Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Beat Krähenmann, Martin Waldburger

Toolkit für Konferenzprogrammverwaltung und -personalisierung

Diplomarbeit DA-2003.03
Wintersemester 2002/2003

Betreuer: Dr. Erik Wilde

Verantwortlicher:
Prof. Dr. Bernhard Plattner

28.2.2003

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abstract

Unsere Untersuchungen von bestehenden Webseiten zu wissenschaftlichen Konferenzen haben gezeigt, dass diese Webseiten fast ausschliesslich statisch generiert sind. So könnensich Konferenzteilnehmer kein eigenes Programm zusammenstellen, ebenfalls dürften für den Webseiten-Administrator Änderungen am Konferenzprogramm mehr als mühsam zu bewerkstelligen sein.

In dieser Arbeit wurden die wichtigsten Eigenschaften von bestehenden Konferenz-Webseiten bestimmt. Diesen bildeten die Grundlage für unser Conference Time-Table Management (CTTM), welches zusätzlich noch ein personalisiertes Programm erlauben soll.

Das CTTM ist als dynamischen Weblösung in Cocoon mit Java Servlets und XML/XSLT implementiert. Ebenfalls sollte das CTTM möglichst allgemein sein: Einerseits muss es für möglichst viele Konferenzen benutzbar sein, andererseits muss es auch auf verschiedenen Plattformen installierbar sein. Dies wird durch die plattformunabhängigen Servlets bewerkstelligt. Die Informationsspeicherung geschieht in einer beliebigen JDBC-kompatiblen SQL-Datenbank. Auf der Benutzerseite ist eine Ausgabe auf verschiedene Medien und in verschiedenen Formaten möglich, wie zum Beispiel HTML, WML, PDF, RTF und XLS (MS Excel). In dieser Version beschränkt sich die Ausgabe vorerst auf HTML, eine Anpassung ist jedoch einfach möglich. Das Webdesign selber ist grösstenteils von den Konferenzdaten entkoppelt. Der Benutzer kann einen Account beantragen, seine Daten ändern und einen persönlichen Stundenplan zusammenstellen. Ferner hat ein Administrator die Möglichkeit, Benutzer-Einstellungen oder den Konferenzplan zu manipulieren. Bei Konferenzplanänderungen wird, falls erwünscht, der Benutzer via Email oder SMS benachrichtigt.

Kapitel 1

Evaluation

1.1 Bestehende Websites

In einer ersten Phase wurde anhand diverser Webauftritte von Konferenzen evaluiert, welche Möglichkeiten diese in ihrem Online-Programm anbieten. Die Websites wurden dabei hinsichtlich folgender Gesichtspunkte untersucht:

- Wie werden die Events dargestellt, auf welche Weise können sie betrachet werden.
- Inhaltsangabe eines Events, wie ist ein Event aufgebaut (Verweis auf Referent, Zeit, Datum, Raum, Abstract,...).
- Statisches HTML (das heisst, die Events sind in statische Webseiten codiert worden, konzeptionelle Änderungen würden einen grossen Aufwand bedeuten).
- Dynamisches HTML (das heisst, die Event-Werte werden aus irgendeinem Backend wie zum Beispiel eine Datenbank generiert).
- Besondere Eigenheiten.

Tabelle 1.1 gibt eine Übersicht über die evaluierten Websites. Dabei wurde festgesellt, dass ein Grossteil der untersuchten Websites völlig statisch aufgebaut ist. Features wie ein personalisiertes Benutzerprogramm oder flexible Darstellungsarten der Events sind kaum implementiert.

Konferenz-Website	Darstellung der Events	Inhaltsangabe	Bemerkungen
2. Oekonux-Konferenz (http://www.oekonux-konferenz.de)	- alle Events - nach Referenten - nach Abstracts - nach Tagen	- Referenten (Email, Lebenslauf, Links, Vorträge) - Titel - Art des Anlasses - Abstract - Zeit und Datum - Dauer - Track	- statisch
XML Conference (http://www.xmlconference.org/xmlusa/)	- nach Zeit - nach Themengebiet - alle Tracks parallel	- Referenten (Organisation, Lebenslauf) - Track - Abstract - Zeit - Fragen via Email an den Referenten	- statisch
WWW2002 (http://www2002.org/)	- nach Track - Wochenübersicht	- sehr wenig Infos über Referenten - dafür viel über den Aufenthalt selber	- statisch
Internetcity (http://www.internetcity.org)	- nach Zeit - nach Track - nach id	- Abstract - Referent (Vortrag, Lebenslauf) - id	- statisch
ACM Multimedia 2002 (http://mm02.eurecom.fr)	- nach Zeit	- Autoren	- statisch
Siggraph 2002 (http://www.siggraph.org/s2002)	- nach Kursarten - nach Zeit	- Ort - Voraussetzungen/Schwierigkeit - Abstract - Schedule eines Kurses	- statisch
Sigcomm 2002 (http://www.acm.org/sigcomm/sigcomm2002/)	- nach Zeit	- Authoren - Abstract - Paper - PDF-Version des Konferenzprogramms statisch	- statisch
Pittcon 2002 (http://www.pittcon.org/)	- nach Zeit - nach Autor - nach Keywords	- Autor - Abstract - als pdf-file	- persönliches Programm! - dynamisch
SVG Open (http://www.svgopen.org/index-2002.shtml)	- nach Zeit - nach Autor - nach Name - nach Thema - nach Titel - nach Land	- Abstract - Paper	- zum teil dynamisch
ISC-Symposium (http://www.isc-symposium.org/isc)	- Konferenzprogramm	- keine - Lebenslauf der Dozenten in pdf	- zwei Sprachen - statisch
ISMB 2002(Biologie) (http://www.isc-symposium.org/isc)	- Konferenzprogramm - Tutorials	- Autor - Abstract - Kontakt - Weblink	- statisch

Tabelle 1.1: Katalogisierung der evaluierten Konferenz Websites

1.1.1 Anforderungen an das CTTM

Damit unser Konferenzsystem auch wirklich gebraucht werden kann, sollte es folgende Eigenschaften haben:

Flexibilität: Das CTTM sollte genügend flexibel sein, um für eine Vielzahl von bestehenden Konferenzen-Websites einsetzbar zu sein.

Integrierbar: Das CTTM muss in beliebige Seiten integrierbar sein. Die vorhandenen Design-Fragmente sollte auch problemlos an andere Designs anpassbar sein.

Erweiterbar: Das ihm Rahmen dieser Diplomarbeit entstandene CTTM sollte mit einem vernünftigen Einarbeitungsaufwand um weitere Funktionen erweiterbar sein.

Medienunabhängig: Die Ausgabe sollte nicht nur in HTML möglich sein, sondern auch in anderen Formaten (PDF, WML, SVG).

Benachrichtigungen: Benachrichtigungs-Meldungen sollten auf die gebräuchlichsten Medien weitergeleitet werden können (Email, SMS, Fax,...)

1.1.2 Spezifische Anforderungen an das CTTM

In einem weiteren Schritt wurde bestimmt, welche Features unser CTTM haben soll. Wir haben dabei zwischen "Need to Haves" und "Nice To Haves" unterschieden.

1.1.2.1 Need to Have

1. Events
   - Titel
   - Inhalt
   - Datum
   - Zeit, Dauer
   - Ort, Raum
   - Referenten
   - Links
2. User- und Administrator-Authentisierung
3. Erstellen eines individuellen Konferenzplans
   - Überprüfung auf Überschneidungen
   - SMS/Email-Notifizierung bei Änderungen
   - SMS/Email-Notifizierung kurz vor Event-Beginn
4. Darstellung der Events
   - Nach Konferenz-Hierarchie
   - Nach Referenten
   - Nach Tagen (Stundenplan)
   - Nach Tracks (alle Parallel, aber auch nur einzelne)

1.1.2.2 Nice to Have

1. Administration
   - Webbasiertes Erstellen des Konferenzprogramms
   - Änderungen des Programms
2. WML, PDF-Versionen von diversen Ansichten
3. Keyword-Search
4. Infos zum Referenten
   - Angebotene Vorträge
   - Lebenslauf
   - Weitere Publikationen
5. Mehrere Userlevels
   - Administrator
   - Referent
   - Konferenzteilnehmer
   - Verantwortlicher für einen Event
6. Mehrere Sprachen
7. Uploads von weiteren Dokumenten zu einem Event

Welche Ziele erreicht werden konnten und welche nicht wird im Abschnitt 6.1 erläutert.

1.2 Wahl der Technologie

Ziel unserer Arbeit ist, ein Tool zu kreieren, das für verschiedenen Konferenzen gebraucht werden kann. Inzwischen existiert eine grosse Anzahl von Technologien, die speziell auf Webapplikationen zugeschnitten sind, oder die Tools anbieten, welche ständig wiederkehrende Aufgaben lösen oder mindestens erleichtern. Für das CTTM wurden diverse Technologien evaluiert, die bei der Implementation von Nutzen sein könnten. Die folgenden Kapitel geben einen Überblick über diese Technologien.

1.2.1 Content Management System (CMS)

Die Idee hinter einem CMS ist die sinnvolle Verwaltung von Inhalten. Die betrachteten CMS implementieren zudem eine Benutzerauthentisierung und die Vergabe von Zugriffsrechten. Dies sind zwei Funktionen, welche auch für das CTTM benötigt werden. Im weiteren implementieren die CMS Funktionen, die eine Trennung von Darstellung, Struktur und Inhalt ermöglichen. In reinen HTML-Webseiten sind diese beiden Punkte immer miteinander verknüpft. Musste eine Änderung am Layout vorgenommen werden, so bedeutet dies, dass sämtliche Webseiten angepasst werden mussten. Dies ist eine Arbeit, die je nach Grösse der Homepage sehr zeitaufwendig werden kann.

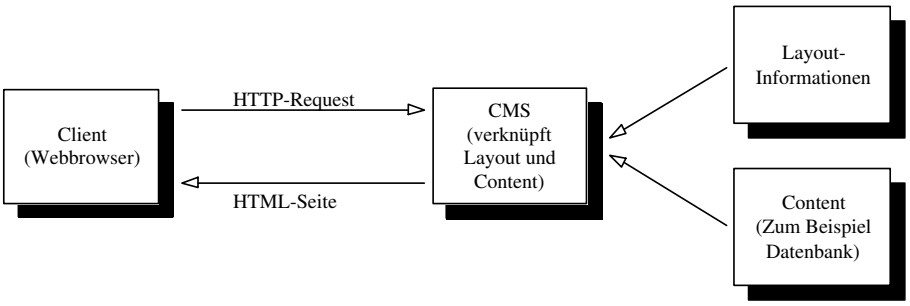


Abbildung 1.1: Beispiel eines CMS

Das CMS generiert die Seiten erst dann, wenn ein Request vorliegt (siehe Abbildung 1.1). Bei einem solchen verknüpft es die verlangten Informationen mit dem Layout und gibt dann die komplette HTML-Seite zurück. Die zurückgegeben Seite existiert jedoch nicht als ganzes auf dem Webserver.

Durch diesen Mechanismus kann ein Webadministrator jederzeit das Layout ändern, ohne das dies Einfluss auf bestehende Seiten hat. Diese werden nach der Änderung automatisch mit dem neuen Layout ausgegeben.

Natürlich war es nicht Ziel dieser Arbeit, ein komplettes CMS zu programmieren. Trotzdem soll das Konferenzsystem anpassbar und und in bestehende Websites intergrierbar sein. Folgende CMS[1] wurden für die Implementation des CTTM in betracht gezogen:

1.2.1.1 PHPNuke

PHPNuke (http://www.phpnuke.org) ist ein CMS mit sehr vielen verschiedenen Modulen wie Download, Diskussionsforen, Umfragen, Gästebuch, Statistiken oder SMS. In dieser Masse von Modulen haben wir jedoch keines gefunden, das eine optimale Wiedergabe von Konferenz-Vortragsplänen ermöglicht. Ferner basiert PHPNuke auf der weiter unten beschriebenen Scriptsprache PHP (Abschnitt 1.2.2), dies hätte ein Anpassen von PHPNuke für die nicht selbsverständliche Cross-Media ermöglicht. Leider ist PHPNuke zu komplex aufgebaut, um einfach "reinzuprogrammieren".

1.2.1.2 Zope

Zope (http://www.zope.org) ist ein CMS, welches in Python geschrieben wurde. Es läuft entweder unter einem CGI oder als Standalone. Zope wurde optimiert für Arbeitsprozesse mit verschiedenstufigen Zugriffsrechten für die einzelnen Administratoren. Ferner gibt es auch eine Sammlung von Zope-Modulen, die spezifischere Anwendungen erfüllen können. Jedoch wurden keine Module gefunden, die speziell für die Implementation unseres CTTM nützlich gewesen wären. Aufgrund der Struktur von Zope wurden auch auf Zope aufbauende Tools wie zum Beispiel Silva nicht weiter in Betracht gezogen.

[1] Siehe auch die gut sortierte Übersicht über CMS unter http://www.clueful.com.au/cmsdirectory/browse/Products%3aFree%20systems.

1.2.1.3 Wyona

Wyona (http://www.wyona.org) baut auf dem XML-Framework Cocoon (Kapitel 1.2.4) auf, ist aber auf die Verwaltung und Online-Editierung von Webseiten optimiert. Analog zu PHPNuke ist die Anpassung von Wyona möglich. Wyona schränkt jedoch Cocoon in denjenigen Bereichen ein, die für die Verwaltung von Konferenzdaten notwendig wären.

1.2.1.4 Schlussfolgerung

Die Verwendung eines CMS musste für unsere Arbeit schlussendlich verworfen werden, da wir nicht in erster Linie einen Texte oder Artikel publizieren, sondern Konferenzdaten in einem angepassten Format verwalten und wiedergeben wollen. Die Anpassung der CMS ist in jedem Fall zu aufwendig.

1.2.2 Serverpages

Eine weitere Möglichkeit für die Erstellung des CTTM ist die direkte Benutzung von sogenannten Serverpages. Einige CMS bauen auf diese auf, schränken jedoch durch vordefinierte Funktionen die volle Funktionalität der Serverpages ein.

PHP Hypertext Preprocessor (PHP)[2], Advanced Serverpages (ASP)[3] und Java Server Pages (JSP)[4] sind die im Moment am meisten verwendeten und deshalb auch bekanntesten Serverpages. Serverpages ähneln sich nicht in ihrer Programmierung, aber in ihrer Funktion: Grundsätzlich sind die Webseiten im einfachsten Fall statisch aufgebaut, an beliebigen Stellen (zum Beispiel wo der Inhalt einer Datenbank angezeigt werden soll) lässt sich dynamisch HTML-Code mit für HTML-Ausgabe optimierten Scriptsprachen oder Logiken erzeugen. Folgendes Beispiel zeigt die Verwendung von PHP in HTML-Code.

```
<table>
   <tr>
      <?php
         if ($value > 5) {
            echo ("<td>Wert ist grösser 5</td>");
         }
      ?>
   </tr>
</table>
```

ASP lehnt an VisualBasic an, JSP ist mit Java verwandt, und PHP ähnelt sehr stark C. PHP bietet neben einer grossen Sammlung von Bibliotheken und Anbindungen (PDF-Bibliothek, Datenbankzugriffe, Rechtschreibeprüfung, ...) die Möglichkeit Ausgaben in C-ähnlichem Code zu schreiben. Serverpages sind in der Regel aber darauf ausgerichtet HTML-Code auszugeben. Für PHP existiert zwar die Anbindung an einen XSLT-

[2] http://www.php.net
[3] http://www.asp.net
[4] http://java.sun.com/products/jsp

Prozessor (Sabloton und Expat[5]), dessen Benutzung durch PHP wurde jedoch nicht weiter verfolgt, da die Dokumentation von PHP in diesem Bereich sehr dürftig ist.

Zudem bestimmt die Technologie oft die Plattform. Die Serverpages sind, aufgrund des Entwicklungsumfelds, des Herstellers oder der Programmierer, noch an deren Betriebssysteme gebunden. ASP ist vor allem in der Windows-Welt zuhause. PHP kommt ursprünglich aus der UNIX-Welt, ist inzwischen aber auch für Windwos erhältlich. Kommerzielle[6] und nicht-komerzielle Emulationsversuche von ASP[7], zum Beispiel mit dem Apache-Server unter UNIX, sind erhältlich. Allerdings darf angezweifelt werden, wie sinnvoll ASP unter Nicht-Windows Systemen einsetzbar ist, da es seine Mächtigkeit vor allem aufgrund des ActiveX-Mechanismus unter Windows erhält. JSP ist ebenso wie Java selber plattformunabhängig.

Zwei wichtige Anforderungen an unser CTTM können nicht erfüllt werden, weshalb eine reine Serverpage-Lösung nicht in Frage kommt. Entweder ist die angebotene Lösung nicht plattformunabhängig (PHP und ASP), oder es können nur mit erheblichem Mehraufwand anderer Formate ausgegeben werden (JSP).

1.2.3 Eigenständige Serverlösungen

Bei einer eigenständigen Serverlösung müssen alle notwendigen Netzwerk-Protokolle implementiert werden (Listener am TCP-Port 80, HTTP-Protokoll,...). Der Vorteil dieser Lösung ist, dass eine absolute Freiheit in der Architektur der Anwendung möglich ist. Es können statische Binaries verwendet werden oder auch Interpreter-Skripte, wie zum Beispiel Perl, welches die notwendigen Funktionen für Ein- und Ausgabe über einen TCP-Port als Modul anbietet. Die immense Freiheit, die eine solche Lösung mit sich bringt, hat aber zwei entscheidende Nachteile: Durch die Benutzung von statischen Binaries ist die Plattformunabhängigkeit nicht mehr gewährleistet, zudem übersteigt der Programmieraufwand den Rahmen einer Diplomarbeit.

1.2.4 Cocoon

Bei unserer Suche nach möglichen Technologien für das CTTM stiessen wir auf Cocoon. Cocoon ist Teil des XML-Apache Open Source Projects[8] und ist ein Framework, welches zum Publizieren von Daten auf XML-Technologien aufbaut. Cocoon wurde für Performance und Skalierbarkeit im Bereich von XML-Technologien entwickelt und ermöglicht eine komplette Trennung von Inhalt, Layout und Logik. Ferner ist das zentralisierte Konfigurationssystem und das ausgereifte Caching eine Hilfe für die Erstellung, für den Einsatz und die Aufrechterhaltung von grossen und komplexen XML Server-Anwendungen. Cocoon arbeitet mit fast allen möglichen Datenquellen zusammen: Dateisysteme, relationale Datenbanken (RDBMS), LDAP und native XML-Datenbanken.

Ebenso kann Cocoon den Inhalt an verschiedene Ausgabemedien anpassen, wie zum Beispiel HTML, WML, PDF, SVG und RTF. Das Design von Coocon ist sehr flexibel aufgebaut: die Erweiterung von Funktionen ist ebenso möglich, wie das Erstellen von

[5] Erhältlich bei http://www.gingerall.com.
[6] ASP von Chilisoft, welche inzwischen von Sun aufgekauft wurde, http://www.chilisoft.com
[7] Nicht-kommerzielles ASP als Perl-Modul, http://www.apache-asp.org.
[8] http://xml.apache.org/cocoon

eigenen Modulen.

Innerhalb von Cocoon können auf XML-Dokumente Transformationen angewendet werden, so dass eine Präsentation der Daten im gewünschten Format generiert wird. Dies geschieht durch sogenannte Pipelines, welche durch Aufruf einer entsprechenden URL ausgeführt werden. Es können eine oder mehrere Transformationen, aber auch andere Aktionen vorgenommen werden. Beispielsweise können Daten in eine Datenquelle geschrieben werden, gleichzeitig wird auch die Ausgabe derselben Daten ermöglicht. Cocon erlaubt auch, Logik in XML-Files einzubinden. Die XML-Pipeline kann dadurch dynamisch gehalten werden.

Cocoon basiert auf einer Vielzahl bekannter Technologien. Um die Funktionsweise von Cocoon zu verstehen, ist es nötig, dass die zugrundeliegenden Technologien bekannt sind. Die folgenden Seiten fassen diese kurz zusammen und erklären die wichtigsten Funktionen.

1.2.4.1 Java

Java ist eine objektorientierte Programmiersprache, welche 1991-1995 von Sun Microsystems vollständig neu entwickelt wurde und ständig weiter entwickelt wird. Java ist plattformunabhängig, das heisst ein kompilierter Code ist auf allen gängigen Betriebssystemen ausführbar. Auf jedem Betriebssystem muss eine so genannte "Java Virtual Machine" (JVM) laufen, welche den Java Bytecode interpretiert. Weitere Features von Java sind das automatische Speichermanagement und das so genannte "Exception Handling".

Java kann für ganz verschiedene Aufgabenbereich eingesetzt werden: von simplen, lokalen GUI-Anwendungen bis zu komplexen Client-Server-Architekturen. Die Stärken von Java liegen im Multithreading, dem parallelen Ausführen von mehreren leichtgewichtigen Prozessen. Ebenso ist mit Java eine problemnahe Programmierung möglich. Viele Klassenbibliotheken für verschiedene Anwendungsbereiche und Probleme stehen bereits zur Verfügung und müssen nicht immer von neuem programmiert werden. Diese sind über das Java "Application Program Interface" (API) verwendbar. Mit einer Zeile im Java-Code kann man eine Vielzahl von Klassen und Methoden importieren.

1.2.4.2 Java Servlets

Java Servlets sind Java-Klassen, welche einen Webserver um zusätzliche Funktionen erweitern (Abbildung 1.2). Sie werden auf dem Webserver in einem sogenannten "Servlet Container" ausgeführt. Durch eine Anforderung (Request) von einem Browser via HTTP an die URL des Servlets wird ein neuer Thread innerhalb des Hauptprozesses der JVM auf dem Server gestartet. Jede Anfrage ist eine eigene Instanz. Das Servlet verarbeitet den Request innerhalb des Java-Programmes und gibt dem Client typischerweise im HTML-Format die Antwort (Response) zurück. Was für eine Art Verarbeitung dieser Java-Code bewirkt, ist offen (Datenbankenverbindungen, Dateisystemzugriffe, Netzwerkzugriffe,...).

