRINT	TEXT	Andere Preprintnummern

Tabelle 4.1: docinfo / temp_docinfo

Diese Tabelle nimmt zusätzliche Metadaten einer Publikation auf, die in OPUS bisher nicht vorgesehen waren. Bei Zeitschriftenaufsätzen sind dies im wesenltichen die bibliographischen Daten der Zeitschrift in der Form, wie sie auch zitiert würden, bzw. der Wissenschaftler der die Primärdaten erfasst diese kennt. Wird eine DOI angegeben kann über diese ein Link zur Originalpublikation erfolgen.

Wichtig Verlangt ein Verlag in seinen Copyrightbestimmungen einen Link auf die publizierte Ausgabe, so muß eine DOI zur Verfügung stehen um diesen zu realisieren.

Die in einigen Communities nötigen Preprint-Nummern können ebenfalls abgelegt werden, wobei die lokale Preprintnummer getrennt gespeichert wird. Werden in der Autorenliste Autoren nicht erkannt so werden diese als Liste im Feld EXTAUTHORS abgelegt. Alle Felder dieser Liste sind fakultativ und nicht bei jeder Publikation nötig oder anwendbar.

Hinweis Die Tabelle temp_docinfo ist mit docinfo identisch. Beim Eintragen eines neuen Dokuments werden die zugehörigen Daten erst dort gespeichert und erst beim Übernehmen und freischalten des Dätensatzes auf docinfo übertragen. (Analog den anderen temp_-Tabellen von OPUS.)

4.2.2 docassoc / temp_docassoc

Primary Key	(source_opus, facultyNo, instituteNo, chairNo, wgNo)	
SOURCE_OPUS	int(11) NOT NULL	OPUS Dokument-ID
FACULTYNO	char(2) NOT NULL	Fakultätsnummer
INSTITUTENO	char(6) NOT NULL	Institutsnummer
CHAIRNO	int NOT NULL	Lehrstuhlnummer
WGNO	int NOT NULL	Arbeitsgruppennummer

Tabelle 4.2: Definition der Tabellen DOCASSOC und TEMP_DOCASSOC zur Dokumentzuordnung

Hier wird das Dokument einer oder mehreren Organisationseinheiten zugeordnet. Keiner der Felder darf ungesetzt sein, der kombinierte Primärschlüssel sorgt dafür, daß die Eintragsanweisungen auf die Tabelle keine Doubletten produzieren und möglichst einfach gehalten werden können, da der Primärschlüssel eventuelle Doubletten verwirft. (Der entsprechende Code in docassoc.inc führt zwangsweise zu solchen Doubletten und verläßt sich darauf, daß die Datenbank diese verwirft. Dies ermöglicht es, diesen Teil des Codes sehr einfach in

Schleifen zu realisieren.) Fakultäts-, Instituts-, Lehrstuhl- und Arbeitsgruppennummern entsprechen den jeweiligen Primärindizees der zugehörigen Tabellen, so dass aus diesen die Namen etc. eindeutig aufgelöst werden können.

Hinweis Die Tabelle temp_docassoc ist mit docassoc identisch. Beim Eintragen eines neuen Dokuments werden die zugehörigen Daten erst dort gespeichert und erst beim Übernehmen und freischalten des Dätensatzes auf docinfo übertragen. (Analog den anderen temp_-Tabellen von OPUS.)

4.3 Autorendaten

Um personalisierte Listen anbieten zu können ist es notwendig, dass die entsprechenden Autoren der Datenbank als solche bekannt sind. Aus diesem Grund werden Numerische Autoren-IDs eingeführt. OPUS kennt dies nicht.

In der Autorentabelle wird auch das Feld CREATOR_NAME geführt, wie es auch in OPUS sonst zur Identifikation verwendet wird. Es sollte die Schreibweise des Autorennamens enthalten, wie er auf den Papieren erscheint. (Viele Autoren ersetzen z.B. Umlaute im Namen durch die entsprechenden Umschreibungen ä=ae usw.)

Hinweis Ändert ein Autor später seine $creator_name (z. B. bei Namensänderungen des Autors) so wird diese Änderung nicht automatisch in den OPUS-Internen Tabellen nachvollzogen. Es wird lediglich in der Tabelle AUTHOR der alte Eintrag ersetzt. Über die gespeicherten numerischen IDs könnte man aber solch eine Namensänderung jederzeit auch in den anderen Tabellen nachvollziehen und es bleibt über diese ID auch die korrekte Verknüpfung mit dem selben Autor erhalten.

Wichtig Bei Namensänderung sollte sich ein Autor **nicht** neu anmelden sondern lediglich seine internen Daten anpassen. So bleiben auch weiterhin alle seine Dokumente ihm zugeordnet.

Im neuen Submitdialog submitform.php wird mit Hilfe der numerischen ID der zugenörige $creator_name ermittelt und in die Standard-OPUS-Felder übertragen. Dadruch wird sichergestellt, daß der Autorenname immer in konsistenter Schreibweise erscheint, und es wird möglich, den $creator_name (nicht eindeutig) der ID zuzuordnen und für den Benutzer sinnvolle Selektionslisten auszugeben (z. B. bei der Übernahme von Daten aus einem Preprintsystem wie arXiv oder PubMed).

4.3.1 author

Primary Key Index	ID creator_name, surename, forename	
ID	integer NOT NULL autoincrement	Eindeutige ID
CREATOR_NAME	varchar(100) NOT NULL	OPUS Autorenname
TITLENO	mediumint	Index für evlt. Titel
FORENAME	varchar(100) NOT NULL	Vorname
SURNAME	varchar(100) NOT NULL	Nachname
EMAIL	varchar(100) NOT NULL	e-Mail-Adresse

Tabelle 4.3: Definitionstabelle für interne Autoren – AUTHOR

Diese Tabelle ordnet der numerischen ID eindeutig OPUS CREATOR_NAME, echte Namen, Titel und e-Mail-Adressen zu. Die Mailadressen werden u. U. für die Kontaktaufnahme mit den Autoren verwendet (e-Mail Funktion). Sie ist nicht zu verwechseln mit der Standard OPUS-Tabelle OPUS_AUTOR, welche die Autoren eines bestimmten Papiers vorhält und deren Reihenfolge definiert.

Wichtig Jeder Autor soll sich nur einmal am System anmelden und ggf. Änderungen in Name, Titel oder e-Mail dann innerhalb seines Datensatzes nachvollziehen.

4.3.2 title

Primary Key	titleNo	
TITLENO	mediumint	Titelindex
TITLE_DE	varchar(25)	Titel auf Deutsch
TITLE_EN	varchar(25)	Titel auf Englisch

Tabelle 4.4: Definitionstabelle für akad. Titel – AUTHOR

Mit Hilfe dieser Tabelle können Titelindizees in die entsprechenden Landessprachlichen Titel übersetzt werden. Weitere Sprachen können durch hinzufügen entsprechender Spalten unterstützt werden.

4.3.3 membership

Primary Key	creator_name, facultyNo, instituteNo, chairNo, wgNo	
CREATOR_NAME	varchar(100) NOT NULL	OPUS Autorenname
FACULTYNO	char(2)	Fakultätsnummer
INSTITUTENO	char(6)	Institutsnummer
CHAIRNO	char(11)	Lehrstuhlnummer
WGNO	char(14) DEFAULT NULL	Arbeitsgruppennummer

Tabelle 4.5: Zuordnungsdefinition der Autoren – MEMBERSHIP

Hier werden alle Zugehörigkeiten eines Autors zu den verschiedenen Organisationseinheiten abgelegt. Jeder Autor kann in beliebig vielen Einheiten Mitglied sein, lediglich die Kombinationen müssen eindeutig sein. Der Primärschlüssel vermeidet Doubletten. Die Daten aus dieser Tabelle werden als Vorschlagswerte für die Zurodnung von Papieren zu den Organisationseinheiten benutzt.

Hinweis Diese Tabelle enthält allerdings **nicht** die Zuordnungen der Papiere sondern die Zuordnungen der Autoren! Die Zuordnung der Papiere zu den Organisationseinheiten wird in DOCASSOC (s.a. 4.2.2) festgehalten.

4.3.4 journals

Primary Key	ISSN	
ZDBID	varchar(50)	ID der ZDB
ISSN	varchar(25) NOT NULL	ISSN
NAME	varchar(250)	Name der Zeitschrift
SHORTCUT	varchar(25)	Int. Abkürzung
PUBLISHER	varchar(100)	Herausgeber
REVIEWED	bool	f. Peer Reviewed True
ISITITLE	varchar(250)	Titel bei ISI
ISISHORT	varchar(50)	Abkürzung bei ISI
ADDRESS	varchar(250)	Adresse des Verlags
COUNTRY	varchar(3)	Land des Verlags

Tabelle 4.6: Normierte Zeitschriftendaten – JOURNALS

Um Artikel eindeutig einer Zeitschrift zuordnen zu können sind normierte Zeitschriftennamen und ein passender Index nötig. Da es (derzeit) nicht möglich ist, auf eine endbenutzergängiges Subset der Zeitschriften Daten-Bank direkt zuzugreifen wird ein solches Subset in dieser Tabelle vorgehalten und manuell gepflegt. Der "ideale Index" wäre hierbei die eineindeutige ID, die von der ZDB vergeben wird. Da ein einpflegen dieser nicht ohne Aufwand ist wird derzeit allerdings die ISSN verwendet, die zumindest eindeutig sein sollte. Die ISI-Felder dienen nur der Statistik und weden dem Benutzer nicht angezeigt. Für Bibliographieexporte wird immer der volle Name der Zeitschrift NAME benutzt, das Feld SHORTCUT dient lediglich dazu, es für den Benutzer suchbar zu machen.

4.4 Gliederungsebenen

Die Abbildung der Gliederungsebenen erfolgt in Erweiterung der `OPUS`-internen Ebenen FACULTY_`$la` und INSTITUTE_`$la`. Hierbei werden die Standardtabellen als Integraler Bestandteil eingebunden und lediglich durch die neuen Tabellen erweitert. In den neuen Tabellen wird allerdings die Sprachunterstützung durch Spalten realisiert und nicht durch jeweils eigene Tabellen. Die Verknüpfung zu den Standard-`OPUS`-Tabellen erfolgt durch deren jeweilige Index-Felder, die genau so auch in den Erweiterungen benutzt werden.

Übergeordnete Einheiten wie Sonderforschungsbereiche, Graduiertenkollegs usw. werden auf die gleiche Struktur abgebildet. Z.B. bilden alle Sonderforschungsbereiche eine Fakultät, jeder SFB entspricht dann einem Institut, jedes Teilprojekt einem Lehrstuhl.

4.4.1 chair

Primary Key	chairNo, facultyNo	
CHAIRNO	char(11)	ID des Lehrstuhls
FACULTYNO	char(2) NOT NULL	Fakultät des Lehrstuhls
INSTITUTENO	char(6) NOT NULL	Institut des Lehrstuhls
LEHREINH	char(6)	Statistikschlüssel
ZPFGRU	char(6)	Statistikschlüssel
ZPFGEB	char(6)	Statistikschlüssel
SAPID	char(8)	ID im SAP

Tabelle 4.7: Lehrstuhldefinitionen und Statistikschlüssel – CHAIR

CHAIRNO ist der eigentliche Index eines Lehrstuhls. Die Felder FACULTYNO und INSTITUTENO nehmen die jeweiligen Primärindicees der Fakultät und des Instituts dem der Lehrstuhl angeschlossen ist. Die Felder LEHREINH, ZPFGRU und ZPFGEB sind Statistikschlüssel der Zentralverwaltung. Alle diese Schlüssel sind auf Lehrstuhlebene vergeben, so daß man sich ggf. von dort aus nach oben arbeiten muß. Insofern ist die Tabelle CHAIR die zentrale Tabelle zum Aufbau der Organisationseinheiten. Das Feld SAPID schließlich hält die im SAP-System der Universität vergebene ID vor.

Die CHAIRNO ergibt sich durch aneinanderhängen der Fakultäts- und Institusnummer, respektive gleichbedeutend durch die ersten Stellen der SAP-ID. Die Einzige Ausnahme bildet die *Medizinische Fakultät*, die beim Import in zwei Teile (Vorklinik und Klinik) geteilt wird. Sie enthält deshalb einen Buchstaben (a und b) im Institutsfeld. Um auch führende Nullen abbilden zu können werden alle ID-Nummern als char-Felder definiert.

4.4.2 chairname

Primary Key	chairNo	
CHAIRNO	char(11)	ID des Lehrstuhls
SUBJECT_DE	varchar(150) DEFAULT NULL	Arbeitsbereich (de)
SUBJECT_EN	varchar(150) DEFAULT NULL	Arbeitsbereich (en)
NAME_DE	varchar(150)	Name (de)
NAME_EN	varchar(150)	Name (en)

Tabelle 4.8: Lehrstuhlnamen und Arbeitsgebiete – CHAIR

Diese Tabelle löst die CHAIRNO eindeutig in den zugehörigen Namen und die Arbeitsgebiete auf. Die Namen der Lehrstühle werden aus der Zentralverwaltung übernommen.

4.4.3 workgroup

Primary Key	wgNo	
WGNO	char(14)	ID der Arbeitsgruppe
CHAIRNO	char(11)	ID des Lehrstuhls

Tabelle 4.9: Arbeitsgruppenzuordnung – WORKGROUP

Ordnet eine Arbeitsgruppe ihrem Lehrstuhl zu. Diese Tabelle ist zu Anfang leer und wird von den Benutzern entsprechend gefüllt. Die ID-Nummer wird fortlaufend vergeben, sie setzt sich aus Fakultäts-, Instituts- und Lehrstuhlnummer sowie der Gruppennummer fortlaufend zusammen.

4.4.4 workgroupname

Primary Key	wgNo	
WGNO	char(14)	ID der Arbeitsgruppe
LEADER	varchar(150) DEFAULT NULL	Arbeitsgruppenleiter
SUBJECT_DE	varchar(150) DEFAULT NULL	Arbeitsgebiet (de)
SUBJECT_EN	varchar(150) DEFAULT NULL	Arbeitsgebiet (en)
NAME_DE	varchar(150) NOT NULL	Name (de)
NAME_EN	varchar(150) NOT NULL	Name (en)

Tabelle 4.10: Arbeitsgruppdetails – WORKGROUPNAME

Hiermit werden die WGNO zu den entsprechenden Namen aufgelöst. Außerdem ist es möglich einen Arbeitsgruppenleiter zu definieren und ein Arbeitgebiet anzugeben. Diese Felder sollen von den Benutzern gefüllt werden.

Kapitel 5

OPUS+ Programmierbibliothek

5.1 Includefiles

Die Dateien in `include/` enthalten durchweg kurze Codestückchen die an passender Stelle direkt eingefügt werden können. I.d.R. enthalten sie keine Funktionen (wie die Bibliothek selbst in `lib/`) sondern immer wiederkehrende Deklarationen und Ausgabeanweisungen.

5.1.1 variables.inc

Alle globalen Variablen sind hier deklariert und mit dokumentierenden Werten vorbelegt. Am ende der Datei werden die Vorbelegungen durch die Settings in den Konfigurationsfiles überschrieben. In dieser Datei werden u. a. auch die Pfade auf die OPUS-Installation selbst gesetzt, da alle Erweiterungen außerhalb von OPUS leben. Diese Werte müssen angepaßt werden!

5.1.2 XPath.class.php

Eine Klasse zum Parsen von XML-Dokumenten. Da dies an verschiedensten Stellen benötigt wird, wird diese Datei von variables.inc includiert, so daß der XML-Parser immer zur Verfügung steht.

5.1.3 classification.js

Die interaktive Hilfe für die Klassifiktionsfunktion von OPUS. Dieser Javascript Code ist im Standard OPUS fest in `neu_allg.php` eingebaut, welches allerdings von OPUS+ ersetzt wird.

5.1.4 createpage.inc

Enthält alle Anweisungen zum Aufbau einer OPUS-Webpage, also zur Erzeugung des HTML-Headers und des Layouts.

5.1.5 docvariables.inc

Alle Variablen die ein Dokument betreffen sind hier als global Anweisung definiert. Includieren dieser Datei stellt sie also der aufrufenden Routine zur Verfügung.

5.1.6 footer.inc

Fügt einen Standard-OPUS-Footer unter der Seite ein, u.a. mit den Links auf andere Sprachversionen der Seite usw.

5.1.7 htmlheader.inc

Ein einfacher Standardkonformer HTML-Header.

5.1.8 htmlfooter.inc

Ein einfacher Standardkonformer HTML-Footer.

5.1.9 oaiservers.inc

Deklarationen zum Preprint-Interface, insbesondere die Datenstruktur, die die OAI-Interfaces aufnehmen. Weiterhin werden Funktionen deklariert, die ein OAI2-Ergebnis für OPUS aufbereiten.

QueryOAI2

Parameter:	
$url	URL des OAI-Interfaces
$useragent	Useragent-String
Return:	Rohes XML-Ergebnis der Abfrage

Tabelle 5.1: QueryOAI2()

Mit Hilfe der curl-Bibliothek wird eine Anfrage an den OAI2-Server unter `$url` erzeugt. Die URL sollte den gesamten Abfragestring in OAI2-konformer Syntax enthalten. Da einige Server einen gültigen Useragent benötigen um sich vor Robots zu schützen kann dieser ebenfalls angegeben werden.