Im Gegensatz zum klassischen CGI, wo für jeden Zugriff ein Prozess mit dem entsprechend aufgerufenen Programm gestartet wird, haben Servlets den Vorteil, dass sie wesentlich effizienter sind. Für jeden Zugriff auf eine URL die ein Servlet aufruft, wird

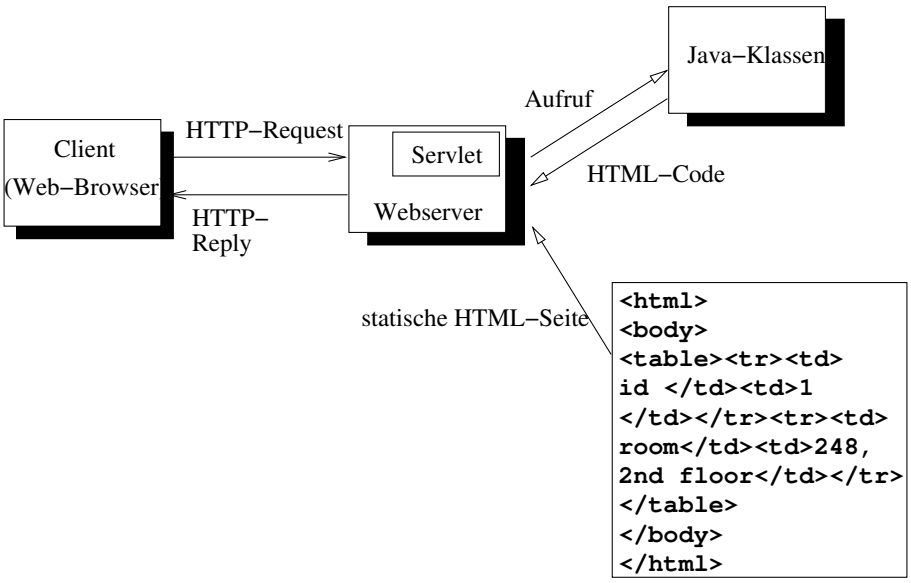


Abbildung 1.2: Schema eines Java-Servlet-Containers

nur ein Thread innerhalb der JVM gestartet. Oft bleiben auch teure Funktionen wie beispielsweise eine Datenbankverbindung offen, um so Zeit und Ressourcen zu sparen. Dasselbe trifft auch für FastCGI zu, das für jedes CGI einen eigenen Prozess (also eine teurere Funktion als ein Thread) startet. FastCGI bietet im Gegensatz zum klassischen CGI den Vorteil, dass mehrere Zugriffe auf dasselbe CGI im selben Prozess abgehandelt werden.

So stehen alle Möglichkeiten von Java zur Verfügung. Die Portierbarkeit von Servlets ist wie bei normalen Java-Anwendungen gegeben. Die Servlets sind plattformunabhängig, lediglich der Webserver muss Servlets unterstützen. Die bekanntesten Beispiele hierfür sind das Apache-Modul Jserv oder der Java-Server Tomcat. An den Client fallen im Normalfall geringe Ansprüche an, da die Servlets alle Aufgaben übernehmen.

Cocoon muss auf einem Webserver laufen, der Java-Servlet Container unterstützt. Dies ist zwar eine kleine Einschränkung, garantiert aber auch, dass unsere Lösung plattformunabhängig ist, da die dazu notwendigen Webserver für jede Plattform existieren. Für unsere Arbeit benutzten wir den Tomcat-Server.

1.2.4.3 Extensible Markup Language (XML) und XSL Transformations (XSLT)

Die meisten Files in Cocoon sind XML-Dokumente. XML ist ein weitverbreitetes Dokumentenformat. Unter http://www.w3.org/XML/ findet sich die Formulierung des genaue XML-Standards.

XML-Dokumente müssen oft weiterverarbeitet werden. Dies geschieht unter anderem mit XSLT. XSLT transformiert XML-Dokumente in XML, HTML oder reinen Text. http://www.w3.org/Style/XSL/ beschreibt den XSLT-Standard. Abbildung 1.3 zeigt

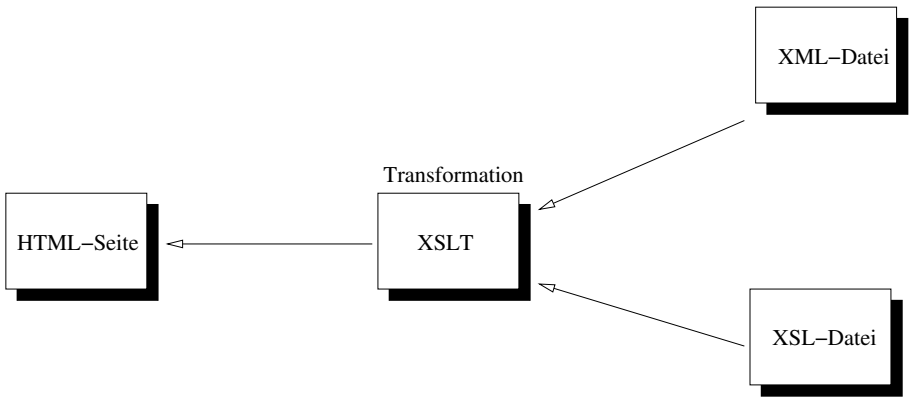


Abbildung 1.3: XSLT-Transformation

die vereinfachte Funktionsweise von XSLT.

Der XSLT-Prozessor kann entweder im Webbrowser (zum Beispiel Microsoft Internet Explorer 6.0), im Webserver oder als eigenständige Softwarelösung (zum Beispiel Saxon) integriert sein. In Cocoon wird defaultmässig Xalan verwendet, welcher in der Java-Version auf dem Server in einem Servlet läuft.

1.2.4.4 Java: Umgang mit XML

Die beide Technologien Java und XML müssen verknüpft werden. In den letzen Abschnitten wurden diese Technologien allgemein beschrieben, die wichtigsten Bezeichnungen erklärt und begründet, warum sie für unser Projekt geeignet sind. Eine der grundlegenden Funktionen in einer Programmiersprache ist das Lesen und Schreiben von Dateien. Selbstverständlich bietet auch Java solche Funktionen an. Jedoch ist es mit dem blossen Einlesen der XML- (und natürlich XSLT-)-Dateien noch nicht getan. Es müssen die Muster im XML-Dokument erkannt und die Elemente, bzw. eine Processing Instruction (PI) in XSLT, identifiziert und ausgeführt werden. Wie schon erwähnt, gibt es für Java eine Vielzahl von Klassen, unter anderem einige Parser. Hier werden diese praktischen Pakete kurz vorgestellt.

Simple API for XML (SAX): SAX ist ein Framework, welches ereignisbasiert vorgeht. Dies bedeutet, dass, sobald ein Element gefunden wurde, Instruktionen ausgeführt werden. SAX ist ein Java-API (`org.xml.sax.*`) und bietet einen "Contenthandler" als Interface an, der die vollständige Kontrolle über den Inhalt eines XML-Dokuments erlaubt, indem er während eines Parse-Durchgangs mit sogenannten "Callback-Routinen" auf bestimmte Ereignisse reagiert. Ein solches Ereigniss kann zum Beispiel das Vorkommen eines bestimmten Elementes oder Attributes sein. Ebenso gibt es Interfaces für Reaktionen auf Fehler (zum Beispiel bei nicht wohlgeformtem XML) und für andere Konstrukte wie zum Beispiel Kommentare. Die Reaktion selbst kann über von diesen Interfaces zur Verfügung gestellten Methoden vom Programmierer individuell festgelegt werden. Darüber hinaus gibt es bei SAX verschiedene Möglichkeiten, um den linearen Parsing-Vorgang durchzuführen. So kann festgelegt werden, ob während dem

Parsing ausser der standardmässigen Prüfung auf Wohlgeformtheit auch zusätzlich die Namensräume der Elemente überprüft und das Dokument gegen eine DTD validiert werden soll.

Document Object Model (DOM): SAX stellt zwar Methoden für den Zugriff auf XML-Dateien und für die Reaktion auf das Auftreten bestimmter XML-Konstrukte zur Verfügung. Allerdings können Dateien oder Elemente nicht mit SAX manipuliert werden. Dies kann jedoch mit DOM bewerkstelligt werden. DOM repräsentiert ein XML-Dokument als Baum, was die Verarbeitung vereinfacht, da Bäume keine exotischen Datentypen sind und deshalb praktisch jede Programmiersprache ausreichend Methoden für deren Be- und Verarbeitung zur Verfügung stellt. Das Modell, das mittlerweile auch offiziell durch das World Wide Web Consortium standardisiert ist, wurde – wie übrigens SAX auch – nicht für eine bestimmte Programmiersprache entworfen, sondern ist theoretisch in jeder Sprache implementierbar.

Ein weiterer Unterschied zwischen den Verfahren liegt darin, dass SAX das Dokument Element für Element nach seinen Vorgaben verarbeitet und dann den Vorgang als abgeschlossen ansieht. DOM dagegen liest das Dokument einmal vollständig in den Speicher, wo es auch bestehen bleibt, so lange kein neuer DOM-Baum angelegt oder der bestehende nicht explizit gelöscht wurde. Das hat zwar den Vorteil, dass sehr schnell auf das Dokument zugegriffen werden kann, führt aber andererseits zum Nachteil, dass je nach Grösse des XML-Dokuments viele Ressourcen beansprucht werden, um den Baum überhaupt erst in den Speicher zu bringen. In der Baum-Ansicht werden alle Elemente des XML-Dokumentes als Knoten repräsentiert, auf die direkt zugegriffen werden kann. Je nach Typ des Kontens sind verschiedene vorgegebene Manipulationen möglich. Genauso können auch ganze Elemente an beliebiger Stelle hinzugefügt oder komplette Dokumente "from scratch" aufgebaut werden.

JDOM: In SAX und DOM ist die Manipulation von XML-Dokumenten sowie das Auswählen eines Parsers relativ schwierig. JDOM behebt genau dieses Problem. Dazu geht es die Problematik von der "Java-Seite" an und optimierten ihre API demnach auf "Programmierbarkeit", was andeuten soll, dass vorherige Lösungen zwar alle notwendigen XML-Werkzeuge theoretisch zur Verfügung stellten, deren Verhalten aber nicht immer den gängigen Programmier-Standards entsprach. Das API steht unter `http://www.jdom.org` zum Download zur Vefügung. JDOM standardisiert das Ein- und Ausgabeverhalten von XML-Daten, was die Arbeit im Vergleich zu anderen Lösungen wesentlich vereinfacht. So können aus einer XML-Datei leicht Teile entnommen und als neue DOM-Objekte eingelesen werden. Diese Dokument-Objekte können dann, während sie im Speicher gehalten werden verändert, und an anderer Stelle abgelegt werden. Dabei bedient JDOM sich aus den beiden zuvor vorgestellten Ansätzen gleichermassen. Es nutzt eine, im Vergleich zur eigentlichen DOM-Spezifikation etwas abgespeckte, Baumstruktur als Repräsentation der XML-Daten, und verknüpft diese mit den performanten Methoden von SAX. Der Ansatz stellt ebenso Funktionen für die Validierung von XML-Daten gegen DTDs zur Verfügung und kann in diesem Zusammenhang auch Namensräume richtig interpretieren. Bei der praktischen Arbeit greift JDOM auf die mächtigen Collection-Klassen `List` und `Map` zurück, deren Verarbeitung mit Java zu den Grundlagen zählt.

Das API ist mit seinen Möglichkeiten beim Erstellen und Manipulieren von struktu-

rierten Daten wie XML geradezu prädestiniert für die Arbeit mit Cocoon. Im Gegensatz zu SAX und DOM, die so ausgerichtet sind, dass sie unabhängig von der letztlich verwendeten Programmiersprache sind, ist JDOM speziell auf Java und seine Stärken zugeschnitten. Dies erklärt auch, warum Cocoon JDOM verwendet.

1.2.4.5 Aufbereitung von XML-Dokumenten in Cocoon

Hier wird dargestellt, wie XML-Dokumente von Cocoon behandelt werden.

Pipeline: Cocoon führt Transformationen und andere Aufgaben in sogenannten Pipelines durch. In diesen werden die Transformationsschritte abgearbeitet. Jede Pipeline beginnt mit einem Generator, welcher auf ein auf einem Datenträger abgespeichertes XML-Dokument zugreift beziehungsweise eines aus einer Datenquelle erstellt. Nach dem Generator kann mit sogenannten Transformern weitergearbeitet werden, dieser transformieren den eingelesenen XML-Baum weiter. Am Schluss der Pipeline steht immer ein sogenannter Serializer, welcher den Ausgabecode in ein Dokument des entsprechenden Typs (HTML, WML, PDF, SVG, RTF,...) verwandelt.

Generator: Der Generator ist der Startpunkt der Pipeline. Er ist zuständig für die Bereitstellung von SAX-Events in der Pipeline. Der einfachste Generator ist der `FileGenerator`: Er liest ein XML-Dokument aus dem Filesystem oder von einer beliebigen URI, parst sie und sendet die SAX-Events weiter in der Pipeline. Der Generator ist so konzipiert, dass er unabhängig von einer Datei ist. Falls SAX-Events von einer anderen Quelle generiert werden können, kann diese auch verwendet werden, ohne über eine temporäre Datei auf dem lokalen Filesystem gehen zu müssen.

Transformer: Ein Transformer kann mit einem XSLT Stylesheet verglichen werden: Er nimmt ein XML-Dokument (beziehungsweise SAX-Ereignisse) und generiert ein neues XML-Dokument daraus (beziehungsweise neue SAX-Events). Der einfachste XML-Transformer ist der Xalan-Transformer (oder der `TraxTransformer`): Er wendet eine XSLT-Transformation auf die SAX-Events an, welche er in der Pipeline empfängt.

Serializer: Ein Serializer ist zuständig für die Transformierung von SAX-Events in ein präsentierbares Format. Serializer arbeiten immer am Ende der Pipeline. Cocoon bietet Serializer an um HTML, WML, PDF, VRML, WAP, PS, PCL, RTF, XLS, etc. generieren zu können. Natürlich können auch eigene Serializer in Java implementiert werden. Der einfachste Serializer ist der `XMLSerializer`, welcher die SAX-Events von der Pipeline erhält und diese als XML-Format wiedergibt.

1.2.4.6 Sitemap

Die Cocoon Sitemap ist das Konfigurations-XML, meist als `sitemap.xmap` benannt, welches die verfügbaren Pipelines und Komponenten für Webapplikationen definiert. Die Sitemap ermöglicht, dass bei Aufruf einer URI die entsprechenden Pipelines ausgeführt werden.

Innerhalb der Sitemap müssen alle Elemente, wie in jedem gültigen XML-File, einem Schema gehorchen. Dies ist in Fall der Sitemap eine DTD.

```
<?xml version="1.0"?>
<map:sitemap
 xmlns:map="http://apache.org/cocoon/sitemap/1.0">
   <map:components>
      <map:generators default="file"/>
      <map:transformers default="xslt"/>
      <map:serializers default="html"/>
   </map:components>
   <map:pipelines>
      <map:pipeline>
         <match:pattern="hello.html">
            <map:generate src="hello-content.xml"/>
            <map:generate src="hello2html.xsl"/>
            <map:serialize type="html"/>
         </match:pattern>
      </map:pipeline>
   </map:pipelines>
</map:sitemap>
```

Die Sitemap definiert Komponenten und Pipelines als Kind-Elemente innerhalb des `sitemap`-Elementes. Komponenten können Transformer, Generatoren oder Serializer sein. Ebenso lassen sich, um die Notation zu vereinfachen, Standard (default-)-Komponenten definieren. So wird im Beispiel als Standard der html-Serializer, der XSLT-Transformer und der File-Generator verwendet.

Hier folgt ein vereinfachtes Beispiel von Komponenten-Einträge in der Sitemap:

```
<?xml version="1.0"?>
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
   <map:components>
      <map:generators default="file">
      <!-- Hier folgt eine Auflistung aller Java-Geneatorklassen
           mit deren Bezeichnung für Abruf in einer Pipeline -->
      </map:generators>
      <map:transformers default="xslt"/>
      <!-- Hier folgt eine Auflistung aller Java-Transformerklassen
           mit deren Bezeichnung für Abruf in einer Pipeline -->
   </map:transformers>
   <map:serializers default="html"/>
   <!-- Hier folgt eine Auflistung aller Java-Serializerklassen
        mit deren Bezeichnung für Abruf in einer Pipeline -->
   </map:components>
   <!-- Hier folgen weitere Einträge -->
</map:sitemap>
```

In der Sitemap werden auch die Pipelines definiert, welche für das Processing der Anfragen zuständig sind. Sobald ein Matching einer Pipeline und einer URI-Anfrage gefunden

wird, werden die entsprechenden Abläufe, die in der Pipeline definiert sind, ausgeführt. Die URI ist nicht abhängig von Dateistrukturen auf dem lokalen Filesystem.

Hier folgte ein vereinfachtes Beispiel einer Pipeline in einer Sitemap:

```
<map:pipelines>
   <map:pipeline>
      <match:pattern="hello.html">
         <!-- Hier folgen die Elemente, welche
              die Anfrage ausführen: -->
         <map:generate src="hello-content.xml"/>
         <map:generate src="hello2html.xsl"/>
         <map:serialize type="html"/>
      </match:pattern>
      <match:pattern="hello.wml">
         <!-- Hier folgen die Elemente, welche
              die Anfrage ausführen: -->
         <map:generate src="hello-content.xml"/>
         <map:generate src="hello2wml.xsl"/>
         <map:serialize type="wml"/>
      </match:pattern>
      <match:pattern="dynhello.html">
         <!-- Hier folgen die Elemente, zum Beispiel mit XSP, welche
              die Anfrage ausführen: -->
         <map:generate src="hello-content.xsp" type="serverpages"/>
         <map:generate src="hello2html.xsl"/>
         <map:serialize type="html"/>
      </match:pattern>
      <match:pattern="querytest.html">
         <!-- Hier folgen die Elemente, zum Beispiel mit einer
              SQL-Transformation (siehe Skizze), welche
              die Anfrage ausführen: -->
         <map:generate src="queryfile.xml"/>
         <map:transform type="sql"/>
         <map:generate src="query2html.xsl"/>
         <map:serialize type="html"/>
      </match:pattern>
      <!-- Hier könnten weitere Match-Blöcke stehen -->
   </map:pipeline>
</map:pipelines>
```

Im ersten “match” wird ein XML-Files in ein HTML-File transformiert, im zweiten in ein WML-File. Im darauf folgenden dritten “match” erfolgt eine Transformation aus einem XSP (welches ein XML-File generiert, siehe nächster Abschnitt) und im letzten Match erfolgt von einem XML-File eine Datenbankquery, die darauf in eine HTML-Ausgabe transformiert wird (Abbildung 1.4 zeigt einen schematischen Ablauf einer solchen Transformation). Das XML-File wird durch den `FileGenerator` eingelesen, in diesem werden die SQL-Instruktionen durch den SQL-Transformer ausgeführt und in der Pipeline weitergegeben, worauf am Schluss der dadurch entstandene XML-Baum in HTML transformiert und ausgegeben wird. Erwähnenswert ist, dass nach einer HTML-

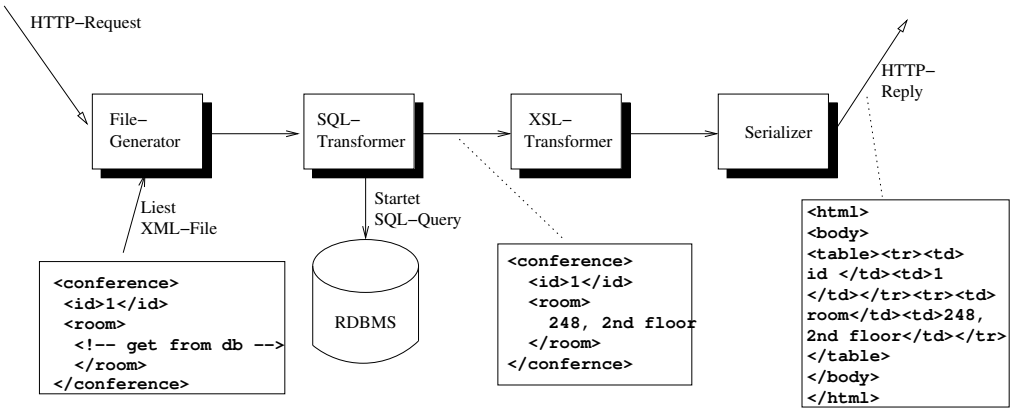


Abbildung 1.4: Schematischer Ablauf in einer Pipeline

Transformation im die Pipeline sogar XHTML entsteht. Der HTML-Serializer formt dies in sauberes HTML um, indem beispielsweise die `<br/>`-Tags in `<br>` verwandelt werden.

1.2.4.7 eXtensible Server Pages (XSP)

XSP wurden mit Cocoon 2.0 eingeführt und haben die Entwicklung von Web-Anwendungen in Cocoon stark vereinfacht. XSP, auch als "Middleware Programming Language" klassifiziert, ist ein gültiges XML-Dokument, welches PHP oder ASP sehr ähnlich ist. Im Gegensatz zu diesen enthält XSP nicht HTML-Elemente sondern XML-Elemente. Diese werden bei einem Request in die Ausgabe kopiert. Die Programmlogik ist in Form von Java-Code vorhanden. Die Logik wird ebenfalls bei Abruf ausgeführt, und deren Ausgabe wird in das XML-File kopiert. Das daraus entstandene XML-File kann in einer Cocoon-Pipeline weiterverarbeitet werden. Etwas genauer formuliert wird nicht ein XML-Dokument in ASCII-Text generiert, wie man sich vorstellen könnte, sondern SAX-Events.

Für das Schreiben eines XSP-Files gelten dieselben Regeln wie die Erstellung eines gewöhnlichen XML Dokumentes. Auch XSP wird mittels einer DTD definiert. Ein XSP-Dokument beginnt mit dem `xsp:page`-Element und beinhaltet Java als Scriptsprache[9], welche in `xsp:logic`-Elementen abgelegt wird. In einem weiteren wichtigen XSP-Element, dem `xsp:expr`-Element, lassen sich direkt Variablen ausgeben. Das `xsp:expr`-Element übernimmt die selbe Aufgabe wie die `toString()`-Methode in Java. Innerhalb der `xsp:logic`-Elemente lassen sich Java-übliche String-Ausgaben machen.

Wie im vorangehenden Abschnitt 1.2.4.6 dargelegt wurde, sind kleine Modifikationen in der Pipeline für die Ausführung der XSP-Files notwendig. Es muss ein entsprechender XSP-Generator gewählt werden, welcher durch eine Vorkompilierung des XSP-Files in eine Java-Klasse die XSP-Logik dann ausführt.

Beispiel eines XSP-Files:

[9] Es kann auch eine andere Sprache als Java in XSP verwendet werden, eine genauere Ausführung würde den Rahmen dieser Erklärungen hier sprengen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsp:page language="java"xmlns:xsp="http://apache.org/xsp">
   <page>
      <title>A Simple XSP Page</title>
      <content>
         <para>Hi there! I'm a simple dynamic page generated
               by XSP (eXtensible Server Pages).
         </para>
         <para>Brought to you by Cocoon at
               <xsp:expr>new Date()</xsp:expr>.
         </para>
         <para>The following list was
               dynamically generated:
         </para>
         <ul>
            <xsp:logic>
               String P = "TestText"
               for (int i=0; i<3; i++) {
            </xsp:logic>
            <li>
               Item <xsp:expr>i</xsp:expr>
            </li>
            <xsp:logic>
               }
            </xsp:logic>
         </ul>
         <xsp:element>
         <xsp:param name="name">
            <xsp:expr>"P".toLowerCase()</xsp:expr>
         </xsp:param>
         <xsp:attribute name="align">left</xsp:attribute>
         <i>This paragraph was dynamically generated by logic
            embedded in the page
         </i>
         </xsp:element>
      </content>
   </page>
</xsp:page>
```

Der XML-Code wird in XSP – also auch in obigem XSP-Codebeispiel – direkt ausgegeben. Die `xsp:expr`-Tags geben den von Java zurückgegebenen String aus. Im ersten `xsp:expr`-Tag in obigem Beispiel ist dies das aktuelle Datum. Innerhalb der Logic-Statements wird der Java-Code ausgeführt, zum Beispiel werden in obiger Schleife 3 Items ausgegeben: `<li>Item: 0</li><li>Item: 1</li><li>Item: 2</li>` und weiter unten folgendes XML-Element:

```
<testtext align="left">
   <i>This paragraph was dynamically generated
      by logic embedded in the page
   </i>
```