OAIResult2HTML

Parameter:	
`$xmlresult`	OAI2-Ergebnis in XML
`$dublincore`	Dublin-Core Elemente
Return:	keine (direkte Ausgabe)

Tabelle 5.2: OAIResult2HTML()

Sucht die Elemente von `$dublincore` in `$xmlresult` und gibt sie als einfache HTML-Tabelle aus. Diese Funktion ist hauptsächlich für Debuggingzwecke gedacht. `$dublincore` muß nicht den Dublin-Core enthalten, sonern kann im Prinzip jedes Feld mit den Identifiern sein, die in `$xmlresult` gesucht werden sollen. http- und doi-Einträge werden als Links ausgegeben.

ParseOAIResult

Parameter:	
`$xmlresult`	OAI2-Ergebnis in XML
`$dublincore`	Dublin-Core Elemente
Global:	
`$dcresult`	Globales Array mit den Dublin-Core Elementen
Return:	keine (setzt globale Variablen)

Tabelle 5.3: ParseOAIResult()

Alle Elemente die in `$dublincore` vorkommen werden gesucht und in das globale Array `$dcresult` mit ihrem jeweiligen Namen als Index eingetragen. Diese Routine ist gedacht um den echten Dublin-Core zu parsen, ist aber generisch genug, daß auch andere Metadatensätze verarbeitet werden können sollten.

Wichtig Um mit Umlauten korrekt umgehen zu können wird iconv() benötigt. Intern wird immer mit ISO-8859-1 gearbeitet, es wird statisch auf

dieses Format hin konvertiert. Weiterhin wird davon ausgegangen, daß `$xmlresult` korrektes XML ist, also ein Content-Charset liefert.

5.1.10 opusinterface.inc

Hier werden zunächst globale Konfigurationsparameter eingelesen und die globale Variable `$opus` erzeugt, die mit der Datenbank verbindet und die Schnittstelle zum Abfragen der `OPUS`-Tabellen zur Verfügung stellt. Weiterhin werden Funktionen definiert die die üblichen globalen `OPUS`-Variablen einlesen oder zurücksetzen.

ReindexArray()

Parameter:	
&`$array`	Zu reindizierendes Feld
Return:	array (Variabelparameter)

Tabelle 5.4: ReindexArray()

Ordnet alle Arrayelemente wieder lückenlos an. Dies ist notwendig wenn z. B. ein Element gelöscht wird, da PHP intern alle Felder auf Hashes abbildet.

ReindexGlobalArrays()

Global:	
`$publisher_faculty`	Fakultäts-IDs (Feld)
`$publisher_inst`	Instituts-IDs (Feld)
`$publisher_chair`	Lehrstuhl-IDs (Feld)
`$publisher_group`	Arbeitsgruppen-IDs (Feld)
Return:	array (Variabelparameter)

Tabelle 5.5: ReindexGlobalArrays()

Reindiziert alle globalen Felder, wenn z.B. ein Feld erweitert oder gekürzt wurde. Es werden sowohl die Felder selbst als auch ihre Darstellung in `$_REQUEST` reindiziert.

ResetSubmitVariables()

Setzt alle Variablen des Submitdialogs auf ihre Startwerte zurück um sie in einen definierten Zustand zu bringen. Diese Funktion setzt auch `$publisher_university` auf einen Wert, sowie den Dokumenttyp auf 17 (Research Paper).

CheckSubmitParameters()

Gültigkeitsprüfung aller globalen Variablen des Submitdialogs. Diese Funktion ersetzt die Zahlreichen entsprechenden Blöcke in `OPUS`.

GetSubmitParameters()

Liest `$_REQUEST` aus und setzt alle globalen Parameter des Submitdialogs auf die zugehörigen Werte.

ReadSubmitFormTexts()

Liest die Texte des Submit-Formulars in die zugehörigen globalen Variablen. Da diese keine Felder sind werden Zahlreiche `$Text*` Werte eingelesen und passend gesetzt.

5.1.11 resulttable.inc

Eine Sammlung von Routinen zur Ausgabe von HTML-Tabellen. Die einfachste Form gibt nur das Ergebnis einer Abfrage in einer Tabelle aus, es können aber auch Tabellen von Checkboxen oder Radiobuttons erzeugt werden.

print_result_table()

Parameter:	
`$result`	Ergebnisstruktur von mysql_query()
`$width`	Breite der Tabelle, Standard: 100%
`$align`	Ausrichtung, Standard: “center”
`$border`	Breite des Randes, Standard: 0
`$cspace`	Zellenabstand, Standard: 1
`$cpad`	Zelleninnenabstand, Standard: 5
`$title`	Ausgabe des Titels, Standard: true
Return:	Keine (direkte Ausgabe)

Tabelle 5.6: print_result_table()

Diese Routine gibt den Inhalt von `$result` als Tabelle aus, wobei für die Ausrichtugnen, Breite und Abstände sinnvolle Vorgaben gemacht werden, so dass diese Routine alleine mit dem Ergebnis der Abfrage aufgerufen werden kann. Diese Routine findet praktisch in allen Tabellenformatierten Ausgaben Anwendung.

FormTable()

Parameter:	
`$type`	"checkbox" oder "radio"
`$result`	Ergebnisstruktur von mysql_query()
`$fieldname`	Feldname des Knopfes
`$passon`	Weiterleitungsparameter
`$no`	Index der Variablen die gesetzt werden soll
`$width`	Breite der Tabelle, Standard: 100%
`$align`	Ausrichtung, Standard: "center"
`$border`	Breite des Randes, Standard: 0
`$cspace`	Zellenabstand, Standard: 1
`$cpad`	Zelleninnenabstand, Standard: 5
Return:	Keine (direkte Ausgabe)

Tabelle 5.7: FormTable()

FormTable() erzeugt eine Tabelle von Check- oder Radio-Buttons, abhänig von `$type`. Übergeben wird das ergebnis einer SQL-Abfrage. Die Auswahl des Benutzers wird als `$_REQUEST[''fieldname'']` zurückgeliefert. Die Routine selbst schreibt aber keinen <form>-Header oder Footer, so dass sie in ein bestehendes Formular problemlos eingebettet werden kann. Normalerweise wird FormTable() nicht direkt aufgefufen sondern von den Aliasfunktoinen RadioFormTable() bzw. CheckFormTable(), die aber lediglich den Typ korrekt setzen.

Wichtig `$fieldname` muß auch im Hash `$passon` enthalten sein!

RadioFormTable()

Parameter:	
`$result`	Ergebnisstruktur von mysql_query()
`$fieldname`	Feldname des Knopfes
`$passon`	Weiterleitungsparameter
`$no`	Index der Variablen die gesetzt werden soll
`$width`	Breite der Tabelle, Standard: 100%
`$align`	Ausrichtung, Standard: "center"
`$border`	Breite des Randes, Standard: 0
`$cspace`	Zellenabstand, Standard: 1
`$cpad`	Zelleninnenabstand, Standard: 5
Return:	Keine (direkte Ausgabe)

Tabelle 5.8: RadioFormTable()

Entspricht FormTable() setzt allerdings den Typ automatisch auf "radio" und ruft diese dann auf. Dient im wesentlichen der besseren Lesbarkeit des Quelltextes.

CheckFormTable

Parameter:	
`$result`	Ergebnisstruktur von mysql_query()
`$fieldname`	Feldname des Knopfes
`$passon`	Weiterleitungsparameter
`$no`	Index der Variablen die gesetzt werden soll
`$width`	Breite der Tabelle, Standard: 100%
`$align`	Ausrichtung, Standard: "center"
`$border`	Breite des Randes, Standard: 0
`$cspace`	Zellenabstand, Standard: 1
`$cpad`	Zelleninnenabstand, Standard: 5
Return:	Keine (direkte Ausgabe)

Tabelle 5.9: CheckFormTable()

Entspricht FormTable() setzt allerdings den Typ automatisch auf "checkbox" und ruft diese dann auf. Dient im wesentlichen der besseren Lesbarkeit des Quelltextes.

print_result_radioform()

Parameter:	
`$result`	Ergebnisstruktur von mysql_query()
`$action`	Target-URL
`$fieldname`	Feldname des Knopfes
`$passon`	Weiterleitungsparameter
`$no`	Index der Variablen die gesetzt werden soll
`$width`	Breite der Tabelle, Standard: 100%
`$align`	Ausrichtung, Standard: "center"
`$border`	Breite des Randes, Standard: 0
`$cspace`	Zellenabstand, Standard: 1
`$cpad`	Zelleninnenabstand, Standard: 5
Global :	
`$useselectedtext`	Text für den Submit-Button
`$resetbuttontext`	Text für den Reset-Button
Return:	Keine (direkte Ausgabe)

Tabelle 5.10: print_result_radioform()

Erzeugt eine Radiobuttontable incl. dem Formular-Header, also ein komplettes, eigenständiges Formular. `$action` ist die URL an die die Auswahl via POST geschickt wird, wenn der Benutzer Submit klickt.

print_result_checkfrom()

Parameter:	
`$result`	Ergebnisstruktur von mysql_query()
`$action`	Target-URL
`$fieldname`	Feldname des Knopfes
`$passon`	Weiterleitungsparameter
`$no`	Index der Variablen die gesetzt werden soll
`$width`	Breite der Tabelle, Standard: 100%
`$align`	Ausrichtung, Standard: "center"
`$border`	Breite des Randes, Standard: 0
`$cspace`	Zellenabstand, Standard: 1
`$cpad`	Zelleninnenabstand, Standard: 5
Global :	
`$useselectedtext`	Text für den Submit-Button
`$resetbuttontext`	Text für den Reset-Button
Return:	Keine (direkte Ausgabe)

Tabelle 5.11: print_result_checkform()

Erzeugt eine Checkboxtable incl. dem Formular-Header, also ein komplettes, eigenständiges Formular. `$action` ist die URL an die die Auswahl via POST geschickt wird, wenn der Benutzer Submit klickt.

5.1.12 updatedocassoc.inc

5.2 Datenbankfunktionen

File `lib/dbfunctions.php`

Tabelle 5.12: dbfunctions.php

Dieses Modul faßt verschiedene Funktionen zum Zugriff auf die `OPUS` Datenbank und deren tabellen zusammen. Dies sind einerseits immer wiederkehrende SQL-Abfragen, andererseits auch das Prozessieren der Ergebnisse und laden der zahlreichen globalen `OPUS`-Variablen.

90	Sonderforschungsbereiche
91	Forschergruppen
92	Graduiertenkollegs
93	Forschungszentren
99	Extern

Tabelle 5.14: Besonderen Organisationseinheiten auf der Ebene einer Fakultät

5.2.1 GetFacultyforAuthor()

Tabellen :	
Lesend:	faculty_$la institute_$la chair membership author title
Schreibend:	keine
Parameter:	
`$creator_name`	Autorenkennung (Primärindex)
Global:	
`$la`	Sprache für die Ergebnisse
Return:	Fakultäts-IDs (Feld)

Tabelle 5.13: GetFacultyforAuthor()

Ermittelst die Fakultäten der ein Autor angehört anhand der Autorenkennung. Zurückgegeben wir die Indicees der entsprechenden Fakultät, nicht der Name.

Hinweis Besondere Einrichtungen (z. B. SFB) bilden alle zusammen eine Fakultät. D. h. die Menge aller Sonderforschungsbereiche bilden die Fakultät 90.

5.2.2 GetInstitueforAuthor()

Tabellen :	
Lesend:	faculty_$la institute_$la chair membership author title
Schreibend:	keine
Parameter:	
`$creator_name`	Autorenkennung (Primärindex)
Global:	
`$la`	Sprache für die Ergebnisse
Return:	Instituts-IDs (Feld)

Tabelle 5.15: GetInstituteforAuthor()

Analog zu GetFacultyforAuthor() werden die entsprechenden Institute geliefert, denen ein Autor angehört.

Der Institusindex ergibt sich hierbei aus dem zusammenhängen der Fakultät mit der fortlaufend numerierten zweistelligen Institutsnummer. D. h. der erste Sonderforschungsbreich bekäme die Instititusnummer 9001 aus 90 für SFB, und 01 für den ersten Eintrag.

Das Institut 9999 steht für Extern.

5.2.3 GetChairforAuthor

Tabellen :	
Lesend:	faculty_$la institute_$la chair membership author title
Schreibend:	keine
Parameter:	
`$creator_name`	Autorenkennung (Primärindex)
Global:	
`$la`	Sprache für die Ergebnisse
Return:	Lehrstuhl-IDs (Feld)

Tabelle 5.16: GetChairforAuthor()

Analog zu GetFacultyforAuthor() werden die Lehrstühle zurückgeliefert, dem ein Autor angeschlossen ist. Die Lehrstuhlnummer ergibt sich wie die Institutsnummer durch aneinanderhängen von Fakultäts- und Institutsnummer mit der fortlaufenden Lehrstuhlnummer (zweistellig).

5.2.4 GetAffiliations()

Tabellen :	
Lesend: Schreibend:	membership keine
Parameter:	

`$opus`
`$creator_name`
&`$publisher_faculty`
&`$publisher_inst`
&`$publisher_chair`
&`$publisher_group`
	OPUS-Objekt für DB-Zugriff
	Autorenkennung (Primärindex)
	Fakultäts-IDs (Feld)
	Instituts-IDs (Feld)
	Lehrstuhl-IDs (Feld)
	Arbeitsgruppen-IDs (Feld)

Global:	
`$la`	Sprache für die Ergebnisse
Return:	Variabelparameter!

Tabelle 5.17: GetAffiliations()

Liefert in den übergebenen Feldern $publisher_faculty, $publsiher_inst, $publisher_chair und $publisher_group die jeweils für den entsprechenden Autor gültigen Einträge zurück, so daß man eine komplette Liste aller Zuordnungen erhält. Alle Einträge der übergebenen Felder werden am ende eindeutig gemacht, so daß keine Doubletten zurückgegeben werden, auch wenn diese in den Abfragen zeitweise entstehen können.

5.2.5 CheckAuthor()

Tabellen : Lesend: Schreibend:	 author keine
Parameter:	
`$opus` `$authorID`	OPUS-Objekt für DB-Zugriff Autorenkennung (Primärindex)
Return:	1=true, 0=false

Tabelle 5.18: CheckAuthor()

Überprüft ob eine Autorenkennung bereits in der Datebank vorhanden ist. Diese Funktion wird z. B. bei der Neuanlage eines Autors benutzt, um die Eindeutigkeit der Autorenkennungen sicherzustellen.

Wichtig Es wird die Tabelle AUTHOR abgefragt, d. h. die Tabelle der als intern bekannten Autoren, nicht OPUS_AUTOR.

5.2.6 GetJournalData()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`include/docvariables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$journalid`	Journal-ID (Primärindex)
Return:	keine; lädt globale Variable

Tabelle 5.19: GetJournalData()

Alle Daten zur Zeitschrift `$journalid` werden in `$journaldata` geladen. Ist `$journalid` ungültig wird `Invalid` bzw. `N/A` zurückggegeben. Als `$journalid` dient derzeit die ISSN.

5.2.7 GetPaperData()

Tabellen :	
Lesend:	opus opus_autor docinfo journals docassoc
Schreibend:	keine
require:	`include/docvariables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$source_opus`	Paper-ID (Primärindex)
Return:	keine; lädt globale Variable

Tabelle 5.20: GetPaperData()

Diese Funktion liefert alle Daten zu der Veröffentlichung mit dem Indexeintrag $source_opus zurück, sofern dieser Eintrag vorhanden ist. Alle Werte werden in die in `OPUS` üblichen globalen Variablen geladen. Es werden außerdem die neu eingeführten globalen Variablen gesetzt, so daß alle globalen Variablen am Ende die Werte haben, wie sie nach dem Ausfüllen des Submitdialogs durch den Benutzer hätten.

5.2.8 ConstructAuthorName()

Tabellen :	
Lesend:	author
	title
Schreibend:	keine
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$creator_name`	Autoren-ID (Primärindex)
`$title`=false	Soll der Titel eingebaut werden?
Global:	
`$la`	Sprache für die Ergebnisse
Return:	Realname des Autors ggf. mit Titel

Tabelle 5.21: ConstructAuthorName()

Wandelt die Autoren-ID in den Realnamen des Autors ggf. incl. Titel für genau einen übergebenen Autor.

5.2.9 Creator2Author()

Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
&`$creator_name`	Autoren-ID (Primärindex, Feld)
`$title`=false	Soll der Titel eingebaut werden?
Global:	
`$la`	Sprache für die Ergebnisse
Return:	Variabelparameter!

Tabelle 5.22: Creator2Author()

Ruft für jeden Autor im übergebenen Feld ConstructAuthorName() auf und setzt den enstprechenden Eintrag in &$creator_name auf den erhaltenen Rückgabewert. Wird kein Feld, sondern ein Skalar übergeben wird dies hier richtig behandelt.

5.2.10 ReturnField()

Parameter:	
`$result`	SQL-Ergebnisstruktur
`$field`	Feldname
`$empty`	Rückgabewert wenn leer
Return:	Feldwert (kann Array sein)

Tabelle 5.23: ReturnField()

Gibt ein Feld einer SQL-Abfrage zurück in dem dieses Feld aus der Datenstruktur extrahiert wird. Im Fall eines leeren Resultats kann der dann zurückzugebende Wert angegeben werden.