```
</testtext>
```

Der Vorteil von XSP im Vergleich zu XSLT ist die Möglichkeit, beliebige Java-Programme direkt auszuführen. Dies erlaubt beispielsweise Datenbank-Abfragen und -Auswertungen, was in in einer Pipeline mit XSLT unter Umständen mit mehreren SQL-Transformationen gemacht werden müsste. So kann zum Beispiel mittels XSP direkt ein Session-Management von Java für die Benutzerauthentisierung verwendet werden.

Die Idee von XSP ist auch, eine dynamische XML-Ausgabe aufgrund von Datenquellen oder Input-Daten machen zu können. In der lauffähigen Umgebung von Cocoon wird ein XSP-Dokument in ein Java .class-File kompiliert.

XSP ist sehr ähnlich zu JSP. Wie in JSP werden die XSP-Dokumente in Java-Quellcode transformiert und zu Java-Klassen kompiliert. Bei der Ausführung lesen die Klassen die Eingabe und kreieren eine Ausgabe. JSP generiert HTML-Code als Ausgabe, XSP aber XML. Dieser kann natürlich in die von Cocoon unterstützten Formate weitertransformiert werden.

Für das CTTM wurden in erster Linie die klassischen Pipeline-Transformationen verwendet, um so die entsprechenden Vorteile von Cocoon nutzen zu können. In den Fällen, wo es nicht möglich war, den Anforderungen mit den konventionellen Pipeline-Transformationen zu genügen, wurde auf XSP zurückgegriffen. Dies war zum Beispiel bei Benutzung des Form-Validators, des Java-Session-Managementes und der Authentisierung der Fall.

Mit den verschiedenen oben beschriebenen XML-Technologien lässt sich die Trennung von Logik, Inhalt und Präsentation[10] mit Cocoon erfüllen[11].

1.2.5 Backend

1.2.5.1 File Storage

Cocoon lässt einem bei der Wahl des Informationsspeichers sämtliche Freiheiten. Ursprünglich sollte die XML-SourceWriter-Klasse in Cocoon verwendet werden, um XML-Files als Informationsspeicher zu generieren. Jedoch sind wir an dessen Grenzen gestossen: Wenn ein XML-File generiert (und auf die Festplatte geschrieben) wird, ist es – aufgrund der Konsistenzgarantie[12] des lokalen Dateisystems – in der Regel nicht möglich, die Datei lesen zu können. Cocoon bringt in einer solchen Situation die Fehlermeldung, dass eine Datei nicht lesbar ist. Ebenso muss die Dateigrösse der XML-Files klein gehalten werden, da diese für die Transformation von Cocoon vollständig in den Arbeitsspeicher eingelesen werden. Dies führt zu einem immensen Ressourcenverbrauch im Arbeitsspeichers während einer Transformation. Bei der Annahme, dass eine mögliche Konferenz gegen 1000 Teilnehmer und 300 Events (sprich Vorträge, Vortragsserien etc.) humfasst, wurde festgestellt, dass Cocoon nicht in der Lage ist, grosse DOM-Bäume zu erstellen.

[10] Präsentation hier im Sinne von “Ausgabe”.
[11] Was nicht heisst, dass dies nicht anders hätte gemacht werden können.
[12] Nur eine Instanz kann das File schreiben, Mehrfachzugriffe ergeben Fehlermeldungen.

1.2.5.2 PDOM

Einen anderen Ansatz, um die grosse Anzahl von Knoten zu umgehen, ist die Verwendung von PDOM (Persistent DOM). Dazu wurde die PDOM Engine von GMD-IPSI, Darmstadt angeschaut[13]. Die PDOM-Engine beinhaltet ihren eigenen Cache und versucht nicht, den ganzen DOM-Tree in den Arbeitsspeicher einzulesen, sondern merkt selber, welche Bereiche immer wieder gebraucht werden. Somit können grössere Files verwendet werden. Um die nötigen Informationen abfragen zu können (das heisst, die gesuchten Knoten zu erhalten), muss XQL (XML Query Langauge) verwendet werden. PDOM ist jedoch relativ schwer in bestehende Lösungen zu integrieren, dies rechtfertigt den Mehraufwand von PDOM nicht. Ausserdem unterstützt die oben erwähnte PDOM-Version nur DOM 1.0. Cocoon jedoch benutzt schon seit mehreren Versionen DOM 2.0, was es unmöglich macht, PDOM mit der Cocoon-Version 2.0.4 zu verwenden. Eine stabiles PDOM, das Dom 2.0 unterstützt, wurde nicht gefunden.

1.2.5.3 Relationale Datenbank

Der naheliegendste Ansatz ist eine relationale Datenbank. Speichern und Abfragen von vielen Informationen, lassen sich mit einer Datenbank viel einfacher, effizienter und schneller organisieren als das Einlesen und Manipulieren von XML-Files. Dies hat aber den Nachteil, dass eine weitere Technolgie installiert werden muss.

Da Cocoon vollständig in Java implementiert ist, lässt sich simpel durch die Verwendung der entsprechenden JDBC-Klassen direkt auf relationale Datenbanken zugreifen. Dies wird von den Cocoon-Entwicklern auch als naheliegendste Methode empfohlen. Folgende Datenbanken wurden angeschaut.

HSQLDB: HSQLDB[14] ist eine von Cocoon mitgelieferte relationale Datenbank, welche die SQL92-Abfragesyntax unterstützt. Diese Datenbank wurde vollständig in Java implementiert und kann so aus einem Java-Archiv, wie dem des CTTM, verwendet werden. Dadurch verringert sich der Installationsaufwand erheblich. Die Datenbank kann durch einen Java-Client, als Standalone oder als Applet administriert werden.

MySQL:

> "Die im Umfeld von Web-Anwendungen wohl beliebteste Datenbank ist MySQL. Webmaster schätzen sie vor allem wegen ihrer leichten Administrierbarkeit und der als hoch eingeschätzten Geschwindigkeit. Aus demselben Grund ist es wohl auch die Datenbank, die man bei den meisten Webhostern standardmässig vorfindet."[15]

Die RDBMS MySQL[16], lässt sich wie die obige Datenbank über JDBC ansprechen. Der Nachteil zu HSQLDB ist, dass sie zu Tomcat und Cocoon separat installiert werden muss, und dass die JDBC-Treiber nicht 100% implementiert sind (da MySQL ja nicht

[13] Weitere Informationen: http://xml.darmstadt.gmd.de/xql.
[14] http://hsqldb.sourceforge.net
[15] Zitat aus c't 5/2003, Seite 143.
[16] http://www.mysql.com

die vollständige SQL92 Syntax unterstützt). Die Administration von MySQL kann über einen eigenen Client (wie MSQL cc) oder zum Beispiel über phpMyAdmin[17] erfolgen, was dem Administrator Datenbankänderungen ohne SQL-Kenntnisse ermöglicht.

1.2.5.4 Xindice

Xindice ist eine Native XML-Datenbank und ist besonders geeignet für das Verwalten von vielen kleinen XML-files. Somit wäre die Ausgabe recht nahe an derjenigen eines XML-Generators in Cocoon. Dies wäre eine Alternatvie zu den relationalen Datenbanken. Jedoch sind XML-Datenbanken noch nicht auf dem selben Performancelevel wie RDBMS. Zudem hätte ein sinnvolles Aufteilen von sämtlicher Informationen des CTTM gefunden werden müssen, was jedoch nicht möglich war.

1.3 Weitere Technologien, Email/SMS-Anbindung

Damit die Teilnehmer über Änderungen einer Konferenz informiert werden können, müssen Kommunikationsmedien verwendet werden, welche Cocoon nicht direkt unterstützt. Das Ziel war hierbei eine möglichst offene Lösung zu implementieren. Falls User informiert werden sollen, wird pro Informationsmedium und User ein XML-File mit den relevanten Daten im /tmp-Verzeichnis des Servers generiert[18]. Der Dateiname setzt sich entsprechend aus dem Namen des Mediums und Empfängeradresse zusammen:

`send2`<*medium value="sms | email"*><*empfängeradresse*>`.xml`

Dadurch ist die Anbindung an ein entsprechendes Interface offen gelassen. Natürlich können auch unter Cocoon weitere Components entwickelt werden, wie zum Beispiel eine SMTP-Komponente, welche ein Email gleich übers Internet versendet[19].

In unserer Beispielkonferenz wurde ein Listener (UNIX-Bash-Script) eingerichtet, welcher mittels eines CRON-Jobs jede Minute nach vorhandenen XML-Files mit obigem Namen schaut und diese gegebenfalls liest, die darin vorhandenen Informationen versendet und dann die Dateien löscht.

Für Email wurde das UNIX-Tool "mail" verwendet. Da für das Versenden von SMS zu einem Provider keine passenden UNIX-Commands gefunden wurden, mussten ein kleines Script in Python[20] implementiert werden, welches auf einem Beispiel eines SMS-Gateway-Betreibers beruht. Dieses Script erstellt eine HTTP-Verbindung zum SMS-Provider (In der Entwicklung des CTTMs wurden die SMS via ASPSMS[21] versandt) und übergibt ein XML-File, dessen Format durch die Spezifikationen des SMS-Providers vordefiniert ist. Weitere Infos, inklusive einer ausführlichen Dokumentation der XML-Anbindung, finden sich auf der entsprechenden Website.

[17] Weitere Informationen siehe http://www.phpwizart.net/projects/phpMyAdmin.
[18] Dies lässt sich eventuell weiter pipelinisieren.
[19] Siehe http://www.cocooncenter.de/cc/documents/resources/sendmail/index.html.
[20] http://www.python.org
[21] http://www.aspsms.com

Kapitel 2

Design

In diesem Kapitel wird das Design des CTTM betrachtet. Dabei steht nicht die eigentliche Implementierung im Vordergrund, sondern viel mehr die Funktionalitäten und Einschränkungen auf abstrakter Ebene.

2.1 Events, Event-Typen und Event-Eigenschaften

Ein Event ist ein beliebiges Ereignis an einer Konferenz. Dies kann zum Beispiel ein Track sein, aber auch ein Speech. Event-Typen definieren die Eigenschaften von Events der selben Art. Beispiele für Event-Typen sind Tracks, Talks und Speeches. Die Idee hinter dem Event-Typ ist, dass die Eigenschaften von Events nicht vorgegeben sind, sondern beliebig für eine Konferenz angepasst werden können.

Am besten wäre es gewesen, wenn in einem Event-Typ beliebige Eigenschaften definiert hätten werden können. Dies hätte jedoch zu zu komplexen Programmierarbeiten geführt. Deshalb wurden die häufigsten Event-Eigenschaften der untersuchten Websites von Abschnitt 1.1 in den Event-Typen vorgegeben. Dies sind:

- Name
- Inhalt
- Start-Datum
- End-Datum
- Start-Zeit
- End-Zeit
- Ort, Raum
- Personen (das heisst Referenten, Authoren,. . .)
- Weblink

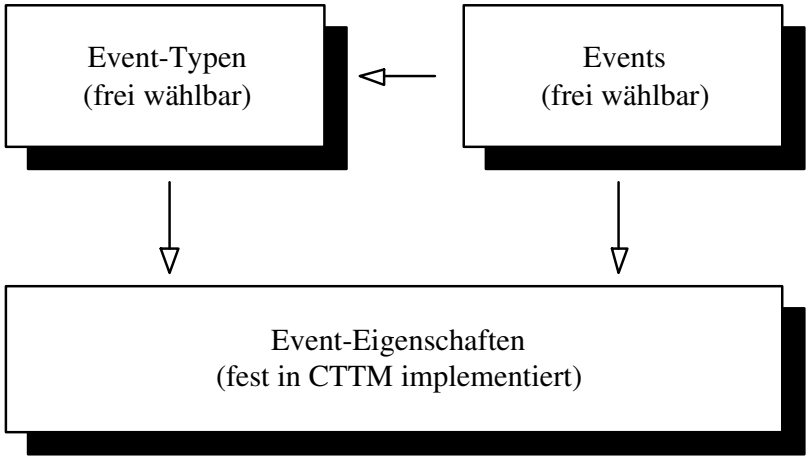


Abbildung 2.1: Events, Event-Typen und Event-Eigenschaften

- Gültige Tickets

Für die Definition eines Event-Typs ist einzig und allein ein Name notwendig, alle anderen Eigenschaften müssen nicht zwingend vorhanden sein. Zusätzlich zu den Eigenschaften muss jeder Event eine eindeutige Event-ID besitzen. Diese wird vom CTTM zugewiesen. Damit unter den Events eine Hierarchie entsteht, müssen die Event-IDs miteinander verknüpft werden. Dies geschieht dadurch, dass in jedem Event die Event-ID des Eltern-Events angegeben wird. Events die zuoberst in der Event-Hierarchie sind, geben als Eltern-ID den Wert "0" an.

Bei den Personen ist je nach Event-Typ jeder beliebige Personen-Typ möglich. Die Personen-Typen werden separat definiert und können nach Bedarf mit den Event-Typen verknüpft werden. So sind dann in einem Event nur noch die mit dem Event-Typ verknüpften Personen-Typen zulässig.

Im Event-Typ wird nur angegeben, ob in einem Event Informationen über gültige Tickets vorhanden sind. Welche Tickets gültig sind, wird erst mit dem Event gespeichert.

Die Event-Typen sind voneinander unabhängig. Das heisst es wird nirgendwo festgelegt, ob ein Event-Typ nur bestimmte Event-Typen als Nachkommen haben darf. Dies wurde so festgelegt, damit ein Event-Typ möglichst oft wiederverwendet werden kann und ein Administrator nicht selber komplizierte Verknüpfungen definieren muss. Das hat allerdings den Nachteil, dass ein Administrator selber für die Beziehungen unter den Events verantwortlich ist und schauen muss, dass diese gemäss seinen Ideen verknüpft sind.

Abbildung 2.1 zeigt die Beziehung von Events, Event-Typen und Event-Eigenschaften.

2.2 Benutzer-Eigenschaften

Die Benutzer-Eigenschaften sind fest vorgegeben und sehen wie folgt aus:

- Benutzername
- Passwort
- Email
- Name
- Vorname
- Titel
- Universität, Firma
- Natel-Nummer
- Website-Link
- Benutzerlevel

Damit ein Benutzer sich registrieren kann, braucht er nur die drei ersten Punkte anzugeben. Der Benutzerlevel wird automatisch auf “1” gesetzt. Dieser Wert ist reserviert für einen Benutzer, der sich nur seinen eigenen Konferenzplan zusammenstallen kann. Benutzerlevel “10” ist für Administratoren reserviert. Die dazwischen liegenden Werte können für zukünftige weitere Benutzer-Level benutzt werden.

2.3 Darstellung

2.3.1 Website-Layout

Das gesamte Layout der Website sollte möglichst einfach anpassbar sein, ohne dass dafür ein komplettes CMS hätte programmiert werden müssen. Dies konnte mit Hilfe von XSLT-Variablen umgangen werden. Das ganze Layout wird in einem XSLT-File abgespeichert. In diesem File sind neben der XSLT-Deklarierung nur HTMl-Elemente erlaubt. Ausser an zwei Stellen, wo mittels zweier XSLT-Variablen[1] der Titel der Webseite und das eigentliche Konferenzprogramm eingefügt.

2.3.2 Programm-Layout

Das Konferenzprogramm muss auf eine sinnvolle Art dargestellt werden. Einerseits sollte ein Administrator möglichst viele Freiheiten haben, andererseits ist eine einfache und vollkommen veränderbare Event-Darstellung sehr schwer zu Programmieren.

Die Struktur der Darstellung der Events wurde vorgegeben, die Tabelle in denen sich die Event-Informationen befinden können nur mit Hilfe von CSS angepasst werden. Eine Umstellung der einzelnen Informations-Fragmente muss direkt im Quellcode vorgenommen werden.

Drei ähnliche Darstellungsarten des Konferenzprogramms mussten berücksichtigt werden:

[1] Siehe Abschnitt 4.2.2.

- Programm für nicht-registrierte Benutzer
- Programm für registrierte Benutzer
- Persönliches Programm für registrierte Benutzer

Ersteres unterscheidet sich von zweitem nur von einer zusätzlichen Zeile: In dieser kann ein registrierter Benutzer einen Event seinem persönlichen Konferenzplan hinzufügen oder wieder entfernen. Zusätzlich kann der Benutzer angeben, ob er bei Änderungen des Events via Email oder via SMS informiert werden will.

Für das Design des persönlichen Programms mussten folgende Fragen beantwortet werden:

- Was passiert wenn ein Benutzer einen Event seinem persönlichen Plan hinzufügt, der Kinder hat?
- Wie werden Events ohne Datums- und Zeitangabe gehandhabt?
- Wie werden die aus der Konferenzstruktur "herausgerissenen" Events im persönlichen Programm dargestellt?

Wählt ein Benutzer einen Event aus, der Kind-Events hat, so wird dessen ID normal gespeichert. Bei der Darstellung des entsprechenenden Events zeigt das CTTM nur die direkt folgenden Kinder. Es wäre zwar möglich auch Kindeskinder etc. anzuzeigen, allerdings wäre dies mit einem sehr schlecht lesbarem Layout verbunden, weshalb darauf verzichtet wurde.

Bei Events ohne Datums- und Zeitangeb nimmt das CTTM an, dass diese immer eng mit einem Eltern-Event verknüpft sind (zum Beispiel Speeches innerhalb eines Talks). Events ohne Zeitinformation müssen deshalb immer als kinderlose Events vorhanden sein. Sie können nur durch die Auswahl ihres Eltern-Events ins persönliche Programm übertragen werden. Diese Einschränkung war notwendig, um weitere Probleme im persönlichen Programm zu verhindern.

Im persönlichem Programm werden sämtliche Events chronologisch dargestellt. Zuerst werden sie nach Start-Datum sortiert, dann nach der Start-Zeit. Zeitlose Events wären also nicht sortierbar gewesen, und es hätte keinen Sinn gemacht sie ans Ende oder an den Anfang des persönlichen Programms zu setzen, da sie dort ohne Zeitinformation nicht viel Nutzen haben. Zeitlose Events können aber durch Auswahl ihreres Eltern-Events im persönlichen Programm erscheinen.

Bei der Selektierung der Events für das persönliche Programm wird nur die Event-ID gespeichert. Mit dieser lässt sich zwar die ID des Eltern-Events herausfinden, die komplette Konferenzstruktur lässt sich jedoch nur sehr mühsam wieder herstellen. Dies hat zur Folge, dass im persönlichen Programm keine Links zu Eltern-Events oder Kinder-Events eines selektierten Events vorhanden sind. Das ist mühsam, lies sich aber nicht anders machen, da in einem Event nur die ID des Eltern-Events gespeichert wurde. Dies hätte verhindert werden können, wenn die Beziehung zwischen Eltern-Event und Kind-Event auch in der anderen Richtung gespeichert worden wäre (in einem Event hätten die IDs sämtlicher Kinder-Events gespeichert werden müssen). Deshalb können im persönlichen Programm auch keine Events hinzugefügt sondern nur entfernt werden.

Eine Überprüfung des persönlichen Programms auf zeitliche Überschneidungen von Events wäre zwar wünschenswert und machbar, konnte jedoch aus zeitgründen nicht genauer betrachtet werden.

Kapitel 3

Dokumentation für Benutzer

Die grundlegenden Technologien sowie das Design, auf welchem das CTTM aufbaut, wurden in den vorangehenden Kapiteln beschrieben. Bevor jedoch die Adminstrierung unseres CTTMs und die Entwicklungs-Möglichkeiten in den beiden folgenden Kapiteln beschreiben werden, folgt hier die Benutzung eines betriebsbereiten CTTMs anhand der Beispielskonferenz.

Zuerst wird in diesem Kapitel erläutert, wie die "Public Pages" mit deren gesamten Konferenzübersicht verwendet werden (Abschnitt 3.1). Danach wird die Benutzung von "myAccount", dem benutzerspezifischen Profil, in Abschnitt 3.2 mit der dazugehörigen Anmeldeprozedur (Abschnitt 3.2) und dem Bearbeiten der persönlichen Daten erklärt. Zum Schluss wird in Abschnitt 3.3 die Benutzung des persönlichen Stundenplans erklärt, insbesondere wie dieser erstellt wird (Abschnitt 3.3.1) und wie der Benutzer ihn anschauen kann (Abschnitt 3.3.2). Die Benutzung sollte sich schlussendlich eigentlich nicht gross von irgendwelchen anderen Webapplikationen unterscheiden und intuitiv klar sein.

3.1 Benutzung der Public Pages

Unter "Public Pages" werden sämtliche Webseiten der Konferenz, welche keinen Login (Authentisierung) benötigen, verstanden. Dies können statische Webseiten sein, wie zum Beispiel Informationen über die Hotelzufahrt, über die Zufahrt zum Konferenzzentrum, Registrationsinformationen,...

Die wichtigste "Public Page" aus Sicht des CTTM ist jedoch diejenige, welche eine Übersicht über das gesamte Konferenzprogramm anbietet. Ein Klick auf den Menüpunkt "Program" (auf der linken Seite der Beispielseiten) führt zum Konferenzprogram. In diesem kann ein beliebiger Benutzer das Program anschauen und sich durch die diversen Events klicken. Es existieren zwei Ansichten: "Compact" und "Detail". Während erstere nur den Namen und den Typ des Events anzeigt, gibt zweitere sämtliche Informationen zu den angezeigten Events wieder. Zwischen diesen beiden lässt sich beliebig hin und her wechseln. Je nach vordefiniertem Typ werden die entsprechenden Angaben in der detailierten Ansicht visualisiert. Man beachte, dass die Auflistung der Events

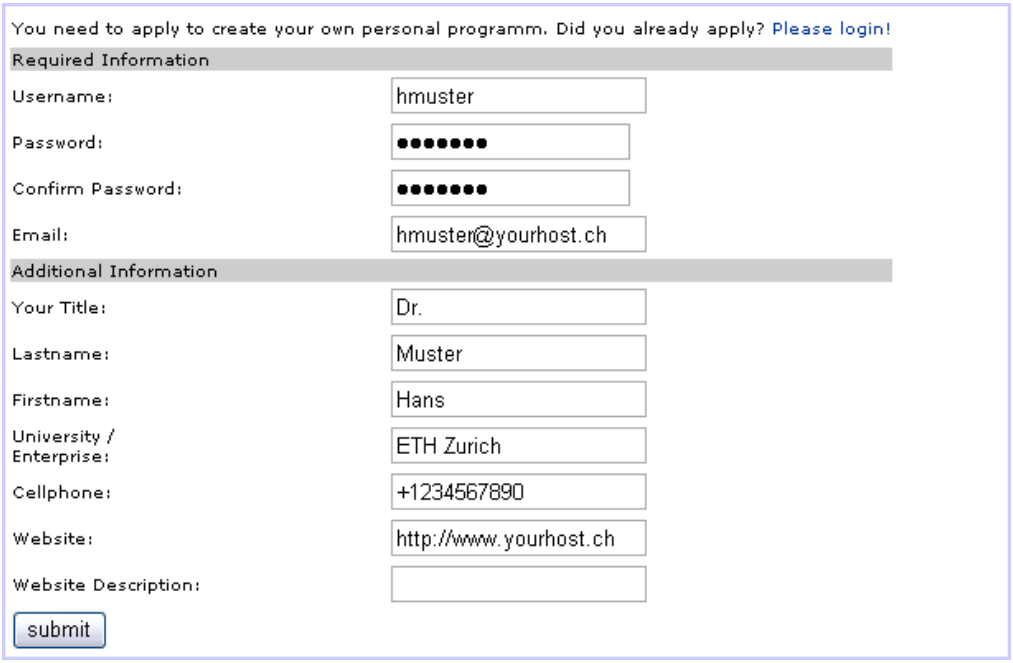


Abbildung 3.1: Benutzer-Anmeldung

hierarchisch erfolgen. Ein Benutzer kann sich durch anklicken des Event-Namens die Kinder-Events anschauen (Der Link ist aktiv, falls Kinderevents existieren). Um zum Vaterevent zurückzukehren muss zuoberst in der Tabelle auf den Namen des Vaterevents geklickt werden. So lässt sich das ganze Konferenzprogramm anschauen.

3.2 Eigener Benutzername und Passwort

Um sein persönlichen Konferenzplan zusammenzustellen, muss der Benutzer sich zuerst registrieren. Dies geschieht via URL (`http://localhost:8080/conference/apply.html`[1]) oder dem Menüpunkt "New Account" der Beispielkonferenz. Wie in Abbildung 3.1 ersichtlich ist, müssen Benutzername, Passwort und Emailadresse angegeben werden. Ebenso sind weitere Angaben (Titel, Vor- und Nachname, Natelnummer, Institution, Webpage,...) optional möglich. Mit der Bestätigung der Formulareingabe wird der Benutzeraccount bereits freigeschaltet. Spätere Änderungen der persönlichen Daten sind in "myAccount" jederzeit möglich.

[1] Anstelle von `localhost` kann eine beliebiger Domainname verwendet werden, auf welchem das CTTM läuft. Der Allgemeinheit wegen werden die noch folgenden URLs des CTTM ebenfalls `localhost` benutzen.

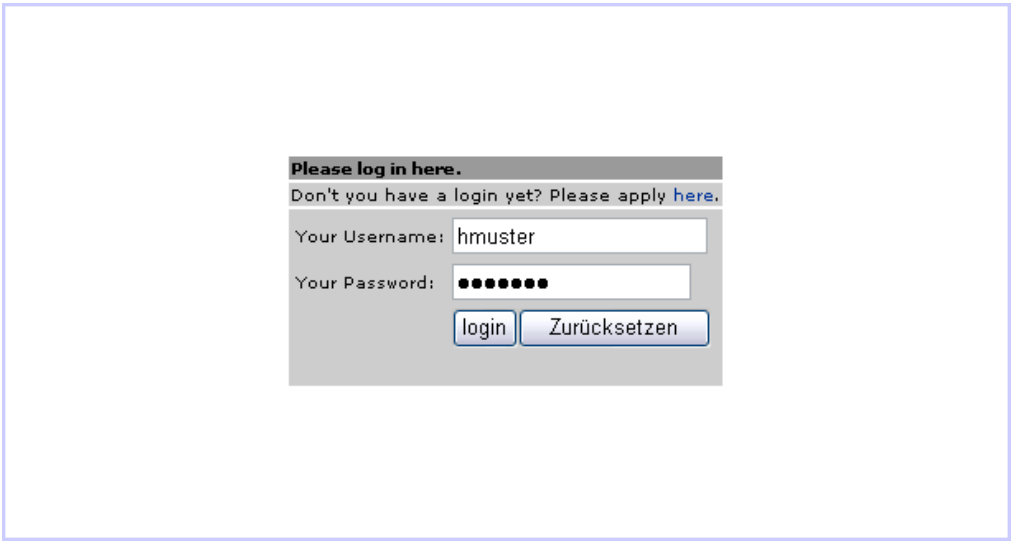


Abbildung 3.2: Benutzer-Login

3.3 Persönlicher Stundenplan bearbeiten und anschauen

Um eigene, persönliche Ansichten der Konferenz-Referate zu erhalten (das heisst durch klicken der Hyperlinks “myPlan” oder “myConf”), muss sich der Benutzer anmelden. Dies geschieht, sobald der Benutzer auf eine entspechende Seite springen möchte (Abb. 3.2). Der Benutzername und das Passwort werden nach Eingabe und drücken des Login-Buttons überprüft. Bei einem gültigem Login wird die gewünschte Seite angezeigt. Falls noch kein Benutzername und Passwort vorhanden sind, muss sich der Benutzer zuerst gemäss Abschnitt 3.2 anmelden.

3.3.1 Events auswählen und ändern

In “myConf” kann ein eingeloggter Benutzer seinen persönlichen Konferenzplan zusammenstellen. Die Darstellung der Events ist identisch zu derjenigen des normalen Programs (siehe 3.1), ist aber um drei Checkboxen erweitert:

Add to myPlan: Durch Auswahl dieser Checkbox wird der Event in den persönlichen Konferenzplan übertragen. Hat der Event Kinderevents, welche eine Datums- oder Zeitangabe besitzen, so müssen diese separat gewählt werden. Dies geschieht durch Klicken auf den Namen des Events, um zu den Kindern zu gelangen. In einem weiteren Schritt können die Kinderevents analog angewählt werden. Bei Kinderevents ohne Datums- oder Zeitangabe werden diese automatisch mitangewählt.

Notify me by SMS: Wenn zusätzlich diese Checkbox angeklickt wird, so wird der Benutzer über Änderungen für diesen Event automatisch via SMS informiert.

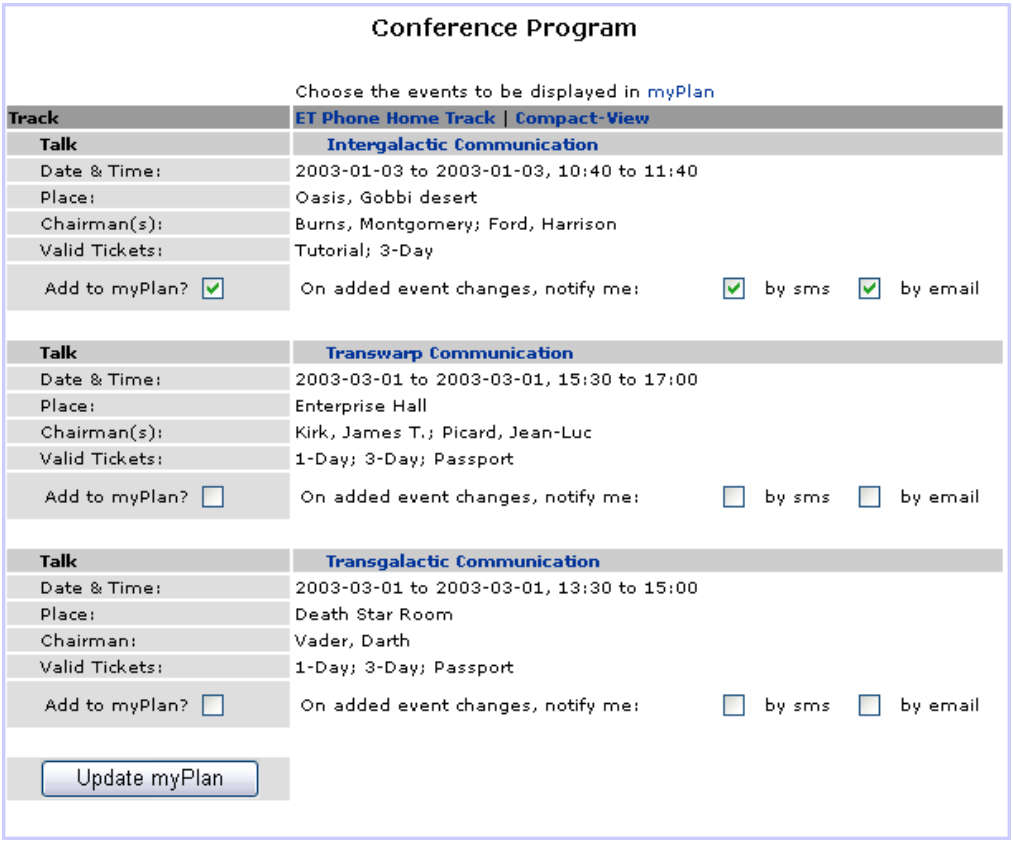


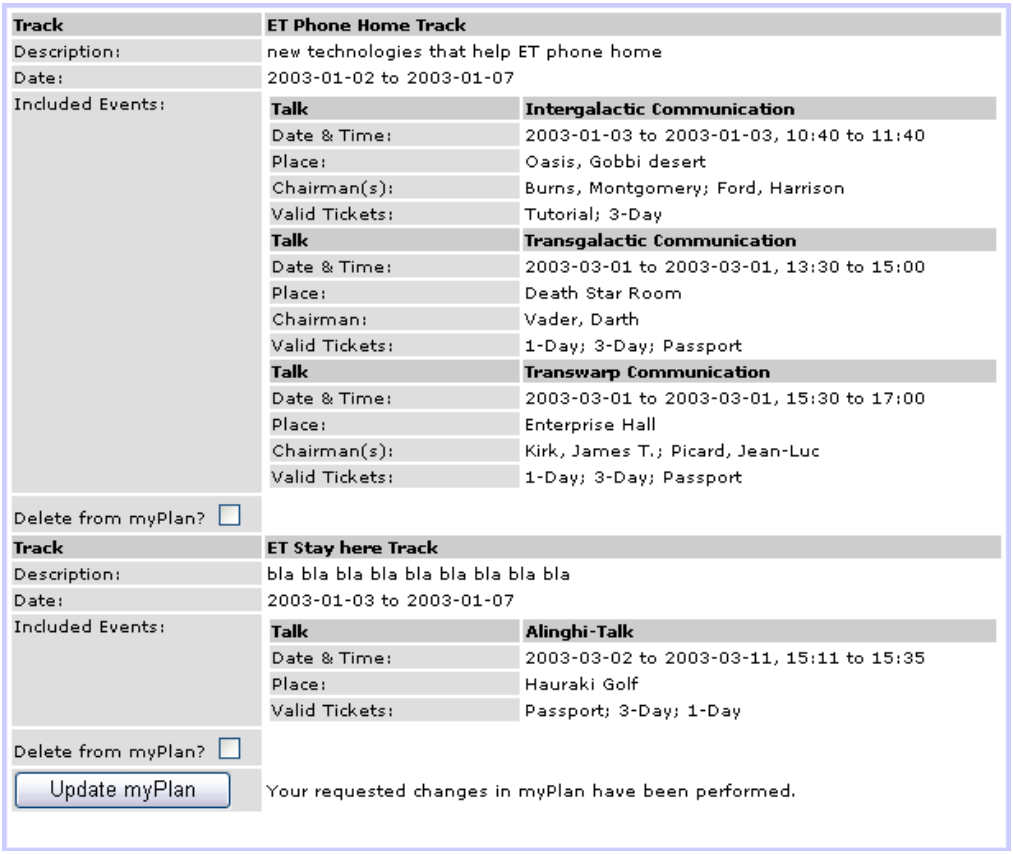
Abbildung 3.3: Zusammenstellen des persönlichen Konferenzplans

Notify me by Email: Analog gilt dasselbe hier: Falls diese Checkbox beim Hinzufügen des Events zum "myPlan" gewählt wird, wird der Konferenzteilnehmer per Email über eine Änderung des Events informiert.

Als Beispiel zeigt die Abbildung 3.3, wie die Tracks (das heisst ein Typ eines Events) der Beispielkonferenz aussehen. Die Informationen werden nur gespeichert, wenn nach dem Anklicken noch auf den "Update myPlan"-Button gedrückt wird.

3.3.2 Events anschauen und abwählen

Wie der interessierte Leser feststellen mag, werden in "myPlan" nur noch die angewählten Events dargestellt, das heisst in "myPlan" kann ein Benutzer seinen selber ausgewählten Konferenzplan anschauen. Hier werden wie beim Konferenzprogramm der "Public Pages" (Abschnitt 3.1) die Events chronologisch angezeigt, jedoch nur noch die vom Benutzer gewählten Events. Die übrigen Events werden nicht angezeigt. Dies ermöglicht dem Benutzer, einen indivduellen Stundenplan für sich zusammenzustellen, welcher nur noch die Events darstellen, die ihn auch interessieren. Ebenfalls werden bei selektierten Events, die Kind-Events enthalten, diese ebenso angezeigt. Allerdings wer-



Track	**ET Phone Home Track**	
Description:	new technologies that help ET phone home	
Date:	2003-01-02 to 2003-01-07	
Included Events:	**Talk**	**Intergalactic Communication**
	Date & Time:	2003-01-03 to 2003-01-03, 10:40 to 11:40
	Place:	Oasis, Gobbi desert
	Chairman(s):	Burns, Montgomery; Ford, Harrison
	Valid Tickets:	Tutorial; 3-Day
	Talk	**Transgalactic Communication**
	Date & Time:	2003-03-01 to 2003-03-01, 13:30 to 15:00
	Place:	Death Star Room
	Chairman:	Vader, Darth
	Valid Tickets:	1-Day; 3-Day; Passport
	Talk	**Transwarp Communication**
	Date & Time:	2003-03-01 to 2003-03-01, 15:30 to 17:00
	Place:	Enterprise Hall
	Chairman(s):	Kirk, James T.; Picard, Jean-Luc
	Valid Tickets:	1-Day; 3-Day; Passport
Delete from myPlan? ☐		
Track	**ET Stay here Track**	
Description:	bla bla bla bla bla bla bla bla bla	
Date:	2003-01-03 to 2003-01-07	
Included Events:	**Talk**	**Alinghi-Talk**
	Date & Time:	2003-03-02 to 2003-03-11, 15:11 to 15:35
	Place:	Hauraki Golf
	Valid Tickets:	Passport; 3-Day; 1-Day
Delete from myPlan? ☐		
Update myPlan	Your requested changes in myPlan have been performed.	

Abbildung 3.4: Anschauen des persönlichen Konferenzplans

den aus darstellungstechnischen Gründen nur die direkten Nachkommen angezeigt (Abbildung 3.4). Unerwünschte Events können durch anklicken der entsprechenden Checkbox gelöscht werden, dies muss allerdings mit dem “submit”-Button bestätigt werden.

Kapitel 4

Dokumentation für Administratoren

In diesem Kapitel wird erläutert, wie ein Web-Verantwortlicher einer Konferenz mit dem CTTM einen Zeitplan einrichten kann. Diese Anleitung ist in erster Linie für Administratoren oder Webmaster gedacht, welche einen Webserver einer Konferenz sowie deren Webseiten erstellen, konfigurieren, installieren und betreuen. Die Lösung kann auch alternativ auch als eine beschränktere Dienstleistung als eine komplette Webpräsenz angeboten werden. Beispielsweise können je nach Konfiguration statt einer kompletten Webseite nur ein HTML-Frame oder ein HTML-Fragment wiedergegeben werden, welches in eine andere Website integriert werden kann. Falls dies der Fall ist, kann in diesem Kapitel direkt zum Abschnitt 4.2.2 gesprungen werden.

4.1 Installation

4.1.1 Grundsystem

Es wird davon ausgegangen, das dem Webmaster oder Administrator ein Server mit vorgegebener Hardware und Betriebssystem zur Verfügung gestellt wird. CTTM läuft auch auf einem Server, welcher bereits andere Dienste beherbergt. Ebenso lässt sich unsere Konferenz auf einem bestehenden Webserver[1] oder als zweiter Webserver auf der gleichen Hardware (zum Beispiel auf Port 8080) betreiben, je nach Installation.

4.1.2 Java

Falls noch nicht erfolgt, muss Java auf dem Server installiert werden. Java kann von `http://java.sun.com` bezogen werden. Die J2SE-Edition ist ausreichend für unsere Anwendung. Für gewisse Unix- und Linux-Distributionen ist anzumerken, dass es bereits vorkonfigurierte Packages gibt, die die Installation wesentlich vereinfachen. Falls dies der

[1] Voraussetzung für den Betrieb unserer Konferenzlösung ist, dass der bestehende Server Java Servlets unterstützt.

Fall ist, wird empfohlen, in erster Linie auf diese Packages zuzugreifen. Der Nachteil ist, dass in diesen Packages nicht immer die neueste Version vorhanden ist. CTTM läuft mit der von uns verwenden Version J2SE (1.3.1_04) einwandfrei. Für die genaue Installation von Java auf einem Server sei auf die oben erwähnte Java-Webseite von SUN verwiesen.

4.1.3 Jakarta Tomcat

Als nächstes muss ein Webserver mit Servlet Containter für die Benutzung von Cocoon gewählt werden. Wir haben unter dem Java-Webserver mit integrierter Servlet Engine, "Tomcat", gearbeitet. "Tomcat" ist ein Subprojekt des Jakarta-Apache-Projektes[2] und welcher der Java Servlet- und JavaServer Page-Server der Apache Gruppe ist. Alternativ liessen sich beliebige Webservers mit integriertem Servlet Container benutzen, wie zum Beispiel das HTTP-Apache-Modul JServ[3] oder – als Vertreter aus dem kommerziellen Bereich – der Applicaton Server "WebSphere" von IBM[4]

Wie erwähnt, werden wir uns ausschliesslich dem Betrieb von Cocoon unter Tomcat widmen. Wir benutzen die Version 4.0.4 mit dem Servlet Container (Catalina), welcher die Servlet 2.3 und JSP 1.2-Spezifikationen erfüllt. Ebenso behandeln wir die Installation von Tomcat unter einem UNIX-Derivat, die Installation unter Windows ist äquivalent, lediglich die Dateipfade bzw. das Setzen von Variablen sind anders. Für genauere Informationen wird auf die Installationsanleitung von Tomcat unter Windows auf der Jakarta-Webpage (Link siehe oben) verwiesen.

Folgende Schritte müssen gemacht werden:

1. Tomcat downloaden (von http://jakarta.apache.org/tomcat) und eventuell den Source-Code kompilieren, falls zum gewählten Betriebssystem kein Binary existiert.

2. Tomcat auf dem lokalen Filesystem auspacken:

```
# cd /opt/jakarta
# gunzip jakarta-tomcat-4.0.4.tar.gz
# tar -xvf jakarta-tomcat-4.0.4.tar
# mv jakarta-tomcat.4.0.4 tomcat
```

3. Envrionment anpassen:

```
# JAVA_HOME=/pfad/zum/JDK
# TOMCAT_HOME=/opt/jakarta/tomcat
```

   Falls diese Variablen nicht vor jedem Start von Tomcat gesetzt werden sollen, können sie in der Datei `$TOMCAT_HOME/bin/tomcat.sh` vor dem ersten 'if'-statement eingefügt werden.

[2] http://jakarta.apache.org/tomcat
[3] http://java.apache.org/jserv
[4] http://www-3.ibm.com/software/info1/websphere/index.jsp. Übrigens verwendet IBM Cocoon für XSLT in WebSphere.

4. Nun ist Tomcat bereit zum Start. Um ihn zu starten, muss nur noch ein cd in das Verzeichnis `$TOMCAT_HOME/bin` eingesetzt werden und folgende Sequenz ausgeführt werden[5], welche im selben Verzeichnis liegt:

```
# cd /opt/jakarta/tomcat/bin
# ./startup.sh
```

Der Startup von Tomcat dauert eine ganze Zeit, dies liegt unter anderem daran, dass die Tomcat .war-Archive, welche im Verzeichnis der Webanwendungen, `$TOMCAT_HOME/webapps`, liegen, ausgepackt werden müssen.

Wenn die obigen Schritte erfolgt sind, sollte mit einem Browser auf die Seite von Tomcat, in unserem Beispiel http://localhost:8080, gesprungen werden können. In unserem Beispiel läuft der Webserver nicht auf dem Standardport 80, da dies die Grundkonfiguration von Tomcat ist. Wir werden alle Beipiele unter dem Port 8080 aufführen. Falls der Betrieb des Servers auf dem Port 80 erfolgen soll, muss in der Datei `$TOMCAT_HOME/conf/server.xml` das Attribut `port="8080"` im Element

```
<Connector className="org.apache.catalina.connector.http.HttpConnector"
           port="8080" minProcessors="5" maxProcessors="75"
           enableLookups="true" redirectPort="8443"
           acceptCount="10" debug="0" connectionTimeout="60000"/>
```

auf `port="80"` gesetzt werden.

Unter der obigen URL sollte nicht nur eine Webpage erscheinen, sondern die Beispiele auf dieser Webpage sollten problemlos ausgeführt werden können. Falls irgendwelche Fehler beim Ausführen eines Servlets oder von JSP oder beim Zugriff auf die Welcome-Page von Tomcat auftreten sollten, müssen die gesetzten Variablen und die Umgebung von Tomcat nochmals überprüft und der Server nochmals gestartet werden[6]. Zusätzlich geben die Logfiles in `$TOMCAT_HOME/logs` Hinweise auf einen möglichen Fehler.

4.1.4 Cocoon

Da Cocoon 2.0.3 in unserer Weblösung enthalten ist, braucht es nicht separat installiert zu werden. Für Interessierte verwiesen wir auf die Dokumentationen auf der Website von Cocoon[7].

4.1.5 Datenbank

Nun muss noch ein Backend gewählt werden. Die Arbeit basiert auf der Verwendung einer SQL-Datenbank. Hierzu kann beispielsweise die in Cocoon mitgelieferte Datenbank HSQLDB verwendet weren. Achtung, HSQLDB 1.61 und Cocoon 2.0.3, welche wir gemeinsam im Einsatz hatten, interpretierten das SQL-Statement "`AS`" in den SQL-`SELECT`s in Cocoon nicht, dies ist ein Fehler von HSQLDB. Dies führte zur Rückgabe

[5] Unter Unix wird das Shellscript `startup.sh` ausgeführt, unter Windows das Batch-File `startup.bat`.

[6] Tomcat kann, analog zum Aufstarten, mit `./shutdown.sh` bzw. mit `shutdown.bat` beendet werden.

[7] http://xml.apache.org/cocoon

von falschen Elementnamen. Wir hoffen, dass dieser Fehler in späteren Versionen behoben sein wird. HSQLDB[8] läuft unter Java und kann dadurch nicht die Performance von andere Datenbanken erreichen. Aus diesen beiden Gründen wird die Verwendung der Open-Source-Datenbank MySQL[9] empfohlen. Dennoch wird kurz auf die Verwendung von HSQLDB eingegangen:

4.1.5.1 HSQLDB

HSQLDB-Administration: Zur Administration von HSQLDB stehen diverse Clients (DatabaseManager) zu Verfügung, ein Java-Applet oder ein Java Client (was auch wir verwendet haben). Dieser befindet sich im Verzeichnis `demo` in der unserem Archiv beigelegten Datei `sqldb-v.1.61.zip` und lässt sich auf einem GUI[10] wie folgt starten:

```
# java -classpath ../lib/hsqldb.jar org.hsqldb.util.DatabaseManager
```

Für den Zugriff auf eine HSQL-Datenbank gibt es mehrere Möglichkeiten:

1. Falls dieser Database-Manager direkt aus dem Server-Dateiverzeichnis gestartet wird, in welchem unsere Datenbank (`cocoondb`) liegt (in unserem Fall in `/opt/jakarta/tomcat/webapps/conference/WEB-INF/db`), wählt man im Dialog zur Datenbank-Verbindung die Einträge gemäss Abbildung 4.2. Gewählt wird neben der Verbindungsart der entsprechende JDBC-Treiber (natürlich können via JDBC auch andere Datenbanken verbunden werden) sowie die URL der Datenbankverbindung, an dessen Ende der Datenbankname steht. Darauf folgen Benutzername und Passwort für den Datenbankzugriff.

2. Es kann auch vom Database-Manager aus eine Verbindung zu einem laufenden HSQLDB-Server gemacht werden. Dieser Server muss zusätzlich zum Database-Manager gestartet werden und ist für den Betrieb von Verbindungen mit Cocoon nicht vonnöten. Der Database-Manager kann auch auf einem anderen Computer ausgeführt werden. Bei einer Verbindung ist der Hostname oder die IP-Adresse des Servers anzugeben. In diesem Fall wird der Datenbankserver direkt im Dateiverzeichnis unserer Datenbank gestartet, analog zum obigen Beispiel in 4.2:

   ```
   # java -classpath ../lib/hsqldb.jar
       org.hsqldb.util.Server -database cocoondb
   ```

   Es muss nicht darauf geachtet werden, aus welchem Verzeichnis der Database-Manager gestartet wird, da er in diesem Fall keine Datenbank im Filesystem suchen muss.

Der Database-Manager kann die Werte in den Datenbanktabellen zwischen beliebigen Datenbanken über JDBC transferieren – die Resultate sind natürlich vom Funktionsumfang der Datenbanken abhängig. Dies kann bei einem nachträglichen Wechsel der Datenbank sinnvoll sein. Beim Start des Database-Managers müssen die entsprechenden Java-Klassen, in welchen die JDBC-Treiber vorhanden sind, mit angegeben werden:

[8] http://hsqldb.sourceforge.net
[9] http://www.mysql.com
[10] Graphical User Interface, zum Beispiel MS Windows oder X-Windows mit KDE, CDE,... oder ähnliches.

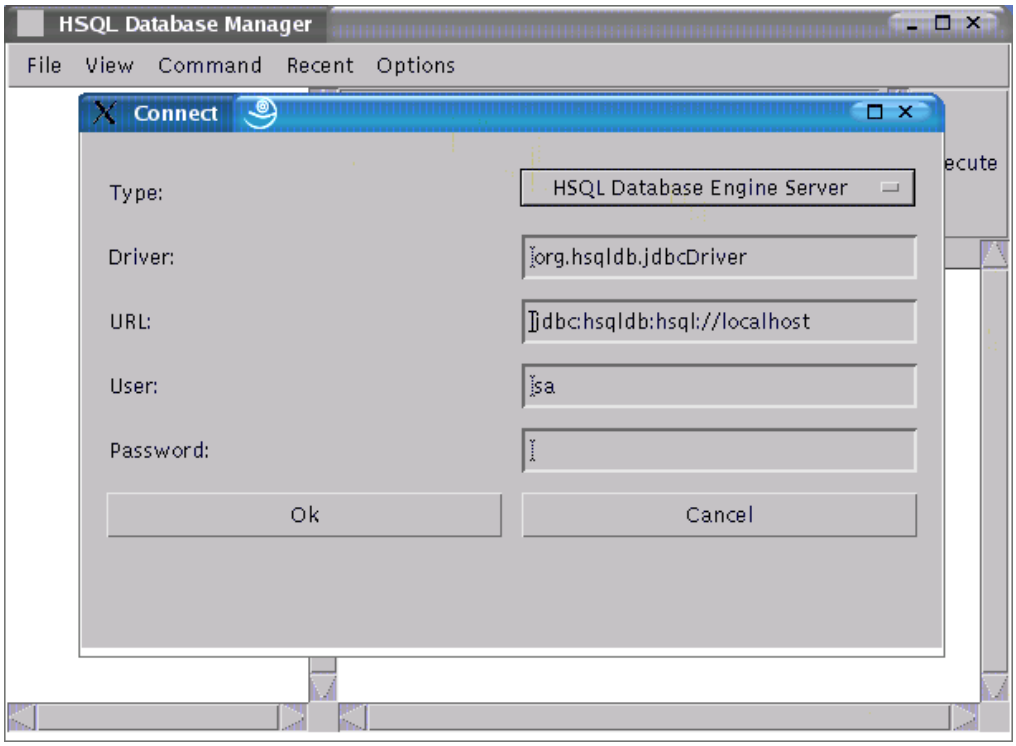


Abbildung 4.1: Verbindung zum HSQLDB-Server

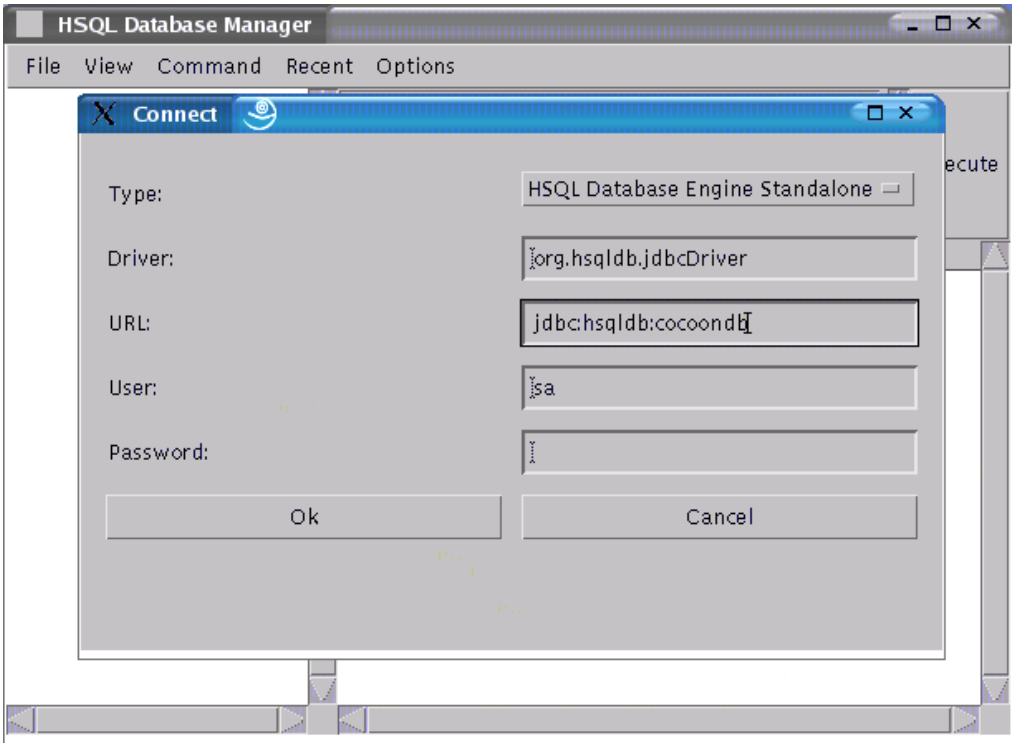


Abbildung 4.2: Standalone-Verbindung zu HSQLDB

```
# java -cp ../lib/hsqldb-1.61.jar:../lib/mysql-connector-java-2.0.14-
bin.jar.org.hsqldb.util.DatabaseManager -database cocoondb
```

Für detailliertere Dokumentationen verweisen wir auf die entsprechenden HTML-Dateien im `doc`-Verzeichnis des mitgelieferten Archives von hsqldb.

Importieren der Konferenz-Datenbank: Der Import der Konferenzdatenbank ist simpel: In `/opt/jakarta/tomcat/webapps/conference/WEB-INF/db` ist von uns eine Datei `cocoondb.script` mitgeliefert. Beim Start der `cocoondb`-Datenbank wird dieses File mit eingelesen. Darin sind alle SQL-Befehle enthalten, welche eine Datenbank initialisiern und diese dadurch lauffähig machen. Falls die HSQLDB bereits mit `cocoondb` als Datenbank gestartet wurde und Tabellen gelöscht wurden, gibt es noch die Möglichkeit, das HSQLDB-Konferenzschema in der Datei `daconf.sql` in den Database-Manager einzulesen und auszuführen.

Anbindung an Cocoon

web.xml Es muss darauf geachtet werden, dass in der Datei `$TOMCAT_HOME/webapps/conference/WEB-INF/web.xml` der JDBC-Treiber von HSQLDB, das heisst die entsprechende Java-Klasse, im Kontext des `init-param` Elementes für das `param-name`-Element der "load-class" angegeben ist:

```
<init-param>
<param-name>load-class</param-name>
    <param-value>
        <!-- HSQLDB driver-->
        org.hsqldb.jdbcDriver
        <!-- her stehen andere Datenbanktreiber -->
    </param-value>
</init-param>
```

In dem von uns mitgelieferten Archiv ist dies bereits ausgeklammert vorhanden.

coocon.xconf Der abschliessende Schritt beim Erstellen der Datenbankverbindung ist die Konfiguration von Cocoon. Dies erfolgt durch modifizieren der Datei (`$TOMCAT_HOME/webapps/conference/WEB-INF/cocoon.xconf`). Dies geschieht durch folgende Verbindungsangaben im `datasources`-Element:

```
<datasources>
    <jdbc name="daconf">
        <pool-controller min="5" max="10"/>
        <auto-commit>true</auto-commit>
        <dburl>jdbc:mysql://localhost:3306/daconf</dburl>
        <user>da</user>
        <password>erik8</password>
    </jdbc>
</datasources>
```

Falls bereits ein XML-Element `jdbc` mit dem gleichen Attribut `name="daconf"` besteht (zum Beispiel aufgrund einer anderen Datenbankverbindung), muss entweder der Name der anderen Datenbankverbindung umbenennt werden, oder das ganze Element mit allen Kinderelementen gelöscht werden. Dies kann bei uns der Fall sein, da wir – um einen einwandfreien Betieb zu ermöglichen – standardmässig das wesentlich leistungsfähige MySQL als Datenbank unterstützen.

Ferner muss noch der HSQLDB-Server gestartet werden. Dies erfolgt mit dem Start von Cocoon automatisch, falls folgende Zeilen in `cocoon.xconf` angegeben werden.

```
<hsqldb-server class="org.apache.cocoon.components.hsqldb.ServerImpl"
 logger="core.hsqldb-server" pool-max="1" pool-min="1">
   <parameter name="port" value="9002"/>
   <parameter name="silent" value="true"/>
   <parameter name="trace" value="false"/>
</hsqldb-server>
```

Der JDBC-Name “daconf” muss mit demjenigen der SQL-Transformers in den Pipelines von Cocoon übereinstimmen. In unserer Arbeit wurde die Datenbankverbindungen stets so benannt. Die HSQLDB befindet sich gezwungenermassen auf dem selben Host wie Cocoon. Nun kann Tomcat (und somit auch Coocoon und HSQLDB) gemäss dem weiter oben erwähnten Verfahren neu gestartet werden.

4.1.5.2 MySQL

MySQL steht für nicht-kommerzielle Zwecke unter der GPL[11] und kann von der MySQL-Homepage[12] heruntergeladen werden. Wie schon weiter oben bemerkt, ist es von Vorteil, die MySQL-Datenbank aus der Unix-Distribution zu verwenden statt die Soruce Codes zu kompilieren.

Nachdem die Installationsschritte gemäss der Anleitung von MySQL erfolgt sind, muss die Datenbank aus unserer Beispielskonferenz importiert werden:

```
# mysql -uroot << daconf.sql
```

Die Datei `daconf.sql` ist in unserem Archiv enthalten.

MySQL Administration: Natürlich müssen die Daten in einer Datenbank von einem Administrator auch manipuliert werden können. Wir empfehlen eine der beiden folgenden Möglichkeiten, um MySQL zu administrieren:

1. Verwendung von “MySQLCC”[13]. MySQLCC ist ein GUI für die Datenbank und läuft unter verschiedenen Plattformen. Wir verweisen für die Installation auf die Anleitung auf der Website.

[11] General Public Licence.
[12] http://www.mysql.com
[13] http://www.mysql.com/products/mysqlcc/index.html

2. Verwendung von "phpMyAdmin"[14]. phpMyAdmin ist eine PHP-Scriptsammlung (einem gelungen HTML-Webinterface). Voraussetzung zum Betrieb ist deshalb ein Webserver (zum Beispiel Apache), welcher PHP unterstützt. Falls diese Grundsätze gegeben sind, hat man durch die Installation von phpMyAdmin die Möglichkeit die Datenbank plattformunabhänig über den Webbrowser zu verwalten.

Ferner gibt es noch den Shell-Befehl `mysql`, welcher mit MySQL mitgeliefert wird und die Eingabe von SQL-Syntax erlaubt. Dieser Befehl erfordert jedoch SQL-Kenntnisse und ist umständlich zu bedienen.

MySQL-Verbindung zu Cocoon: Nun muss nur die Schnittstelle zwischen Cocoon und MySQL erstellt werden. Dies erfolgt durch die Modifikation von zwei Konfigurationsfiles von Cocoon:

web.xml Es muss darauf geachtet werden, dass in der Datei `$TOMCAT_HOME/webapps/conference/WEB-INF/web.xml` der JDBC-Treiber von MySQL, das heisst die entsprechende Java-Klasse, im Kontext des `init-param`-Elements für das `param-name`-Element der "load-class" angegeben ist:

```
<init-param>
<param-name>load-class</param-name>
    <param-value>
        <!-- MySQL driver-->
        org.gjt.mm.mysql.Driver
        <!-- her stehen andere Datenbanktreiber -->
    </param-value>
</init-param>
```

In dem von uns mitgelieferten Archiv ist dies bereits vorinstalliert. Ebenso sollte der JDBC-Driver Mysql Connector/J[15], welcher die JDBC-Requests in ein Protokoll verwandelt und dadurch eine Kommunikation mit MySQL ermöglicht, bereits im richtigen Verzeichnis vorhanden (`$TOMCAT_HOME/webapps/conference/WEB-INF/lib/mysql-connector-java-2.0.14-bin.jar`) sein.

Ferner muss noch die obige Java-Klasse von Cocoon erkannt werden (das heisst im Java-Classpath vorhanden sein). Dies erledigt sich am einfachsten durch einen weiteren Eintrag in der oben erwähnten Datei `web.xml`:

```
<init-param>
<param-name>extra-classpath</param-name>
    <!-- hier können evenuell schon Einträge stehen -->
    <param-value>
      WEB-INF/lib/mysql-connector-java-2.0.14-bin.jar
```

[14] http://www.phpwizard.net/projects/phpMyAdmin/
[15] http://www.mysql.com/products/connector-j/index.html

```
    </param-value>
</init-param>
```

Somit ist die Grundkonfiguration der JDBC-Datenbankverbindung abgeschlossen. Für andere Datenbanken erfolgt dies analog, man muss jedoch den entsprechenden JDBC-Treiber der verwendeten relationalen Datenbank installieren.

coocon.xconf Der abschliessende Schritt im Erstellen der Datenbankverbindung ist die Konfiguration von Cocoon. Dies erfolgt durch Modifizieren der Datei (`$TOMCAT_HOME/webapps/conference/WEB-INF/cocoon.xconf`). Dies geschieht durch folgende Verbindungsangaben im `datasources`-Element:

```
<datasources>
    <jdbc name="daconf">
        <pool-controller min="5" max="10"/>
        <auto-commit>true</auto-commit>
        <dburl>jdbc:mysql://localhost:3306/daconf</dburl>
        <user>da</user>
        <password>erik8</password>
    </jdbc>
</datasources>
```

Falls bereits ein XML-Element `<jdbc name="daconf"/>` mit dem gleichen Attributwert besteht, muss entweder der Name umbenannt werden oder das ganze Element mit allen Kinderelementen gelöscht werden. Dies kann bei uns der Fall sein, da wir, um eine einfachere Installation zu ermöglichen, auch die wesentlich weniger leistungsfähige HSQLDB als Datenbank unterstützen.

Der JDBC-Name “daconf” muss mit demjenigen der SQL-Transformers in den Pipelines von Cocoon übereinstimmen. In unserer Arbeit haben wir die Datenbankverbindungen stets so benannt. Wir sind davon ausgegangen, dass die MySQL-Datenbank sich auf dem selben Host befindet wie Cocoon. Falls dies nicht der Fall ist, muss anstelle von `localhost` die URI des entsprechenden Hosts angegeben werden. `3306` ist die Portnummer, an welcher die MySQL-Datenbank Verbindungen entgegennimmt und der darauffolgende String, `daconf`, bezeichnet den Namen der Datenbank in MySQL. Nun kann Tomcat (und somit auch Coocoon) gemäss dem weiter oben erwähnten Verfahren neu gestartet werden.

4.1.6 Email-/SMS-Gateway

Wie in Abschnitt 1.3 beschrieben, haben wir in unserem Projekt die Anbindung an einen beliebigen EMail- und SMS-Provider bewusst offen gelassen. Als Beispiel verwenden wir den UNIX-Command `mail` zum Versenden von Email und ein XML-RPC-Interface um Versenden von SMS. Hierzu sei noch erwähnt, dass beim entsprechenden SMS-Provider (in unserem Fall `http://www.aspsms.com` ein XML-Account beantragt werden muss, und die entsprechenden Zugangsdaten in `user_notification.sh` gemäss den Angaben des Providers eingetragen werden müssen.

Der Listener ist nicht Teil unserer Implementation in Cocoon, sondern ein eigener, separater Teil. Die Gründe dazu liegen auf der Hand: Einerseits wurde die bewusst offene Architekur gewährleistet: Falls eine neue Java-Klasse mit eintsprechenden Funktionen implementiert würde, ist bereits wieder eine Abhängigkeit aufgrund des Interfaces nach "aussen", das heisst an eine Anbindung an einen Provider, gegeben. Ferner kommt noch dazu, dass die Implementation einer weiteren Java-Klasse den Rahmen dieser Arbeit sprengen würde. Ein möglicher Ansatz für eine Folgearbeit wäre die in Abschnitt 1.3 erwähnte Sendmail-Klasse, welche in der Pipeline integriert werden könnte und Emails sowie SMS (bis zum SMS-Provider ebenfalls als Email) versandt.

Die Installation des Listeners ist einfach: Wir verwendeten das Listener-Sciptfile `user_notification.sh`, welches periodisch prüft, ob ein neues XML-File mit bestimmten Dateinamen im Verzeichnis `/tmp` abgelegt wurden. Falls dies der Fall ist, werden die XML-Files eingelesen und versandt. Dazu muss im Scriptfile der Dateipfad des Intrepreters und des für den Versand aufzurufenden Python-Scriptfile (`pscript.py`) angepasst werden: In unserem Beispiel haben wir die Dateien in `/home/da/smsscript` abgelegt. Falls diese wo anders liegen, muss der Dateipfad beim Aufruf von `pscript.py`, dem File, welches die SMS versendet, auf den entsprechenden Pfad angepasst werden. Zuletzt muss noch die `user_notification` gestartet werden:

```
# ./user_notification.sh
```

4.2 Konfiguration

4.2.1 Konferenz installieren

Um die Website einer Konferenz mit Cocoon in Tomcat lauffähig zu machen, muss unser `.war`-Archiv in das Verzeichnis `webapps` von Tomcat kopiert werden. Nach einem Restart von Tomcat wird diese Datei ausgepackt und unsere Konferenz ist mit einem Beispiellayout unter `http://localhost:8080/conference` online.

4.2.2 Aufsetzen des Layouts

Das HTML-Layout der gesamten Beispielkonferenz ist in einem einzigen XSLT-File abgelegt. Dies ist zu finden unter `$TOMCAT_HOME/webapps/conference/conf/header2.xsl`. Hier zur Illustration die Grundstruktur des XSLT-File:

```
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template name="make_page">
    <xsl:param name="page_title" select="''"/>
    <xsl:param name="page_content" select="''"/>
    <!-- hier beginnt der HTML-Code, welcher
         auf jeder Webseite dargestellt wird -->
    <html>
        <head>
```

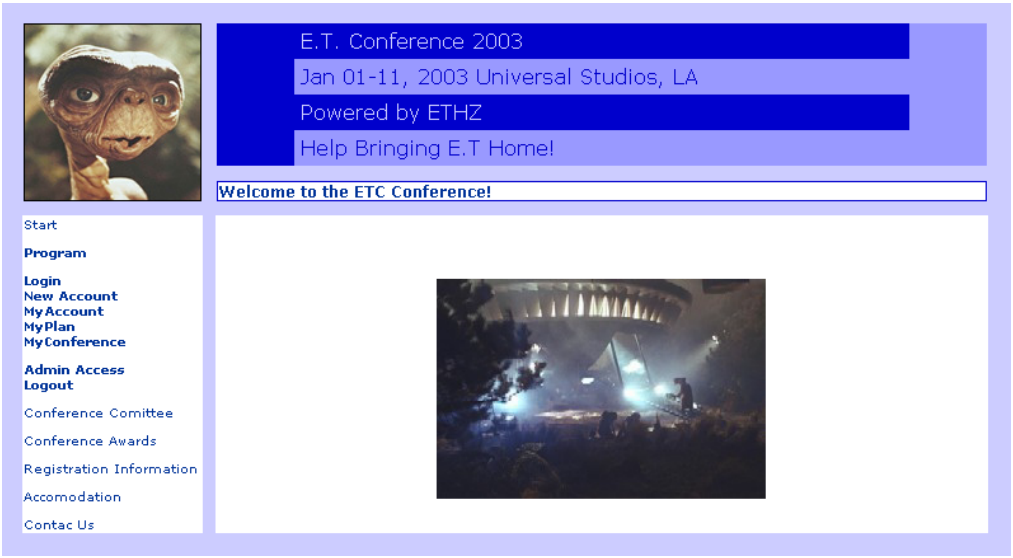


Abbildung 4.3: Layout der Beispielkonferenz

```
            <title>
                <!-- Hier wird der Titel für die
                Fensterzeile des Webbrowsers dargestellt: -->
                <xsl:copy-of select="$page_title"/>
            </title>
            </head>
            <body>
            <!-- hier beginnt der Inhalt des HTML-Layouts -->
            <!-- Irgendwo wird zum Beispiel ein Titel der Konferenz stehen,
                sowie eine Menuliste für die Auswahl der
                einezelnen Funktionen -->
            <!-- Hier wird der Inhalt aktuellen Seite dargestellt -->
                <xsl:copy-of select="$page_content"/>
            <!-- Noch einige Abschliessende Tags -->
            </body>
        </html>
    </xsl:template>
```

Für das Webdesign lässt sich auch, wie es sehr beliebt ist, ein `.css`-File[16] verwenden. Dies war, wie man dem `header2.xsl` entnehmen kann, in unserem Beispielslayout der Fall.

Es kann ein beliebiges Layout entworfen werden und der HTML-Code in dieses XSLT-File hineinkopiert werden. Der HTML-Code, welcher auf jeder Webseite ändert, ist in unserem Fall in einer Zelle einer HTML-Tabelle abgelegt und wird mit dem XSLT-Befehl `<xsl:copy-of select="$page_content"/>` abgerufen. Diese Zeile muss zwingend im XSLT-File drin bleiben!

[16] Cascading Stile Sheet, eine Datei, welche genauere Vorgaben zum HTML-Layout enthält.

Konferenzlogo, Titel	
Menu	page_content

Tabelle 4.1: Schematischer Layout-Aufbau der Beispielskonferenz

Das Beispiel des `header2.xsl` zeigt unser Beispiel-Konferenzlayout nach dem folgenden Schema (siehe Tabelle 4.1). Es wurden keine Frames, sondern lediglich Tabellenzellen verwendet, um so die Darstellung auf den verschiedenen Browser möglichst einfach zu halten.

Natürlich kann `header2.xsl` so verändert werden, dass nur HTML-Fragmente ausgegeben werden. Dies ist zum Beispiel dann sinnvoll, wenn unser Konferenztool als ein Dienst angeboten wird, welcher HTML-Fragmente anbietet, und ein separater Webserver mit diesem Fragment einen Include auf unsere URLs[17] in seine Webseiten integriert. Auf diesen Punkt wird jedoch nicht weiter eingegangen.

Die Datei `header2.xsl` liesse sich auch für eine Ausgabe auf andere Medien, wie zum Beispiel auf WML, konzipieren. Hier sei jedoch angemerkt, dass jede unserer Pipelines die HTML-Tags für die Ausgabe in der `page_content` generiert. Hierzu müssten noch die jeweils letzten Transformationen in jeder Pipeline angepasst werden[18].

Der Webdesigner möchte unter Umständen auch das Layout der Authentisierungsformulare ändern. Dies ist ganz einfach möglich, auch wenn die Textboxen inmitten der von der Konferenz vorgegebenen HTML-Ausgaben liegen. Das Layout der Login-Meldungen und -Formulare befindet sich in der Datei `login_forms.xsl`. Die Veränderungen der Layoutangaben erfolgen analog denjenigen in `header2.xsl`, wobei hier nicht ganze HTML-Seiten, sondern nur wohlgeformte XHTML-Fragemente erzeugt werden, welche in der Pipeline mit der ganzen HTML-Seite zu HTML serialisiert werden.

4.2.3 Datenbank

Im Abschnitt 4.1.5 haben wir die Installation einer geeigneten Datenbank und den Import unser Beispiels-Konferenzdaten erläutert. Wir gehen davon aus, dass ein lauffähiges Datenbank-Beispiel unserer Konferenz vorhanden ist und der Administrator mit der Verwendung seines Datenbankadministrations-Werkzeugs vertraut ist.

Als nächster Schritt muss die genauere Definition der Event-Typen bestimmt werden: Die Event-Typen sind in unserer Datenbank in der Tabelle `eventType` gespeichert. In unserer Datenbank sind "per default" fünf Eventtypen vorgegeben, wie aus Tabelle 4.2 entnommen werden kann. Jeder Event-Typ hat Eigenschaften (Inhalt, Beginn-Datum und -Zeit, Ende-Datum und -Zeit, Raum, Person, Weblink und (zugelassene) Tickets), ihre Verwendung ist fakultativ. Es können nach Belieben weitere Typen hinzugefügt oder Bestehende verändert oder gelöscht werden.

Nachdem die Event-Typen angepasst wurden, können die einzelnen Events auf `http://localhost:8080/conference/myadmin.html` eingetragen werden (siehe Abb. 4.5).

[17] Eine Web-Adresse, wie um Beispiel `http://dret.net`, welche unsere Fragemente "abruft" und der URL entsprechend zurückgegeben werden.

[18] Eine mögliche Erweiterung wäre beispielsweise die letzte Transformation browserspezifisch mit den entprechenden XSLT-Files ausführen zu lassen. Dies erfolgt durch mit den `<map:when browser="..."/>`-Tag.

eventType_id	1	2	3	4	5
name	Track	Talk	Speech	Tutorial	Workshop
has_content	yes	no	yes	no	no
has_sdate	yes	yes	no	yes	yes
has_edate	yes	yes	no	yes	yes
has_stime	no	yes	no	yes	yes
has_etime	no	yes	no	yes	yes
has_room	no	yes	no	yes	yes
has_person	no	yes	yes	no	no
has_link	yes	yes	yes	yes	yes
has_ticket	no	yes	no	yes	yes

Tabelle 4.2: Vorgegebene Event-Typen

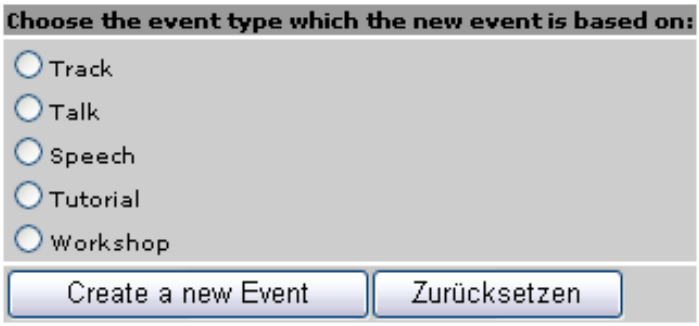


Abbildung 4.4: Auswählen des Event-Typs

Hierzu muss sich der Administrator auch als solcher einloggen und unter “Admin Access” des Punkt “add event” auswählen. Für den vorgegebenen Administrator-Login lautet der Benutzername `admin` und das Passwort ebenfalls `admin`. Vor Freigabe des ganzen Projektes in den produktiven Betrieb sollte mindestens das Passwort des Admins in der Datenbank aus Sicherheitsgründen in der Datenbank-Tabelle `user` geändert werden. Um ein Event hinzuzufügen, muss erst der Eventtyp gewählt werden. Nach Bestätigung der Auswahl erfolgt ein Formular mit Eventtyp-spezifischen Angaben. Nach dem Bestätigen der Formulareingabe erfolgt die Aufschaltung des Events. Zum Ändern oder Löschen eines Events kann analog vorgegeangen werden. Events können jedoch nur gelöscht werden, wenn diese keine Kinder mehr enthalten. Die Abbildungen 4.4, 4.5 und 4.6 verdeutlichen den Vorgang.

Es ist noch anzumerken, dass, wenn Änderung von Event-Typen durchgeführt werden, nachdem bereits Events dieses Types eingetragen wurden, die Konsistenz der Tabellen in der Datenbank vom Administrator zu überprüfen ist. Falls Event-Typen hinzugefügt werden, beeinflusst dies bestehende Events nicht. Bei Änderung eines Typs, zu welchem bereits Events bestehen, muss damit gerechnet werden, dass einzelne Einträge nicht mehr angezeigt werden. Dies passiert, sobald ein `has_*`-Eintrag von `yes` auf `no` gesetzt wird. Somit geht Information verloren. Im umgekehrten Fall, das heisst wenn `has_*`-Eintrag von `no` auf `yes` gesetzt wird, muss damit gerechnet werden, dass die Events dieses Typs keine Ausgabe bei Abfrage dieses Eintrages machen. Darum empfiehlt es

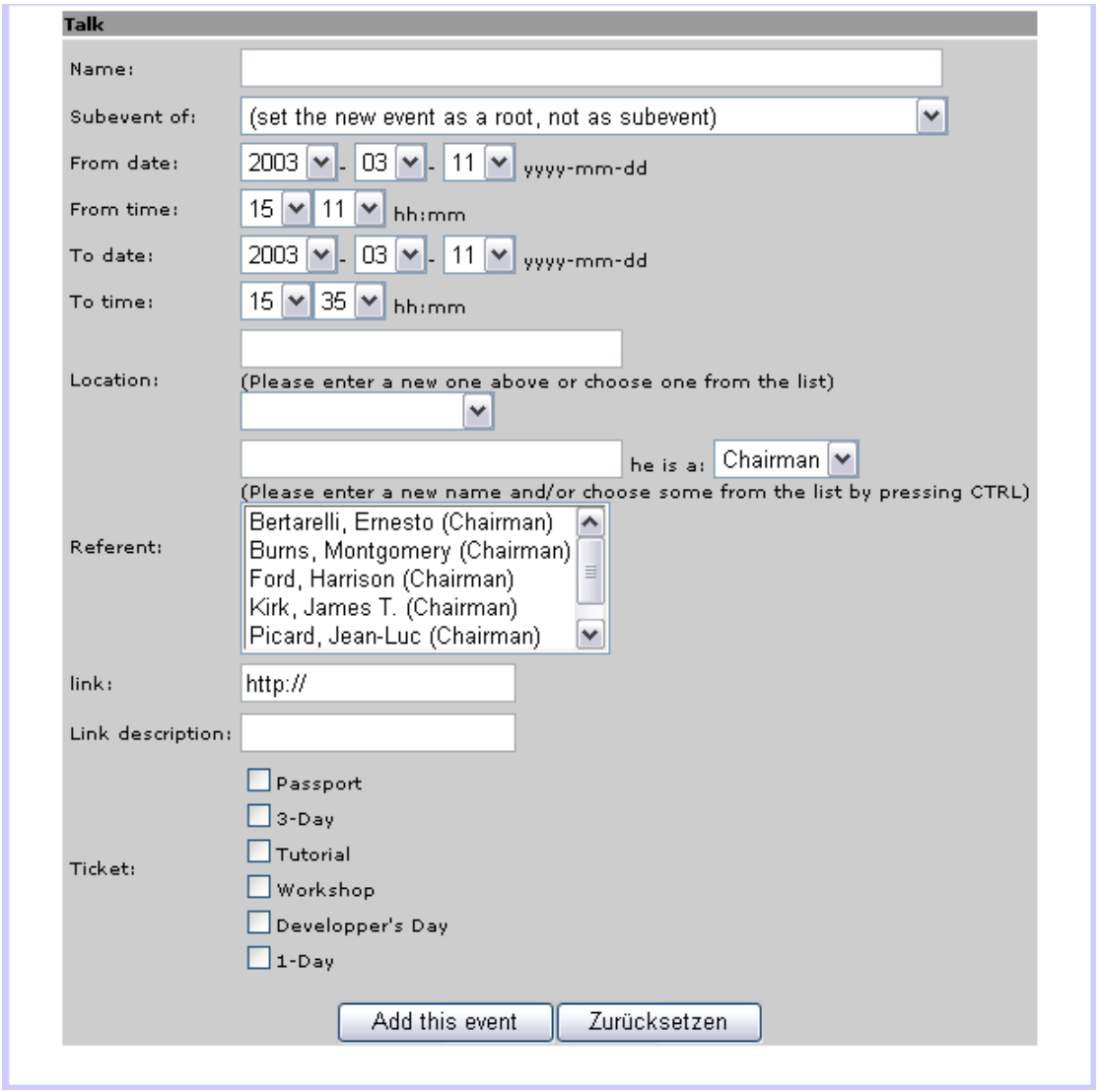


Abbildung 4.5: Hinzufügen eines Events

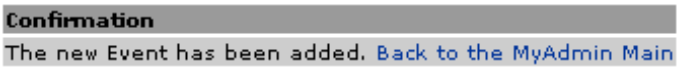


Abbildung 4.6: Bestätigungsmeldung

sich, beim Ändern von Event-Typen die dazugehörigen Events in der Tabelle `event` von Hand durchzuprüfen und gegebenfalls anzupassen.

Ferner ist aufgrund des Datenbank-Schemas nicht vorgegeben, ob, aufgrund des Event-Typs, ein Event eines gewissen Typs Kind eines Events eines anderen Types sein darf oder nicht. Diese Konsistenz muss der Administrator selber überprüfen.

Neue Benutzer des gesamten Online-Tools können sich direkt `http://localhost:8080/conference/apply.html` anmelden. Standardmässig wird dem Benutzer das Userlevel 1 vergeben. Dieses Benutzerlevel erlaubt nur das Betrachten, Zusammenstellen und Ändern des persönlichen Konferenz-Programmes und der eigenen Angaben. Falls ein Benutzer in den Administratorstand erhoben werden soll, muss ihm der Webmaster in der Datenbank `person` ein Userlevel von 10 geben. Die dazwischen liegenden Werte sind für weitere Userlevel reserviert.

4.2.4 Email-/SMS-Gateway

Ein Benutzer kann ja bei Änderungen (zum Beispiel eine neue Startzeit oder eine Raumänderung) von Events wünschen, via SMS oder Email darüber informiert zu werden. Der Text, welcher versandt wird, kann für Email und SMS separat definiert werden. Dies passiert durch Modifikation der Variablen in der Datei `$TOMCAT_HOME/webapps/conference/conf/report_message.xsl`.

Kapitel 5

Dokumentation für Entwickler

In diesem Kapitel wird die Implementation des CTTM betrachtet. Dieses Kapitel ist besonders wichtig für eine Weiterentwicklung des CTTM, da es die Ideen und Gedankengänge zeigt, die zur aktuellen Version geführt haben. Im Abschnitt 5.1.1 wird der Aufbau der Datenbank und der dazu erstellten Tabellen beschrieben. Danach werden im Kapitel 5.2 sämtliche Funktionen ums Login und ums Session-Management erklärt. Die Administrier-Funktionen zum Erstellen, Ändern und Löschen werden in Abschnitt 5.3 genauer betrachtet. Das Erstellen, Ändern und Löschen von Benutzeraccounts folgt in Abschnitt 5.4. Zuletzt werden noch die verschiedenen Ausgabemöglichkeiten des Konferenzprogramms im Kapitel 5.5 erklärt.

5.1 Datenbank

5.1.1 Datenbank-Design

Der Grundstein für das ganze CTTM legen die verschiedenen Tabellen einer SQL-Datenbank. Abbildung 5.1 zeigt die verschiedenen verwendeten Tabellen. Für alle Tabellen gilt: Der Wert hinter dem Spaltentyp (zum Beispiel Text) ist die maxmimale Länge des Wertes.

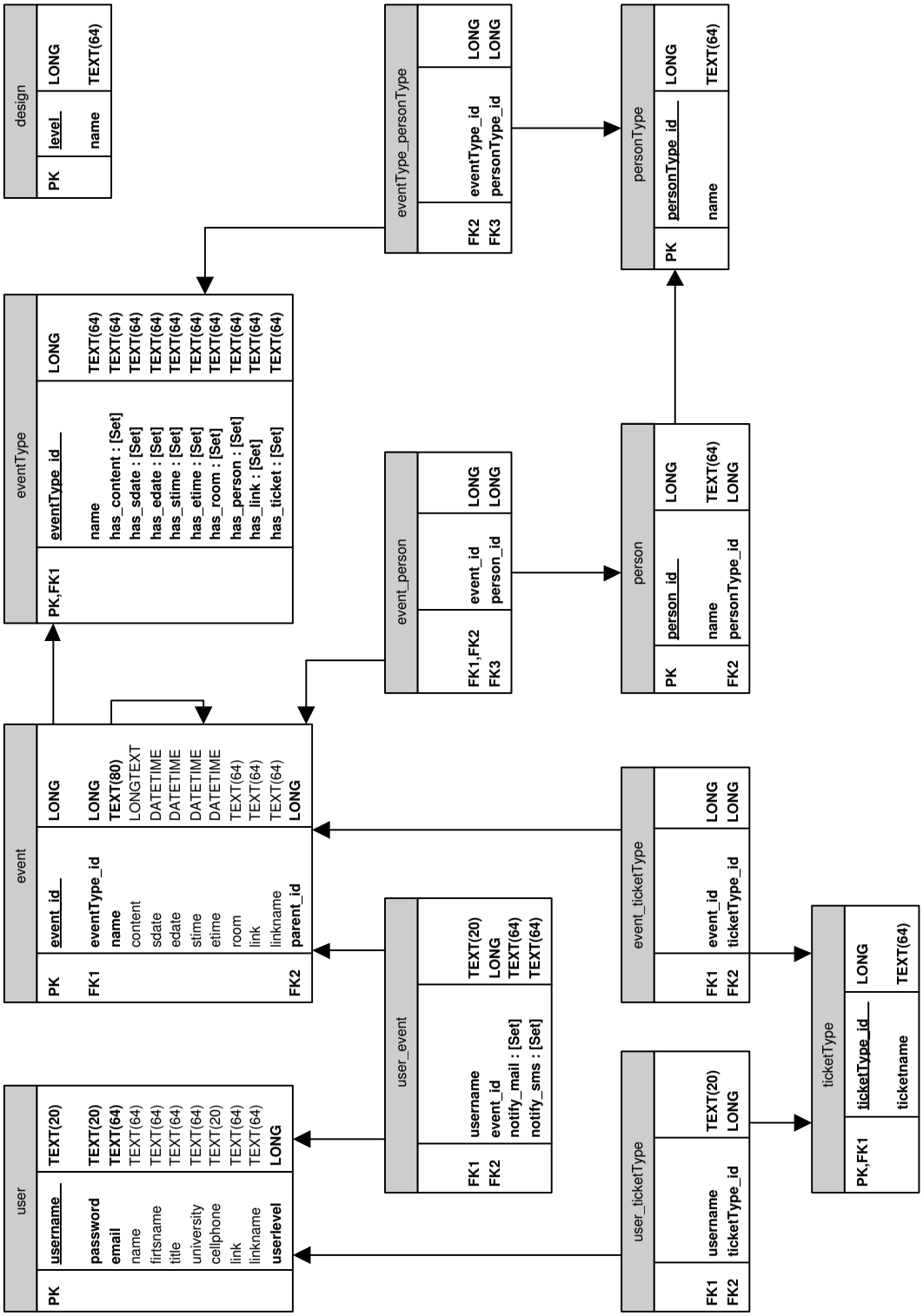


Abbildung 5.1: SQL-Datenbank-Tabellen

user: In dieser Tabelle werden die Benutzereinstellungen gespeichert (jedoch nicht der persönliche Konferenzplan). Verlangt werden die Spalten "username", "password", "email" und "userlevel", die anderen Spalten sind nicht zwingend und sollten selbsterklärend sein. Der Username ist zugleich der primäre Schlüssel und muss deshalb eindeutig sein. Userlevel sind im Moment zwei definiert: Userlevel 1 entspricht einem Benutzer, und Userlevel 10 entspricht einem Administrator. Die dazwischen liegenden Werte könnten für etwaige weitere User-Hierarchien benutzt werden. Die Validierung der eingegeben Daten geschieht mittels des Form-Validators (Kapitel 5.4.1).

eventType: Hier werden die verschiedenen EventTypes gespeichert. Alle Spalten müssen angegeben werden. Der primäre Schlüssel ist die "eventType_id", dieser ist ein "auto-value". Das heisst wenn ein neuer Typ eingefügt wird, sollte der automatisch einen Wert erhalten, der um eins höher ist, also der Wert des zuletzt eingefügten Types. Die Spalten die mit "has_" beginnen, aktzeptieren nur "yes" oder "no" als gültige Werte. Die Werte legen fest, welche Eigenschaften ein Type haben darf.

event: Der primäre Schlüssel ist die "event_id". Diese ID wird ebenfalls von SQL (Autowert) zugewiesen, um sicherzustellen, dass nicht der Administrator aus Versehen zwei Events dieselbe ID zuweist. Von SQL werden ebenfalls die Spalten "eventType_id", "name" und "parent_id" verlangt. Für ein funktionierendes CTTM müssen jedoch noch nicht in SQL vorhandene Beziehungen zwischen zwei Tabellen gelten. So muss "eventType_id" identisch sein mit einem Wert der Spalte "eventType_id" der "eventType"-Tabelle. Events die diese Bedingung nicht erfüllen, werden im CTTM ignoriert. Dasselbe gilt für "parent_id". Dieser Wert verweist auf die Spalte "event_id" derselben Tabelle und verknüpft so die Events untereinander. Die übrigen Spalten korrespondieren mit den entsprechenden "has_"-Spalten der "eventType"-Tabelle. Wenn in einer dieser Spalten der Wert "yes" steht, so gibt CTTM den korrespondierenden Wert der "event"-Tabelle aus. Steht "no", so gibt CTTM nichts aus, selbst wenn in der Datenbank ein Eintrag gemacht wurde.

user_event: In dieser Tabelle wird der persönliche Konferenzplan der Benutzer gespeichert. Dabei wird für jeden selektierten Event eine eigene Zeile benötigt. "Username" und "event_id" entsprechen Werte der Tabelle "user" beziehungsweise "events". "notify_sms" und "notify_mail" können die Werte "yes" oder "no" haben, sie geben an, ob ein Benutzer bei einer Änderung des entsprechenden Events benachrichtigt werden wollen oder nicht. Alle Spalten müssen einen Wert haben.

ticketType: "ticketType_id" ist der primäre Schlüssel und ein Autowert, "ticketname" ist der Name des Tickets. Beide Werte müssen vorhanden sein.

user_ticketType: Hier wird gespeichert, welche Tickets der Benutzer besitzt. So könnte eine Warnung implementiert werden, wenn ein Benutzer einen Event auswählt, für den er kein Ticket hat. Diese Funktion wurde leider nicht in der aktuellen Version des CTTM implementiert. "username" ist ein Wert, der dem gleichnamigen Wert der Tabelle "user" entspricht, analoges gilt für "ticketType_id".

event_ticketType: Die Tabelle verknüpft die Events mit den für einen Event gültigen TicketType. Die Bemerkungen sind analog des obigen Abschnitts. Ein Event kann mehreren TicketTypes zugeordnet werden.

personType: In dieser Tabelle werden die verschiedenen Funktionen, die eine Person an einer Konferenz haben kann, gespeichert (Zum Beispiel author, chairman, speaker,...). Der "name" ist zugleich der Typ, die "personType_id" ist ein primärer Schlüssel. Beide Werte werden verlangt.

eventType_personType: Hier wird festgelegt, welche Personen-Typen in einem Event-Typ erlaubt sind. "eventType_id" beziehungsweise "personType_id" korrespondieren mit gültigen Werten der gleichnamigen Spalten der Tabellen "eventType" und "personType". Beide Werte müssen vorhanden sein. Mit einem Event-Typ können beliebig viele Personen-Typen verknüpft werden.

person: Sämtliche Personen, die in einem Event vorkommen, werden in dieser Tabelle gespeichert. Alle Werte müssen angegeben werden. "person_id" ist der primäre Schlüssel und ein Autowert, "name" ist der komplette Name der Person. Dabei bleibt es dem Administrator überlassen, die Personen konsequent zu benennen (zum Beispiel "Nachname", "Vorname"). "personType_id" verweist auf einen gültigen Wert der gleichnamigen Spalte der Tabelle "personType". Wenn eine Person zwei Funktionen hat, so ist ihr Name zweimal in der Tabelle mit unterschiedlichen IDs und PersonTypes gespeichert!

event_person: Diese Tabelle ordnet die verschiedenen Personen den dazugehörigen Events zu. "event_id" beziehungsweise "person_id" korrespondieren mit gültigen Werten der gleichnamigen Spalten der Tabellen "event" und "person". Beide Werte werden verlangt. Zu einem Event können beliebig viele Personen assoziiert werden. Nicht von SQL, aber von CTTM überprüft wird zusätzlich folgende Bedingung: Einem Event können nur Personen hinzugefügt werden, deren Personen-Typ im dem Event zugehörigen EventTyp erlaubt ist.

design: Diese Tabelle ist unabhängig von den anderen. Sie hat einzig und alleine die Aufgabe, eine Standard-Ansicht für das Konferenzprogramm zu definieren. Die Spalte "level" ist der primäre Schlüssel und entspricht der Tiefe des Konferenzprogramms. Die erste Ansicht des Programms entspricht dem Wert "1", wenn dort auf einen Event geklickt wird, führt dies den Benutzer in Level "2" etc. Im Moment sind zwei gültige Werte für die Spalte "name" möglich: "compact", dies hat die kompakte Ansicht der Darstellung zur Folge, und "detail", dies hat die detaillierte Ansicht zur Folge. Es könnten weitere Ansichtsarten implementiert werden, diese könnten dann hier auch als Standardansicht für einen Level eingetragen werden.

5.1.2 Datenbank-Abfragen

Die Datenbank kann auf zwei verschiedene Arten abgefragt werden. Entweder geschieht dies in einem XSLT-Dokument oder in einem XSP-Dokument.

5.1.2.1 Abfrage in XSLT

Damit in Cocoon in einem XSLT-Dokument eine SQL-Abfrage gemacht werden kann, muss XSLT erweitert werden. Dies geschieht durch Elemente die durch den Namensraum `xmlns:sql="http://apache.org/cocoon/SQL/2.0` vorgegeben sind.