5.3 HTML-Funktionen

File:	`lib/htmlfunctions.php`
requires:	`lib/network.php`

Tabelle 5.24: htmlfunctions.php

Enthält funktionen zum erzeugen immer wiederkehrender HTML-Elemente wie Eingabezeilen, Textbereiche, Buttons etc. sowie Hilfsfunktionen wie z. B. zum Ausgeben der `$GET`-/`$POST` Parameter, einladen externer HTML-Seiten, durchreichen aller `$REQUEST`-Parameter etc.

5.3.1 RecodeCharsToHTML()

Parameter:	
`$string`	String zum recodieren
Return:	Recodierter String

Tabelle 5.25: RecodeCharsToHTML()

Codiert spezielle Zeichen in ihre HTML-äquivalente. Derzeit nur < und >, kann aber jederzeit erweitert werden.

5.3.2 PrintReloadButton()

Parameter:	
`$graphic`=false	Graphischer Button?
Global:	
`$iconpath`	Pfad zu den Bitmaps
`$reloadpng`	Bitmap für Reloadbuttons
`$reloadtitle`	Text für Reloadbuttons
Return:	keine

Tabelle 5.26: PrintReloadButton()

Schreibt direkt den HTML-Code für einen Reload-Button, entweder als Textknopf oder als graphschen Knopf.

5.3.3 PrintContinueButton()

Parameter:	
`$graphic`=false	Graphischer Button?
Global:	
`$iconpath`	Pfad zu den Bitmaps
`$continuepng`	Bitmap für Continuebuttons
`$continuetitle`	Text für Continuebuttons
Return:	keine

Tabelle 5.27: PrintContinueButton()

Schreibt den HTML-Code für einen Continue-Button, graphisch oder als Text.

5.3.4 PrintError()

Parameter:	
`$str`	Fehlerstring
Global:	
`$erroricon`	HTML-Code für ein Fehlerbitmap
Return:	keine

Tabelle 5.28: PrintError()

Ausgabe des Fehlerstrings bei fehlenden Elementen in einer roten Tabelle.

5.3.5 DumpData()

Parameter:	
`$Data`	Hash mit den auszugebenden Daten
Return:	keine

Tabelle 5.29: DumpData()

Ausgabe des Hashes `$Data` in einer Tabelle incl. handling von Arrays innerhalb von Data. Als Beschriftung dienen die Hash-Keys.

5.3.6 DumpRequest()

Gibt alle Parameter aus `$_REQUEST` mit Hilfe der Funktion DumpData() aus.

5.3.7 PassData()

Parameter:	
`$Data`	Hash mit den Weiterzugebenden Daten
Return:	keine

Tabelle 5.30: PassData()

Schreibt input type="hidden" Formularzeilen für jeden Eintrag des übergebenen Hashes um diese bei auswahl des Submitbuttons an das nachfolgende Formular weiterzugeben.

5.3.8 PassRequest()

Benutzt die Funktion PassData() um alle Einträge von `$_REQUEST` weiterzuleiten.

5.3.9 SetFormHeadline()

Parameter:	
`$ico`	Icon (fertiges HTML-Element) vor der Überschrift
`$helpform`	URL des Hilfeformulars
`$la`	Sprache
`$tag`	Anchor innerhalb des Hilfeformulars
`$headline`	Überschrifttext
`$text`=""	Erklärungstext
`$ending`=""	Angehängter Text
Return:	keine

Tabelle 5.31: SetFormHeadline()

Gibt eine OPUS-üblich formatierte Überschirft aus. `$ending` ist dazu gedacht ggf. Links oder Buttons hinter der Überschrift ausgeben zu können. Die anderen Elemente entsprechen dem Standard-OPUS-Style.

5.3.10 SetTextArea()

Parameter:	
$ico	Icon (fertiges HTML-Element) vor der Überschrift
$tag	Anchor innerhalb des Hilfeformulars
$helptitle	Titel des Hilfelinks
$text	Überschrifttext
$textpreset	Voreinstellung des Areawertes
$areaname	Name des Formularelements
$cols=80	Breite
$rows=2	Zeilen
Return:	keine

Tabelle 5.32: SetTextArea()

Erzeugt eine Textarea im OPUS-Layout mit link auf das Standardhilfeformular $doku_pfad/hilfe_formular.php, wobei ein Text voreingestellt werden kann und die größe des Feldes frei definierbar ist.

5.3.11 SetTextField()

Parameter:	
$ico	Icon (fertiges HTML-Element) vor der Überschrift
$tag	Anchor innerhalb des Hilfeformulars
$helptitle	Titel des Hilfelinks
$text	Überschrifttext
$fieldvalue	Voreinstellung des Areawertes
$fieldname	Name des Formularelements
$endtext=""	Text am Ende der Eingabezeile
$width=80	Breite
$maxlength=100	Maximale Länge
$paragraph=true	Feld in einem Paragraph?
Return:	keine

Tabelle 5.33: SetTextField()

Erzeugt eine Texteingabezeile im Standard-OPUS-Layout mit Link auf das Standardhilfeformular $doku_pfad/hilfe_formular.php, wobei ein Text vor-

eingestellt werden kann. Der Feldwert kann voreingestellt werden, es können Breite und Maximallänge definiert werden, wobei die Defaults den üblichen OPUS-Voreinstellungen entsprechen. Wird $paragraph=true gesetzt wird das Eingabefeld in einen Absatz gesetzt. Endetext kann zusätzliche HTML-markups aufnehmen um z. B. am Ende Des Feldes einen Button o. ä zu plazieren.

5.3.12 SetTextInfo()

Parameter:	
$ico	Icon (fertiges HTML-Element) vor der Überschrift
$tag	Anchor innerhalb des Hilfeformulars
$helptitle	Titel des Hilfelinks
$text	Überschrifttext
$fieldvalue	Voreinstellung des Areawertes
$fieldname	Name des Formularelements
$endtext=""	Text am Ende der Eingabezeile
$width=80	Breite
$maxlength=100	Maximale Länge
$paragraph=true	Feld in einem Paragraph?
Return:	keine

Tabelle 5.34: SetTextInfo()

Analog zu SetTextField() aber der eingegetragene Text ist nicht veränderbar.

5.3.13 GetCheckboxString()

Parameter:	
$name	Name des Formelements
$value	Value des Formelements
$text	Überschrifttext
$checked=false	Voreinstellung des Areawertes
Return:	HTML-Markup für eine Checkbox

Tabelle 5.35: GetCheckboxString()

Liefert das HTML-Markup für eine Checkbox zurück, die entweder gesetzt oder nicht gesetzt ist, abhängig von `$checked`.

Hinweis Das Element wird nicht ausgegeben, sondern das Markup als String zurückgeliefert, so daß dieses in andere Elemente eingebaut werden kann.

5.3.14 GetButtonString()

Parameter:	
`$ico`	Icon des Buttons
`$alt`	alt-Tag für graphische Buttons
`$name`	Name des Buttons
`$buttonvalue`	Value des Buttons
`$graphic`=true	Graphischer Button?
Return:	HTML-Markup für einen Button

Tabelle 5.36: GetButtonString()

Erzeugt das für einen Button nötige HTML, wobei der Button entweder ein Textknopf oder ein graphischer Knopf sein kann, je nach Wert von `$graphic`. Graphische Buttons sind clickbare Bilder.

Hinweis Das Element wird nicht ausgegeben, sondern das Markup als String zurückgeliefert, so daß dieses in andere Elemente eingebaut werden kann.

5.3.15 GetClickButtonString()

Parameter:	
`$ico`	Icon des Buttons
`$alt`	alt-Tag für graphische Buttons
`$name`	Name des Buttons
`$buttonvalue`	Value des Buttons
`$graphic`=true	Graphischer Button?
Return:	HTML-Markup für einen Button

Tabelle 5.37: GetClickButtonString()

Erzeugt das für einen graphischen Button nötige HTML, wobei der auch die Auszeichnung eines Textknopfes hat, also als echter Knopf erscheint.

Hinweis Das Element wird nicht ausgegeben, sondern das Markup als String zurückgeliefert, so daß dieses in andere Elemente eingebaut werden kann.

5.3.16 AddFormElement()

Parameter:	
`$elementname`	Name des Elements
`$host`	Hostname/IP des Forumlars
`$form`	URL des Formulars ohne Host
`$referer`	Referer
`$elementval=""`	Wert des Elements
Return:	Formular mit zusätzlichem Element

Tabelle 5.38: AddFormElement()

Fügt ein Element in ein Formular ein, indem der zugehörige `$_REQUEST`-Eintrag um ein Element mit dem Wert von `$elemval` erweitert wird. Danach wird ein POST-Request an den Webserver `$host` geschickt, http-Header entfernt und das resultierende HTML zurückgeliefert. Die Prozedur setzt voraus, daß das angeforderte Formular seine Übergabeparameter entweder mit foreach() oder einer äquivalenten Schleife abarbeitet, so daß die Anzahl der Elemente eines Eintrags nicht statisch ist.

5.3.17 RmFormElement()

Parameter:	
`$elementname`	Name des Elements
`$host`	Hostname/IP des Forumlars
`$form`	URL des Formulars ohne Host
`$referer`	Referer
`$elementno="__LAST__"`	Index des zu entfernenden Elements
Return:	Formular mit einem Element weniger

Tabelle 5.39: RmFormElement()

Das Gegenstück zu AddFormElement(). Es gelten die gleichen Bedingungen wie dort. `$elementno` ist der Index des zu entfernenden Elements (Standardmäßig das letzte).

Hinweis Soll nicht das letzte Element entfernt werden, so muß das Aufgerufene Formular foreach()-Schleifen verwenden!

5.3.18 RefreshForm()

Parameter:	
`$host`	Hostname/IP des Forumlars
`$form`	URL des Formulars ohne Host
`$referer`	Referer
Return:	HTML-Seite mit dem Formular

Tabelle 5.40: RefreshForm()

Sendet einen http-POST-Request an den `$host` und fordert `$form` an. Hierbei werden alle `$_REQUEST` parameter übergeben. Der http-Header wird entfernt und der resultierende HTML-Code zurückgegeben.

5.3.19 SelectFromQuery()

Parameter:	
`$num`	Anzahl der Einträge in `$res`
`$res`	mysql-query Ergebnis
`$name`	Name des Formularfeldes
`$selected`	Dieser Eintrag wird Vorselektiert
`$valfield`=0	Index von `$res`, der den Wert enthält
`$textfield`=1	Index von `$res` mit den Ausgabetexten
`$size`=1	Zeilenzahl der Selectbox
Return:	HTML-Markup für eine Selectbox

Tabelle 5.41: SelectFromQuery()

Erzeugt aus dem Ergebnis einer SQL-Abfrage eine Selectbox. Wird dieser String zu groß schein PHP bisweilen ein Problem mit Speicherüberläufen zu bekommen.

5.3.20 OpenWindow()

Fügt den JavaScript-Code ein um ein neues Fenster zu öffnen.

5.3.21 FetchHTTP()

Parameter:	
`$url`	URL
`$useragent`	Useragent
Return:	HTML-Page

Tabelle 5.42: FetchHTTP()

Holt `$url` via http ab, wobei vorgegeben wird `$useragent` zu sein. Viele Seiten brauchen einen gültigen Useragent.

5.3.22 GetSherpaData()

Parameter:	
`$journaledata`	`$journaldata` (Feld)
Return:	Hash mit den SHERPA/RoMEO Daten des Journals

Tabelle 5.43: GetSherpaData()

Fragt mit dem internen Datenblock `$journaldata` SHERPA/RoMEO ab um die Rechte eines Autors für Publikationen in der angegebenen Zeitschrift zu erfragen. Die resultierende XML-Seite wird geparsed und in einem passenden Hash zurückgegeben.

5.3.23 PrintSherpaInfo()

Parameter:	
`$sherpadata`	Ergebnis von GetSherpaData()
Return:	keine

Tabelle 5.44: PrintSherpaInfo()

Gibt eine Tabelle mit den SHERPA/RoMEO-Daten aus, die in `$sherpadata` übergeben werden.

5.4 network

File	`lib/network.php`

Tabelle 5.45: network.php

In dieser Datei sind Funktionen zum Zugriff auf das Netzwerk zusammengefaßt.

5.4.1 Download()

Parameter:	
`$data`	Zu schickende Daten
`$type`	Content-type der Daten
`$name`	Name der Zieldatei
Return:	keine

Tabelle 5.46: Download()

Diese Funkition initiiert im Webbrowser einen Download, wobei keine temporären Dateien erzeugt werden. Dem Webbrowser wird durch schreiben des entsprechenden HTML-Headers lediglich vorgespielt, daß eine Datei gesendet würde. `$type` sollte auf den korrekten Content-Type gesetzt werden, damit der Webbrowser in die passende externe Anwendung verzweigen kann. Die Größe des gesendeten Datenblocks ermittelt die Funktion aus der größe von `$data`

5.4.2 WriteHiddenForm()

Global:	
$_REQUEST	Alle der Seite übergebenen Parameter
Return:	keine

Tabelle 5.47: WriteHiddenForm()

Für jede in $_REQUEST enthaltende Variable werden enstprechende <input type="hidden"> zeilen erzeugt, so daß alle Parameter weitergegeben werden können.

5.4.3 ConstructPostString()

Global:	
$_REQUEST	Alle der Seite übergebenen Parameter
Return:	Poststring

Tabelle 5.48: ConstructPostString()

Aus dem Inhalt von $_REQUEST wird ein entsprechender POST-string konstruiert um alle Parameter an ein Formular weiterzuleiten. Diese Funktion konsturiert z. B. den passenden $data-String für PostToHost().

5.4.4 PostToHost()

Parameter:	
$host	Host des Formulars
$path	URL des formulars ohne Host
$referer	Referer-URL
$data	Datenstring
Return:	Ergebnisformular

Tabelle 5.49: PostToHost()

Öffnet einen Socket auf Port 80 des Zieles `$host` und generiert eine POST Anfrage. `$data` enthält alle Parameter die die Formular an der Zieladresse übergeben werden sollen. Zurückggeben wird die Antwort von `$host` also im Erflogsfall das (vor-)ausgefüllte Formular.

Diese Funktion scheint abhängig von der PHP-Version nicht immer zu funktionieren, der Originalcode ist deswegen kommentiert und durch einen entsprechenden Aufruf von HTTP_Post() ersetzt.

5.4.5 HTTP_Post()

Parameter:	
`$host`	Host des Formulars
`$path`	URL des formulars ohne Host
`$referer`	Referer-URL
`$data`	Datenstring
Return:	Ergebnisformular

Tabelle 5.50: HTTP_Post()

Diese Funktion ist äquivalent zu Abschnitt 5.4.4. Es wird allerdings curl benutzt statt dem händischen öffnen des Ports.

5.5 exporters

File	`lib/exporters.php`

Tabelle 5.51: exporters.php

Dieses Modul enthält alle Exportroutinen in die verschiedenen unterstützten Formate. Jede Exportroutine bearbeitet einen einzelnen Dokumenteintrag in der Datenbank. Die Routinen deren Name mit As anfängt liefern einen entsprechend formatierten String zurück und können nach Aufruf von ResetSubmitParameter() auch alleine benutzt werden um entsprechende Strings zu erzeugen.

5.5.1 AsBibTeX()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`lib/dbfunctions.inc`
require:	`include/docvariables.inc`
require:	`include/variables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$source_opus`	Dokument-ID (Primärindex)
Return:	Formatierter String

Tabelle 5.52: AsBibTeX()

Benutzt GetPaperData() um alle gespeicherten Informationen zu `$source_opus` abzufragen und erzeugt daraus einen BibTeX-String mit den verwendbaren Metainformationen. Zusätzlich zu den Standard-BibTeX-Feldern werden weitere Felder für *DOI*, *URL* und *abstract* erzeugt. Alle anderen Daten werden in die üblichen BibTeX-Felder eingetragen, der Dokumenttyp wird anhand des `OPUS`-Dokumenttyps ermittelt. Zusätzliche Felder werden so abgelegt, daß sie problemlos mit `JabRef`weiterverarbeitet werden können und entsprechende Links zu den Volltexten vorhanden sind.

5.5.2 AsMedline()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`lib/dbfunctions.inc`
require:	`include/docvariables.inc`
require:	`include/variables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$source_opus`	Dokument-ID (Primärindex)
Return:	Formatierter String

Tabelle 5.53: AsMedline()

Benutzt GetPaperData() um alle gespeicherten Informationen zu `$source_opus` abzufragen und erzeugt daraus einen Medline-String mit den verwendbaren Metainformationen. Hierbei wird das Format von *pubmed* benutzt, so daß *Endnote* unter Verwendung des mitgelieferten pubmed-Filters die Daten direkt importieren kann.

5.5.3 AsHTML()

Tabellen : Lesend: Schreibend:	 journals keine

require:
require:
require:
	`lib/dbfunctions.inc`
	`include/docvariables.inc`
	`include/variables.inc`

Parameter:	
`$opus` `$source_opus`	OPUS-Objekt für DB-Zugriff Dokument-ID (Primärindex)
Return:	Formatierter String

Tabelle 5.54: AsHTML()

Benutzt GetPaperData() um alle gespeicherten Informationen zu `$source_opus` abzufragen und erzeugt daraus einen html-String mit den verwendbaren Metainformationen. Ist ein DOI vorhanden wird die Quellangabe der Zeitschrift automatisch zum Link auf den Volltext via DOI, wobei der Resolver der deutschen Bibliothek (nbn-resolving.de) zum Auflösen des DOI benutzt wird. Das Ausgabeformat ist ähnlich dem von arXiv.org

5.5.4 BibTeXPaper()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`lib/dbfunctions.inc`
require:	`include/docvariables.inc`
require:	`include/variables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$source_opus`	Dokument-ID (Primärindex)
Return:	Keine (Direkte Ausgabe)

Tabelle 5.55: BibTeXPaper()

Setzt globale Variablen zurück, gibt das Ergebnis von AsBibTeX() direkt aus.

5.5.5 EndnotePaper()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`lib/dbfunctions.inc`
require:	`include/docvariables.inc`
require:	`include/variables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$source_opus`	Dokument-ID (Primärindex)
Return:	Keine (Direkte Ausgabe)

Tabelle 5.56: EndnotePaper()

Setzt globale Variablen zurück, gibt das Ergebnis von AsMedline() direkt aus.

5.5.6 HTMLPaper()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`lib/dbfunctions.inc`
require:	`include/docvariables.inc`
require:	`include/variables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$source_opus`	Dokument-ID (Primärindex)
Return:	Keine (Direkte Ausgabe)

Tabelle 5.57: HTMLPaper()

Setzt globale Variablen zurück, gibt das Ergebnis von AsHTML() direkt aus.

5.5.7 DumpPaper()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`lib/dbfunctions.inc`
require:	`include/docvariables.inc`
require:	`include/variables.inc`
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
`$source_opus`	Dokument-ID (Primärindex)
Return:	Keine (Direkte Ausgabe)

Tabelle 5.58: DumpPaper()

Setzt globale Variablen zurück, schreibt eine HTML-Tabelle mit allen Daten (unsortiert) die in der Datenbank zu einem Paper vorhanden sind. Hauptsächlich für Debugging.

5.6 opufields

File:	lib/opusfields.php
require:	lib/network.inc
require:	lib/htmlfunctions.inc

Tabelle 5.59: opusfields.php

Verschiedene Funktionen um übliche OPUS-Felder auszugeben. Einige dieser Funktionen werden aktiv nicht benutzt, da sich z. B. Drop-Down-Listen mit allen Instituten usw. als unzweckmäßig erwiesen haben.

5.6.1 FacultyField()

Tabellen :	
Lesend:	faculty_$la
Schreibend:	keine
Parameter:	
$ico	Icon am Anfang des Headers
$publisher_faculty	Fakultätsfeld
$fakultaet	Text der Überschrift
$helpform	URL des Hilfeformulars
$la	Sprache (wie global $la)
$iconpath	Pfad zu den Icons
$addpng	Bitmap für "hinzufügen"
Global:	
$opus	OPUS-Objekt
$spacericon	HTML-String für ein leeres Icon
Return:	$publisher_faculty

Tabelle 5.60: FacultyFields()

Funktion für "original OPUS": gibt eine Selectionbox mit allen Fakultäten direkt aus.

5.6.2 InstituteField()

Tabellen :	
Lesend:	institute_$la
Schreibend:	keine
Parameter:	
$ico	Icon am Anfang des Headers
$publisher_inst	Institutsfeld
$fakultaet	Text der Überschrift
$helpform	URL des Hilfeformulars
$la	Sprache (wie global $la)
$iconpath	Pfad zu den Icons
$addpng	Bitmap für "hinzufügen"
Global:	
$opus	OPUS-Objekt
$spacericon	HTML-String für ein leeres Icon
Return:	$publisher_faculty

Tabelle 5.61: InstituteField()

Analog zu FacultyField(): Eine Dropdown-Liste aller Institute.

5.6.3 PrintSelectedTable()

Parameter:	
$res	SQL-Result
Return:	Keine (Direkte Ausgabe)

Tabelle 5.62: PrintSelectedTable()

Ruft print_result_table() auf, wobei die Ergebnistabelle 80% der "Seitenbreite" einnimmt, um diese optisch abzuheben. Helperfunction für Selected*Table()-Routinen.

5.6.4 SelectedFacultiesTable()

Parameter:	
`$res` Global:	SQL-Result
`$publisher_faculty`	Fakultätsliste
Return:	Keine (Direkte Ausgabe)

Tabelle 5.63: SelectFacultiesTable()

Schreibt alle Einträge der Fakultätsliste als Hidden-Form-Elemente und gibt die Liste selbst mit PrintSelectedTable() für den Benutzer aus.

5.6.5 SelectedInstitutesTable()

Parameter:	
`$res` Global:	SQL-Result
`$publisher_inst`	Institutsliste
Return:	Keine (Direkte Ausgabe)

Tabelle 5.64: SelectedInstitutesTable()

Schreibt alle Einträge der Institutsliste als Hidden-Form-Elemente und gibt die Liste selbst mit PrintSelectedTable() für den Benutzer aus.

5.6.6 SelectedChairsTable()

Parameter:	
`$res` Global:	SQL-Result
`$publisher_chair`	Lehrstuhlliste
Return:	Keine (Direkte Ausgabe)

Tabelle 5.65: SelectedChairsTable()

Schreibt alle Einträge der Lehrstuhlliste als Hidden-Form-Elemente und gibt die Liste selbst mit PrintSelectedTable() für den Benutzer aus.

5.6.7 SelectedGroupsTable()

Parameter:	
`$res` Global:	SQL-Result
`$publisher_group`	Arbeitsgruppenliste
Return:	Keine (Direkte Ausgabe)

Tabelle 5.66: SelectedGroupsTable()

Schreibt alle Einträge der Arbeitsgruppenliste als Hidden-Form-Elemente und gibt die Liste selbst mit PrintSelectedTable() für den Benutzer aus.

5.6.8 GetLineColour()

Parameter:	
`$number`	Integerzahl
Return:	HTML-string “class=”

Tabelle 5.67: GetLineColour()

Um in Ausgabetabellen jede zweite Zeile farbig zu unterlegen wird ermittelt ob `$number` gerade oder ungerade ist. Abhänig davon wird der String class="oddline" bzw. class="evenline" zurückgeliefert, so daß dieser in ein HTML-Element eingebaut werden kann. Das Stylesheet sollte diese beiden Klassen entsprechend deklarieren, um die gewünschte Farbgebung zu erreichen.

5.6.9 PrintOPUSPaperdata()

Tabellen :	
Lesend:	dokumentart language_`$la` bereich_`$la` faculty_`$la` institute_`$la` chairname workgroupname sachgruppe_ddc klassifikation_`$la` schriftenreihen
Schreibend:	keine
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
Global:	
`$*`	alle für eine Veröffentlichung relevanten
Return:	Keine (Direkte Ausgabe)

Tabelle 5.68: PrintOPUSPaperdata()

Unter Verwendung aller für die eingegebene Veröffentlichung relevanten globalen OPUS-Variablen wird eine Tabelle aufgebaut und dem Benutzer ausgegeben, damit dieser vor dem Endgültigen Fileupload seine Eingaben überprüfen kann.

5.6.10 ChairTableFromArray()

Tabellen :	
Lesend:	chairname
Schreibend:	keine
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
Global:	
`$checkedico`	Icon für "geprüft"
`$helpform`	Hilfeformular
`$la`	Sprache
`$chairstring`	Titeltext
`$publisher_chair`	Lehrstuhlliste
Return:	Keine (Direkte Ausgabe)

Tabelle 5.69: ChairTableFromArray()

Aus der globalen Variablen `$publisher_chair` wird mit Hlfe von SelectedChairsTable() eine Tabelle der ausgewählten Lehrstühle erstellt.

5.6.11 GroupTableFromArray()

Tabellen :	
Lesend:	workgroupname
Schreibend:	keine
Parameter:	
`$opus`	OPUS-Objekt für DB-Zugriff
Global:	
`$checkedico`	Icon für "geprüft"
`$helpform`	Hilfeformular
`$la`	Sprache
`$groupstring`	Titeltext
`$publisher_group`	Arbeitsgruppenliste
Return:	Keine (Direkte Ausgabe)

Tabelle 5.70: GroupTableFromArray()

Aus der globalen Variablen `$publisher_group` wird mit Hlfe von SelectedGroupTable() eine Tabelle der ausgewählten Arbeitsgruppen erstellt.

5.6.12 NewGroupLink()

Global:	
`$iconpath`	Pfad zu den Icons
`$workgrouppng`	Icon für "Arbeitsgruppe addieren"
`$groupaddalt`	Alt-Tag für das Icon
`$la`	Sprache
`$reloadpng`	Icon für "Neu laden"
`$reloadformtitle`	Text für "Neu laden"
`$graphbutton`	Graphische Buttons?
`$NewWorkgrouptitle`	Titelstring
Return:	Keine (Direkte Ausgabe)

Tabelle 5.71: NewGroupLink()

Setzt einen Button zum Anlegen einer neuen Arbeitsgruppe. Dieser Button öffnet ein neues Fenster mit dem enstprechenden Formular.

5.6.13 SelectGroupWhizard()

Tabellen :	
Lesend:	journals
Schreibend:	keine
require:	`lib/dbfunctions.inc`
require:	`include/docvariables.inc`
require:	`include/variables.inc`
Parameter:	
`$publisher_faculty`	Fakultätsliste
`$publisher_inst`	Institutsliste
`$publisher_chair`	Lehrstuhlliste
`$publisher_group`	Arbeitsgruppenliste
`$opus`	OPUS-Objekt für Datenbankzugriff
`$la`	Sprache
`$creator_name`	Autoren-ID
`$forename`	Vorname
`$surname`	Nachname
`$requiredico`	Icon für "Pflichtfeld"
`$checkedico`	Icon für "geprüft
`$helpform`	Hilfeformular
`$fakultaet`	Fakultätstitel
`$institut`	Institustitel
`$chairstring`	Lehrstuhltitel
`$groupstring`	Arbeitsgruppentitel
`$uniquestring`	String für "muß eindeutig sein"
`$iconpath`	Pfad zu den Icons
`$finalpng`	Bitmap für "Alles Fertig, Übernehmen"
`$graphbutton`	Graphische Buttons?
`$enterbuttontext`	Text für Fertigstellen-Knopf
Return:	Keine (Direkte Ausgabe)

Tabelle 5.72: SelectGroupWhizard()

Baut einen Wizard auf, in dem sich der Benutzer der Reihe nach durchclicken kann um Fakultät, Institut, Lehrstuhl und Arbeitsgruppe auszuwählen. Als Basis für die möglichen Auswahlfelder dienen die die Zugehörigkeiten aller in `$creator_name` eingetragenenen Autoren jeweils durch or verknüpft. Einträge in der nächst höheren Ebene werden ignoriert. D. h. gehört ein Autor z. B.

zur Fakultät 10 und 11 wählt aber im Wizard nur 11, so kann er bei den Instituten trotzdem noch ein Institut wählen, daß der Fakultät 10 angehört, da ein Papier ja von einer Fakultät und einem Institut erstellt werden kann, die nicht notwendig zusammengehören. Beispiel: Der Dekan der Fakultät 11 erstellt eine Festschrift mit einem Mitglied eines Instituts der Fakultät 10 und einem Mitarbeiter eines SFB-Teilprojekts.

5.6.14 EditableAssociationList()

Tabellen :	
Lesend:	faculty_$la intitute_$la chair workgroup
Schreibend:	keine
Global:	
$publisher_faculty	Fakultätsliste
$publisher_inst	Institutsliste
$publisher_chair	Lehrstuhlliste
$publisher_group	Arbeitsgruppenliste
$opus	OPUS-Objekt für Datenbankzugriff
$la	Sprache
$creator_name	Autoren-ID
$forename	Vorname
$surname	Nachname
$requiredico	Icon für "Pflichtfeld"
$checkedico	Icon für "geprüft
$helpform	Hilfeformular
$fakultaet	Fakultätstitel
$institut	Institustitel
$chairstring	Lehrstuhltitel
$groupstring	Arbeitsgruppentitel
$uniquestring	String für "muß eindeutig sein"
$iconpath	Pfad zu den Icons
$finalpng	Bitmap für "Alles Fertig, Übernehmen"
$graphbutton	Graphische Buttons?
$enterbuttontext	Text für Fertigstellen-Knopf
Return:	Keine (Direkte Ausgabe)

Tabelle 5.73: EditableAssociationList()

Erzeugt eine "editierbar" Zuordnungsliste. D. h. für die Werte in $publisher_faculty, $publisher_inst, $publisher_chair und $publisher_group werden die entsprechenden Texte aus der Datenbank abgefragt und eine Zuorndungsliste als Tabelle dargestellt. Hinter jeder Organisationseinheit wird ein Button plaziert, der ein entsprechendes Event auslöst, mit dem dieses Feld editiert

werden kann. Editierbar sind hierbei die bereiche für die ein “EEvent”: ist in $_REQUEST z. B. 'EFaculty' gesetzt ist die Fakultät editierbar. Analog für 'EInst', 'EChair' und 'EGroup'. Die entsprechenden Editierbuttons erzeugen genau diese Events, das Backendscript muß diese entsprechend behandeln.

5.6.15 CreatorInputFields()

require:	include/variables.inc
Global:	
$opus	OPUS-Objekt
Parameter:	
$creator	Feld mit den Namenseinträgen
$creafieldname	Feldname für Namenseinträge
$numid	Feld mit numerischen IDs
$numidfieldname	Feldname für numerische IDs
$LookupEvent	Name des “Nachschlagen”-Events
$AddEvent	Name des “Feld Addieren”-Events
$RemoveEvent	Name des “Feld Entfernen”-Events
$verfasser	Titeltext der Eingabefelder
$text10	Zusätzlicher Hinweistext
Return:	Keine (Direkte Ausgabe)

Tabelle 5.74: CreatorInputFields()

Erzeugt aus den übergebenen Parametern eine Tabelle von Eingabefeldern für Autoren oder sonstige Mitwirkende. Weiterhin werden Buttons erzeugt um zusätzliche Autorenfelder anzufordern oder das letzte zu entfernen. Außerdem wird ein Button eingefügt, der ein “Reload”-Event erzeugt, das in GetSumbmitParameters() ausgewertet wird und versucht anhand der eingegebenen $creator_names die zugehörigen numerischen IDs zu ermitteln. Die entsprechenden Feldnamen werden in dieser Form in den erzeugten HTML-Code geschrieben.

5.6.16 AuthorInputFields()

;

require:	include/variables.inc
Global:	
$creator_name	Namensliste Autoren
$numauthorid	Numerische IDs
$Verfasser	Titeltext (OPUS)
$text10	Hinweistext (OPUS)

Tabelle 5.75: AuthorInputFields()

Ruft CreatorInputFields() auf um Autorenfelder zu erzeugen.

5.6.17 ContributorInputFields()

require:	include/variables.inc
Global:	
$contributors	Namensliste weitere Personen
$numcontribid	Numerische IDs
$sonstige_personen	Titeltext
$text14	Hinweistext 1 (OPUS)
$text15	Hinweistext 2 (OPUS)

Tabelle 5.76: ContributorInputFields()

Ruft CreatorInputFields() auf um Felder für sonstige beteiligte Personen (Herausgeber etc.) zu erzeugen.

5.7 Lookup-Module

5.7.1 LookupAuthor()

File	Lookups/L-Author.php

Tabelle 5.77: L-Author.php

Tabellen :	
Lesend:	membership
	faculty_$la
	institute_$la
	chairname
	author
	title
Schreibend:	keine
require:	include/opusinterface.inc
require:	include/variables.inc
require:	include/createpage.inc
require:	include/resulttable.inc
Parameter:	
$expression	Suchbegriff (regexp!)
$no	Index des Autors der nachgeschalgen wird
$passon	Variablen die weitergereicht werden sollen ($_REQUEST)
Global:	
$useselectedtext	Text für "Weiter" Button
$resetbuttontext	Text für Reset-Button
Return:	Keine (Direkte Ausgabe)

Tabelle 5.78: LookupAuthor()

Übernimmt den Suchbegriff $expression und schlägt diesen bei den Autorenkennungen nach. Hierbei sind regular expressions erlaubt. Die globale Variable $creator_name wird an Position $no auf den aus einer Radiobuttonliste ausgewählten Wert gesetzt.

5.7.2 LookupJournal()

File	Lookups/L-Journal.php

Tabelle 5.79: L-Journal.php

Tabellen :	
Lesend:	membership
	faculty_$la
	institute_$la
	chairname
	author
	title
Schreibend:	keine
require:	include/opusinterface.inc
require:	include/variables.inc
require:	include/createpage.inc
require:	include/resulttable.inc
Parameter:	
$expression	Suchbegriff (regexp!)
$passon	Variablen die weitergereicht werden sollen ($_REQUEST)
Global:	
$useselectedtext	Text für "Weiter" Button
$resetbuttontext	Text für Reset-Button
$opus	OPUS-Objekt für DB-Zugriff
$la	Sprache
$newjournaltext	Text für den "Neu anlegen" Knopf
$unknownjournal	Text für "unbekannte Zeitschrift"
$journallookuptitle	Titel der Seite
Return:	Keine (Direkte Ausgabe)

Tabelle 5.80: LookupJournal()

Fragt die lokale Datenbank nach der Eingegebenen Zeitschrift ab. Dabei werden nacheinander Name, Abkürzung und Indexnummer abgefragt bis ein Treffer erziehlt wird. Wir nichts gefunden erscheint eine entsprechende Meldung, bei mehreren Treffern eine Liste mit Radioknöpfen aus denen der Benutzer auswählen kann um den korrekten Eintrag zu übernehmen. Dieses Nachschlagen stellt sicher, dass normierte Namen verwendet werden und eine eindeutige Zuordnung vorhanden ist.

Literaturverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Index