```
<!-- beliebige XSLT-Elemente-->
<sql:execute-query>
   <sql:use-connection>daconf</sql:use-connection>
   <sql:query>
      SELECT person.name, personType.name AS Type
      FROM person, personType
      WHERE person.personType_id=personType.personType_id
   </sql:query>
</sql:execute-query>
<!-- beliebige XSLT-Elemente-->
```

Das Beispiel zeigt den grundlegenden Ablauf einer SQL-Abfrage in XSL, dieser geschieht im Element `sql:execute-query`. Zuerst muss im Element `sql:use-connection` angegeben werden, welche Verbindung benutzt wird. Der angegebene Wert muss mit einer definierten Datenbankverbindung von `cocoon.xconf` übereinstimmen (vergleiche auch mit Abschnitt 4.1.5). Im Element sql:query wird dann das SQL-Statement gemäss der für die Datenbank spezifischen Syntax ausgeführt.

Die Ausgabe kann dann zum Beispiel wie folgt aussehen:

```
<sql:rowset>
   <sql:row>
      <sql:name>Hans Muster</sql:name>
      <sql:type>Speaker</sql:type>
   </sql:row>
   <sql:row>
      <sql:name>Martha Muster</sql:name>
      <sql:type>Chairman</sql:type>
   </sql:row>
<sql:rowset>
```

Zu beachten ist, dass sämtliche Elemente dem SQL-Namespace zugeordnet werden. Die Lösung der Abfrage wird in das Element `sql:rowset` und die einzelnen zurückgegeben Zeilen in `sql:row` geschrieben. Diese Ausgabe muss dann im Normalfall in einem weiteren Schritt in der Pipeline weiterbearbeitet werden.

Damit in der Pipeline auch tatsächlich die SQL-Tranformation ausgeführt wird, muss folgendes stehen:

```
<map:transform type="sql">
   <map:parameter name="use-connection" value="daconf"/>
</map:transform>
```

Auch hier muss "value" ein Wert zugewiesen werden, der in `xconf` definiert wurde.

Cocoon verarbeitet dann das kombinierte Dokument (mit XSLT-Befehlen und SQL-Abfragen wie folgt: Zuerst werden SQL-Statement gesucht und ausgeführt, diese werden dann ins XSLT-Dokument "geschrieben". Erst danach wird das Dokument vom XSLT-Prozessor verarbeitet.

5.1.2.2 Abfrage in XSP

Auch für die SQL-Abfrage muss in XSP der entsprechende Namensraum angegeben werden. `xmlns:esql="http://apache.org/cocoon/SQL/v2`. Dieser muss zusätzlich zum XSP-Namensraum angegeben werden.

```
<esql:connection>
   <esql:pool>daconf</esql:pool>
   <esql:execute-query>
      <esql:query>
         SELECT person.name, personType.name AS Type
         FROM person, personType
         WHERE person.personType_id=personType.personType_id
      </esql:query>
      <esql:results>
         <esql:row-results>
            <xsp:logic>
               sql_name = <esql:get-string column="name"/>
               sql_name = <esql:get-string column="type"/>
            </xsp:logic>
         </esql:row-results>
      </esql:results>
   </esql:execute-query>
</esql:connection>
```

Der erste Teil ist, abgesehen von anderen Namen der Elemente, identisch mit der Abfrage in einem XSLT-File. Der grosse Unterschied ist jedoch, dass die Resultate der Abfrage im selben Dokument durch das Element `esql:results` verfügbar sind (im Gegensatz zur SQL-Abfrage in XSLT, wo die Resultate erst durch den SQL-Transformator in der Pipeline vorhanden sind). So können diese zum Beispiel in Variablen gespeichert und direkt weiter verwendet werden. Dies ist um einiges übersichtlicher als die SQL-Abfrage in XSLT, da bei dieser mit zwei Files gearbeitet werden muss.

5.2 Login, Logout und Session-Management

Gewisse Webseiten sind nur für registrierte Benutzer erreichbar. Dies ist aus zwei Gründen notwendig: Einerseits um eine Sicherheit über die persönlichen Angaben garan-

tieren zu können und andererseits um benutzerspezifische Einstellungen, zum Beispiel von Events, wiedergeben zu können.

5.2.1 Session-Management

Wir hatten den Login in dem in Kapitel 1.2.4.7 beschriebenen XSP implementiert. Dies begründet sich dadurch, dass – um den Login eine Sitzung lang bestehen lassen zu können – das Java-Session-Management (`javax.servlet.http.HttpSession`) verwendet wird, und dass die für die Authentisierung nötige Datenbank-Query gleich kompakt im selben File verarbeitet werden kann. Ein Session-Management ohne XSP wird von Cocoon nicht unterstützt. Mittels eines Session-Managements können so serverseitig Variablen, das heisst Zustände und Information, über die Sitzung des aktuellen Clients gespeichert werden.

5.2.2 Pipeline

Wie in Abschnitt 1.2.4.7 erwähnt, wird in einer Pipeline aus dem XSP ein dynamisches XML-File (oder SAX-Events) bei Abruf der URI, die auf die Pipeline matcht, generiert. Daher muss das Login-XSP-File, `login.xsp`, als Generator in jeder Pipeline sein, in welcher eine Benutzer-Authentisierung erfolgen soll. Hier folgt ein Blick in die Pipeline, welche bei URI-Requests endend auf `my*.html` gestartet wird und dadurch eine Benutzer-Authentisierung erfordern.

```
<map:match pattern="my*.html">
   <!-- check if login needed -->
   <map:generate src="conf/login.xsp" type="serverpages">
      <map:parameter name="realm" value="{1}"/>
   </map:generate>
   <!-- hier folgen die Transformer und der Serializer der
    entprechenden Seite -->
</map:match>
```

5.2.3 Authentisierung

Der Login ist mit einer Ausnahme unabhängig von den folgenden Transformationen in der Pipeline: Er muss für die darauffolgenden Transformationen einige bestimmte, vordefinerte Elemente in der Pipeline weitergeben. Deshalb gibt es folgende Punkte zu beachten:

5.2.3.1 perform-login

Wie aus dem `login.xsp`-Code sichtbar ist, überprüft `login.xsp`, ob der Benutzer schon eingeloggt ist (das heisst, ob eine gültige Sesson besteht). Falls der Benutzer sich nicht schon authentisiert hat und keine Session besteht, wird eine Session erstellt. Das `login.xsp` gibt als Root-Element das Element `sess` in die Pipeline weiter. Falls noch keine Session bestand, wird ein Kind-Element namens `perform-login` generiert.

Dieses erzwingt die Ausgabe des Login-Forms. Dessen Kinder sind Elemente, die für das (HTML-)Login-Formular notwendig sind:

```
<sess>
   <perform-login>
      <pipeline><xsp-request:get-uri/></pipeline>
      <idfield>id</idfield>
      <passfield>password</passfield>
      <cancelurl>/</cancelurl>
      <applyurl>apply.html</applyurl>
   </perform-login>
</sess>
```

Das Element `pipeline` beinhaltet die URI, die beispielsweise ein HTML-Formular als Action-Attribut besitzt. An dieser Stelle wird der Login weiterbearbeitet. Im Normalfall ist dies die eigene Pipeline, welche auch durch `xsp-request:get-uri` wiedergegeben wird. Die Elemente `idfield` und `passfield` bezeichnen die Namen des Benutzername- und Paswort-Feldes.

Das Element `cancelurl` bestimmt die URI, an welche gesprungen werden soll, falls ein Benutzer den Login-Vorgang abbrechen möchte und uner der URI im Element `applyurl` kann sich ein Benutzer neu anmelden. Nach der entsprechenden Transformation der Elemente `<sess><perform-login/></sess>` (und deren Kindern), welche ein XSLT-File am Ende der Pipeline vornehmen muss, sieht dieses HTML-Formular in unserem Konferenzlayout, folgendermassen aus (vereinfachte Darstellung).

```
<form method="post" action="/conference/myplan.html">
   <table border="0">
      <tr>
         <td>Your Name/Id: </td>
         <td><input name="id" type="text"></td>
      </tr>
      <tr>
         <td>Your Password: </td>
         <td><input name="password" type="password"></td>
      </tr>
   </table>
   <input value="login" type="submit">
</form> <form method="get" action="/">
   <input value="abbrechen" type="submit">
</form>
```

Das HTML-Layout zum obenstehenden Beispiel wird in in der Datei `login_forms.xsl` definiert. Ebenfalls wird die Session-Variable `logstatus` auf `logincheck` gesetzt. Diese Session-Variable informiert über den aktuellen Zustand des Logins und was die nächsten Schritte von `login.xsp` sein sollen.

5.2.3.2 no-login

Dieser Abschnitt betrifft einen nicht erfolgreichen Login. Dies ist der Fall, wenn ein falscher Benutzername oder ein falsches Passwort eingegeben wurde. Falls eine Session besteht, die Session-Variable `logstatus` auf `logincheck` gesetzt ist, und der Benutzer gemäss Abfrage der Datenbanktabelle "user" eine falsche oder keine Benutzername/Passwort-Angabe gemacht hat, wird von `login.xsp` die Elemente `<sess><no-login/></sess>` generiert. Ein XSLT am Ende der Pipeline sollte diese Elemente matchen und eine entsprechende Meldung über den misslungenen Login ausgeben. In diesem Falle wird auch die aktuelle Session invalidiert.

5.2.3.3 user

Falls eine Session besteht, die Session-Variable `logstatus` auf `logincheck` gesetzt ist und der Benutzer gemäss Abfrage der Datenbanktabelle "user" eine richtige Benutzername/Passwort-Angabe gemacht hat, wird von `login.xsp` die Elemente `<sess><user/></sess>` generiert. Weitere Transformationen in der Pipeline können auf diese Elemente matchen und so von einer zuverlässigen Authentisierung ausgehen.

Ferner ist noch zu erwähnen, dass auch der Userlevel eine Rolle spielt: Jeder Benutzer besitzt ein Userlevel, welches bei einer Neuanmeldung auf 1 gesetzt wird. Administratoren benötigen ein Userlevel von mindestens 10. Administratoren können andere Benutzer als Administratoren definieren, indem sie den Userlevel in der Datenbanktabelle `user` auf 10 erhöhen. Wie aus dem Code von `login.xsp` zu entnehmen ist, wird bei einer Authentisierung auf die Administrations-Seiten (beginnend mit `myadmin`, wobei der Parameter `realm` aus der Pipeline gelesen wird und lediglich auf `admin` matchen darf) ein Userlevel von 10 verlangt. Falls ein User nicht diesen Level besitzt, wird sein Login ungültig – analog zum Vorgang im Abschnitt 5.2.3.2.

Für Java Session-Management gibt es sowohl ein URL-Rewriting[1] als auch die Verwendung von Cookies auf der Client-Seite. Aufgrund des zeitlich engen Rahmens diese Projekt haben wir uns auf die Verwendung von Cookies beschränkt.

Ferner haben wir noch die Verwendung von HTTP-Authentisierung gemäss RFC 2617[2] in Betracht gezogen, doch eine weitere Entwicklung verworfen, da die Dokumentation über Authentisierung über ein HTML-Formular vorhanden war. Ferner bietet die HTTP-Authentisierung das Problem des zustandslosen HTTP-Protokolls, was ein Login auf jede abgerufene URL bedeuten würde. Zeitgemässe Web-Browser senden darum nach einem Login den gleichen Benutzername und dasselbe Passwort bei jedem Abruf einer authentisierungs-bedürftigen Webseite aus demselben sogenannten "Realm"[3]. Der Nachteil davon ist, dass aus diesem Grund auf Serverseite, zum Beispiel bei klicken eines "Logout"-Links durch den Benutzer, kein Logout vorgenommen werden kann. Die einzige Lösung eines Logouts wäre die, dass der Benutzer sich mit einer fehlerhaften Benutzer/Passwortkombination nochmals einloggt. Dies ist aber ein äusserst umständliches und unsauberes Vorgehen.

[1] Bei URL-Rewriting wird die die Session-ID ans Ende der URL gehängt.

[2] `ftp://ftp.isi.edu/in-notes/rfc2617.txt`

[3] Ein "Authentisierungs-Realm", welches Teil der HTTP-Authentisierung ist, wird identifiziert durch einen String. Für alle Seiten unter dem gleichen Realm gelten die gleichen Login/Benutzername-Kombination, welche der Browser bei jedem Abruf sendet.

Was dem interessierten Leser noch im unteren Drittel des Quellcodes von `login.xsp` auffallen mag, sind die `request.getParameter("...")` . Dies wird in verschiedenen Pipelines bei der Erstellung und Veränderung von Events (in den Folgepipelines von `myadmin.html`) und beim Hinzufügen von Events zu myPlan (in `myconf.html`) verwendet. Da wir in der Literatur keine Hinweise auf die Pipeline-Verarbeitung von Mehrfach-Antworten aus HTML-Formularen gefunden hatten, mussten wir auf die entsprechenden Java-Klassen zurückgreifen, welche dieses Problem lösen können. Das besondere bei Mehrfach-Auswahlen in HTML ist, dass über HTTP-POST vom Client beim Abschicken des Formulars für jede Auswahl eine Variable gesandt wird und innerhalb einer Auswahl die verschiedenen Werte mitgegeben werden, aber alle Variablen den gleichen Namen haben. Damit schien Cocoon Probleme zu haben, wir hatten in unseren Tests nur jeweils das erste Element gekriegt. Mit Hilfe von Java-String-Arrays konnten wir bei Mehrfachauswahlen jedes Element einzeln in ein XML-Element ausgeben.

5.2.4 Logout

Der Logout ist wesentlich einfacher zu implementieren: Wie im Code `logout.xsp` zu sehen ist, wird eine aktuelle Session invalidiert. Danach werden folgende Elemente in die Pipeline weitergegeben, deren Textinhalt von einem XSLT-File, wie in unserem Beispiel `login.xsl` und dem Layoutfile `login_forms.xsl`, zu HTML konvertiert werden kann:

```
<sess>
   <session-killed>
      <message>
         You're sucessfully logged out. Please visit us soon.
         Have a nice day.
      </message>
      <retrylink>mylogin.html</retrylink>
   </session-killed>
</sess>
```

In der Sitemap sehen die Schritte folgendermassen aus:

```
<map:match pattern="logout.html">
   <map:generate src="conf/logout.xsp" type="serverpages"/>
   <map:transform src="conf/logout.xsl"/>
   <map:serialize type="html"/>
</map:match>
```

5.3 Eventverwaltung

Die Administrationsseiten, welche auf die URI `myadmin*.html` matchen, werden durch die Pipeline mit mehreren Schritten und mehrfachen Transformationen verarbeitet (siehe Sitemap-Code im Ahnang B.1).

Wie der Pipeline[4] entnommen werden kann, werden mindestens zwei XSLT-Dateien für

[4]Siehe Anhang B.1.

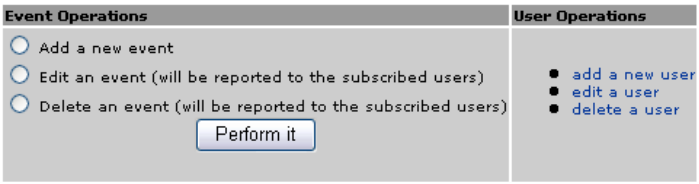


Abbildung 5.2: Administrator-Menu

einen Pipeline-Durchlauf benötigt. Dies begründet sich dadurch, dass die erste Transformation spezifische Arbeiten wie Datenbank-Abfragen durchführt (darum folgt darauf der SQL-Transformer) und die letzte XSLT-Transformation macht aus den Datenbank-Replies, welche in üblichen XML-Elementen stecken, browser-konformen HTML-Code. Wie wir obiger Pipeline ebenfalls entnehmen können, wird mit einem Element `map:when` die URI `myadmin3.html` abgefangen und speziell behandelt. Dies liegt daran, dass für diese aufgrund der Komplexität zwei SQL-Transformationen hintereinander stattfinden.

Für alle Operationen an den Events erhält der Benutzer durch Transformation mit `myadmin.xsl` eine Auswahl, ob er ein Event erstellen, bearbeiten oder löschen möchte. Diese Liste wird gemäss dem Matching in der Pipeline erstellt mit Transformation von `myadmin.xsl` und `myadmina.xsl`, Abbildung 5.2 zeigt das ausgegebene Menü.

5.3.1 Erstellen von Events

Wenn der Benutzer in `myadmin.html` das Erstellen eines Events wählt, wird er im nächsten Schritt vor die Wahl des Event-Types gestellt (`myadmin1.html`). Danach, in `myadmin2.html`, füllt er die Event-Typ-spezifischen Daten in das entsprechende Formular und erhält eine Bestätigung über die neuen Einträge, welche vollzogen wurden, in `myadmin3.html`.

5.3.1.1 Vorgehen in der Pipeline

Im Schritt nach der Wahl des Erstellens eines Events, wird in `myadmin1.html` in der Transformation durch `myadmin1.xsl` der EventType ausgelesen und mit `myadmin1a.xsl` zur Auswahl durch den Benutzer in HTML ausgegeben. Danach folgt eine Bestätigung der Auswahl und es wird die die URI `myadmin2.html` aufgerufen. Durch die Datei `myadmin2.xsl` werden die notwendigen Formulareinträge für einen neuen Event des gewählten Types ausgewählt. Ebenso werden durch dieselbe XSLT-Datei Daten ausgelesen, welche zur Auswahl stehen: Mögliche Parent-Events aus der Tabelle `event`, bereits verwendete Rooms (natürlich kann auch ein neuer Room im HTML-Formular angegeben werden), verfügbare Tickets (aus `event`) sowie mögliche Referenten. Mit `myadmin2a.xsl` werden diese Datenbank-Antworten als HTML dargestellt. Insbesondere ist in dieser Datei die Formulardarstellung aufwendig. Bei `selects` müssen alle möglichen Angaben auswählbar sein. Wie aus dem Quelltext zu entnehmen ist, wird Auflistung der Jahre, Monate, Monatstage, Stunden und Minuten rekursiv auf-

gezählt. Wie man aus obigem Pipeline-Fragment entnehmen kann, hat `myadmin3.html` zwei SQL-Transformationen hintereinenader. Das ist notwendig, da zuerst (durch die Transformation mit `myadmin3.xsl`) das neue Event in die Datenbanktabelle `event` eingetragen werden muss. Danach wird die neu vergebene ID des Events ermittelt und in einem zweiten Schritt in `myadmin3u.xsl` werden die Tabellen `event_person` und `event_ticketType`, welche Informationen zum neuen Event enthalten, um die vom Benutzer eingegebenen Werte (Ticket, Referenten) erweitert. Die Datei `myadmin3r.xsl` wird beim Erstellen eines Events nicht gebraucht, da keine Benutzer informiert werden. Im letzten Schritt (`myadmin3a.xsl`) erfolgt eine HTML-Ausgabe mit Bestätigung über die Erstellung des Events.

5.3.2 Editieren von Events

Wenn in `myadmin.html` das Editieren eines Events gewählt wird, ist im nächsten Schritt die Wahl des zu editierenden Events möglich (`myadmin1.html`). Nach Änderung der Event-Angaben in `myadmin2.html` werden die Daten in der Datenbank geupdated, dem Benutzer die Änderung in `myadmin3.html` und das Versenden der Änderungen an die angemeldeten Benutzer bestätigt.

5.3.2.1 Vorgehen in der Pipeline

Die einzelnen Schritte erfolgen analog wie in Abschnitt 5.3.1, dazu werden im Formular über die Änderung eines gewählten Events die aktuellen Daten vorher ausgelesen und als markierte oder gewählte Werte angezeigt. Hierzu ist es notwendig, dass `myadmin2.xsl` die entsprechenden ursprünglichen Werte des zu ändernden Events in den Elementen `chosen-event`, `chosen-persons` und `chosen-ticket` in die Pipeline "hineinträgt". Dadurch können in der HTML-Ausgabe des Formulars in `myadmin2a.xsl` die vorhandenen Werten als Standardwerte oder selektiert bzw. markiert mitgegeben werden. Somit muss der Benutzer nicht alle Daten des Events neu eingeben, sondern die bestehenden Daten werden bei keiner Änderung übernommen. Die darauffolgende Ausführung einer Pipeline, `myadmin3.html`, welche bei Bestätigung der Änderungen durch den Benutzer erfolgt, ist äquivalent zu demjenigen in Abschnitt 5.3.1, jedoch mit folgenden Ausnahmen: Die eingefügten Daten werden in die Tabelle `event` nicht mit einem SQL-`INSERT`-Befehl versehen, sondern mit einem `UPDATE`, da der Event schon vorhanden ist und lediglich die Daten im Datensatz geändert werden sollen. Ebenso werden alle Verknüpfungen mit dem aktuellen Event in den Tabellen `event_person` und `event_ticketType` gelöscht und in einem weiteren Schritt innerhalb der Pipeline durch Transformation mit `myadmin3u.xsl` und der SQL-Transformation neu erstellt. Dies schien uns die einfachere Möglichkeit zu sein, als die Tabellen mit einem `UPDATE` auf den aktuellen Stand zu bringen, zumal Einträge (zum Beispiel ein Referent) neu gewählt, abgewählt und sogar neu erstellt werden können. Die Neuerstellung ist notwendig, falls ein Referent hinzugefügt wird, der nicht schon in anderen Events einen Vortrag hält beziehungsweise für ein anderes Event zuständig ist.

5.3.2.2 Email/SMS

Eine Transformation in der Pipeline wurde im Abschnitt 5.3.1 nicht dokumentiert: die Benachrichtigung von Benutzern. Dies erfolgt mit einer `filewriter`-Transformation nach der Vorbereitung der Daten in den entsprechenden Namensräumen durch `myadmin3r.xsl`, wie auch schon in Abschnitt 4.1.6 erwähnt. Um diese Transformation mit korrekt geschriebenen XML-Files ablaufen lassen zu können, muss zu jedem der Medien (Email und SMS) und zu jedem eingeschriebenen Benutzer die Mobile-Nummer oder Emailadresse herausgelesen werden. Ebenso muss abgefragt werden, ob dieser Benutzer überhaupt über den Event informiert werden möchte. Diese Schritte erfolgen in der Pipeline durch eine SQL-Transformation nach der Transformation mit `myadmin3u.xsl`. Zusätzlich müssen noch die neuen Daten der Events, welche an den Benutzer gesandt werden sollen, in den zu schreibenden XML-Files vorhanden sein. Diese Daten wurden beim Absenden des HTML-Formulars schon durch die HTTP-Parameter übermittelt und sind daher in der Pipeline in `event` bereits vorhanden, was uns um zusätzliche Datenbankabfragen erleichtert.

5.3.3 Löschen von Events

Wenn in `myadmin.html` das Löschen eines Events gewählt wird, wird im zweiten Schritt (`myadmin1.html`) ermöglicht, den Event zu wählen, welches entfernt werden soll. Darauf, in `myadmin2.html`, wird der Benutzer zur Sicherheit gefragt, ob er den Event wirklich löschen möchte. Falls der Event keine Kind-Events hat, wird das Löschen des Events und alle damit verbundenen Einträge in den benachbarten Tabellen vollzogen und in `myadmin3.html` bestätigt, andernfalls wird unter derselben URI die Fehlermeldung ausgegeben, dass noch Kindelements vorhanden sind. Aufgrund der Konsistenz der Datenbank ist es nur möglich, Blatt-Events zu löschen. Knoten-Events im Event-Baum können erst gelöscht werden, wenn alle Kinder gelöscht wurden.

Wie man aus obigem Pipeline-Fragment entnehmen kann, hat `myadmin3.html` zwei SQL-Transformationen hintereinander. Das ist notwendig, da zuerst die Benutzer und die Information zur Benachrichtigung extrahiert werden müssen, danach wird den Event gelöscht.

5.3.3.1 Vorgehen in den Pipeline

Wir verweisen für die beiden ersten Schritte der Transformationen in der Pipeline auf die bedien vorangehenden Kapitel 5.3.1 und 5.3.2. Im dritten Schritt, also in `myadmin2.html`, wird eine Bestätigung eingeholt, ob dieser Event wirklich gelöscht werden soll. In der nächsten Ausführung der Pipeline `myadmin3.html` werden die Elemente `<sess><answ/></sess>` erstellt, welches den Text, gemäss der vorangehenden Formularantwort, `yes` oder `no` hat. Ebenso wird in `myadmin3.xsl` nach Kinderevents gesucht. In `myadmin3u.xsl` wird darauf überprüft, ob Kinderevents vorhanden sind. Falls dies nicht der Fall ist und die Löschbestätigung vom Benutzer bejaht wurde, werden die entsprechenden Konferenzteilnehmer analog zur Erklärung über den `filewriter` in Abschnitt 5.3.2 informiert. Ebenso wird das Event von `event` und die Einträge mit der ID des gelöschten Events in `user_event`, `event_person` und `event_ticketType` gelöscht.

Ganz am Schluss der Pipeline findet eine Transformation des Resultates (ob der Event

gelöscht werden konnte oder nicht) mit `mysqladmin3a.xsl` in HTML statt.

5.4 Benutzerverwaltung

5.4.1 Erstellen eines neuen Accounts

Gelangt ein Benutzer auf die URL `apply.html`, so kann er einen eigenen Account erstellen. Das Schwierige bei der Programmierung dieses Teils war die Validierung der eingegeben Daten. Zum Beispiel sollten für den Benutzernamen nicht alle möglichen Zeichen eingegeben werden dürfen, zudem muss überprüft werden, ob der Benutzername bereits existiert. Oder eine Email-Adresse sollte zumindest syntaktisch korrekt sein. Eine grosse Hilfe bei der Lösung dieses Problems war das XSP-Logicsheet[5] formval[6].

Zuerst müssen die zu validierenden Felder im File `apply.descriptor.xml` definiert werden. Eine gekürzte Version sieht dabei so aus:

```
<root>
<parameter name="username" type="string" min-len="4" max-len="20"
   matches-regex="^[A-Za-z1-9\-\_]+$"/>
<parameter name="password" type="string" min-len="6" max-len="20"
   matches-regex="^[A-Za-z1-9\-\_@]+$"/>
<parameter name="conf_password" type="string" min-len="6" max-len="20"
   matches-regex="^[A-Za-z1-9\-\_@]+$"/>
<parameter name="email" type="string" max-len="64"
   matches-regex="^([\w-_.]+@[a-zA-Z-]+\.([a-zA-Z-.]+
                \.[a-zA-Z]{2,6}|[a-zA-Z]{2,6})){1}$"/>
<constraint-set name="apply-form">
   <validate name="title"/>
   <validate name="username"/>
   <validate name="password"/>
   <validate name="conf_password"/>
</constraint-set>
</root>
```

In den `parameter`-Elementen wird der Name und der Typ (string, integer) festgelegt. Weitere Attribute sind zum Beispiel `min-len` und `max-len`, diese geben die minimale beziehungsweise die maximale Länge des Wertes an. Es macht keinen Sinn, hier Werte anzugeben, die im Widerspruch mit der maximalen Länge der korrespondierenden Werte in der Datenbank-Tabelle "user" stehen! Das wichtigste Attribut ist jedoch `matches-regex`, in diesem kann ein regulärer Ausdruck stehen, gegen den dann validiert wird. So ist es nun möglich, den Zeichensatz einzuschränken, oder zum Beispiel sicherzustellen, dass eine Email-Adresse syntaktisch korrekt ist.

Im zweiten Teil müssen mit dem Element `constraint-set` die Parameter zu mindestens einer Validierungsgruppe zusammengefasst werden. Jeder Gruppe muss mit dem

[5] Logicsheets sind Erweiterungen der XSP-Logic, welche viel benutzte Funktionen implementieren, http://xml.apache.org/cocoon/userdocs/xsp/logicsheet-concepts.html.

[6] http://xml.apache.org/cocoon/userdocs/xsp/logicsheet-forms.html.

Attribut `name` ein eindeutiger Name gegeben werden. Durch diese Validierungsgruppen ist es möglich, für verschiedene Formulare gleiche Parameter wieder zu verwenden.

Im File `apply.ERROR.xsp` wird das Dokument generiert, in welchem der Benutzer seine Daten eingeben kann. Der HTML-Code wird zwar immer wieder durch XSP-Logik unterbrochen, die entsprechenden Formulare sollten jedoch erkennbar sein.

Am Anfang des Files wird mit einem Hidden-Tag festgestellt, ob der Benutzer schon die Möglichkeit hatte, etwas auszufüllen. Ist dies nicht der Fall, so wird an den meisten Stellen der XSP-Code ignoriert und es findet keine Validierung statt. Wenn der Benutzer Daten eingegeben hat und den submit-Button drückt, werden die Werte nach dem Descriptor-File validiert. Das heisst das für jeden eingegeben Ausdruck überprüft wird, ob sein Typ stimmt, seine Länge innerhalb der angegebenen Mindest- und Maximal-Länge liegt und sein Ausdruck im Einklang mit dem angegebenen regulären Ausdruck ist. Ist eine Bedingung nicht erfüllt, so generiert das formval-Logicsheet entsprechende Fehlermeldungen. Die Mindest- und die Maximallänge könnten zwar auch im regulären Ausdruck angegeben werden, jedoch ist dann die Fehlersuche schwieriger, da formval nur meldet, dass der eingegeben Ausdruck nicht dem regulären Ausdruck entspricht.

Entstehen dabei Fehler, so wird erneut das `apply.ERROR.xsp`-Dokument aufgerufen. Die vorher eingegeben Werte bleiben erhalten. Zusätzlich können nun mit dem formval-Logicsheet verschiedene Fehlermeldungen abgefangen und ausgegeben werden. Dies geschieht durch die Verknüpfung von Java-Code und XSP-formval-Elementen (erkennbar am `xsp-formval`-Namensraum). Die Fehler werden dann zusätzlich zu den Formularfeldern ausgegeben.

Sind alle Werte validiert, so wird das File `apply.OK.xsp`, aufgerufen. Bis jetzt konnte noch nicht überprüft werden, ob ein Benutzernamen schon existiert. Dies war im vorhergehenden Schritt nicht möglich, da im Descriptor-File keine SQL-Abfrage gemacht werden konnte. Das wird nun nachgeholt. Existiert der Benutzername bereits, so wird kein neuer User in der Datenbank erstellt, und es erscheint eine Fehlermeldung mit einem Link zurück zur ersten Seite, in welcher ein neuer Name gewählt werden muss. Leider konnte dies bis jetzt nicht eleganter gelöst werden.

Ist der Benutzername frei, so werden die Benutzerdaten mittels dem SQL-Befehl `INSERT` in der Tabelle “user” gespeichert und die Registrierung ist abgeschlossen.

Die Pipeline sieht für die Form-Validierung wie folgt aus:

```
<map:match pattern="apply.html">
   <map:act type="form-validator">
      <map:parameter name="descriptor"
           value="conf/apply.descriptor.xml"/>
      <map:parameter name="validate-set" value="apply-form"/>
      <map:generate src="conf/apply.OK.xsp" type="serverpages"/>
      <map:transform src="conf/apply.confirm.xsl">
         <map:parameter name="view-source" value="conf/apply.OK.xsp"/>
      </map:transform>
      <map:serialize/>
   </map:act>
   <map:generate src="conf/apply.ERROR.xsp" type="serverpages"/>
   <map:transform src="conf/apply.confirm.xsl">
```

```
      <map:parameter name="view-source" value="conf/apply.ERROR.xsp"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>
```

Zuerst wird das Descriptor-File ausgewählt (in unserem Fall `apply.descriptor.xml`). Dann muss aus diesem eine Validierungsgruppe selektiert werden. Ist die Validierung ungültig oder fand keine statt, so springt die Pipeline direkt zu `apply.ERROR.xsp`, diese befindet sich aber im zweiten Teil der ganzen Pipeline. Das File `apply.confirm.xsl` transformiert die erzeugten Daten in das Konferenzlayout.

Falls die Validierung in Ordnung ist, so wird wie gehabt `apply.OK.xsp` aufgerufen und die Ausgabe dieses File in das Layout verpackt. Der zweite Teil der Pipeline wird dabei ignoriert.

5.4.2 Editieren eines Accounts

Für diese Funktion mussten zwei bestehende Funktionen verknüpft werden, nämlich das Session-Management mit dem Erstellen eines Accounts. Dies geschieht durch Aufruf der URL `myaccount.html`. Leider konnte dies aufgrund der speziellen Anordnung der Pipeline-Elemente des Formvalidators nicht in die bestehende Session-Pipeline integriert werden, was zur Folge hat, dass viel Code doppelt vorhanden ist. Die Editierfunktion unterscheidet sich nur unwesentlich vom Session-Management und dem Erstellen eines neuen Accounts: Nun wird schon beim ersten Durchlauf validiert (was sinnlos ist, jedoch am einfachsten zu implementieren) und die bisher gespeicherten Werte werden von Anfang an in den Formular-Textboxen angezeigt, mit Ausnahme des Benutzernamens, welcher nachträglich nicht verändert werden kann.

Die Files `myaccount.ERROR.xsp`, `myaccount.OK.xsp` und `myaccount.confirm.xsl` entsprechen den Files die für das Erstellen eines neuen Accounts benötigt werden, sie sind nur mit den schon bekannten Session-Funktionen erweitert worden.

Falls ein Benutzer vom Administrator bearbeitet werden soll, erfolgt dies im Administrationsmenu. Es werden dieselben Dateien wie oben verwendet, jedoch wird dem Administrator vor der Bearbeitung eines Benutzereintrages eine Liste der Benutzer zur Auswahl gegeben (`myadminacc.html`). Nach Auswahl eines Benutzers lässt sich dieser durch Aufruf der URI `myuseracc.html` bearbeiten. Der gewählte Benutzername wird dabei bei der Ausführung von `myaccount.ERROR.xsp` in die Java-Variable `username` eingelesen. Falls der HTTP-Parameter `username` nicht vorhanden ist, wird der Benutzername des eingeloggten Benutzers genommen. Damit werden die Daten des entsprechenden Benutzers aus der Datenbank gewählt und nach Änderung aktualisiert.

5.4.3 Löschen von Benutzern

Das Löschen von Usern erfolgt durch Aufruf von `myuserdel.html`. Es wird dieselbe Pipeline verwendet wie bei den obigen `myadmin.html`-Funktionen. Dies bedeutet, dass eine Authentisierung garantiert wird und die entsprechenden XML-Elemente bereits in der Pipeline vorhanden sind. `myuserdel.xsl` listet alle User auf, welche in der Datenbanktabelle `user` vorhanden sind und gibt sie mit `myuserdela.xsl` als HTML wieder.

Abbildung 5.3: User löschen

Durch eine Auswahl des Benutzers wird dieselbe Pipeline mit denselben XSLT-Dateien durchlaufen. Es erfolgt eine Bestätigung, ob der Benutzer wirklich gelöscht werden soll. Nach Bejahen wird dies ausgeführt. Diese Schritte werden gesteuert durch das Element `action` innerhalb des Root-Elementes `sess`. Der Wert von `action` wird vom HTML-Formular übernommen. Ebenso wird, nachdem ein User gewählt wurde, der Username als Element `username` innerhalb des Root-Elementes `sess` wiedergegeben. Das Element `action` hat folgende Werte: wenn es leer oder nicht vorhanden ist, bedeutet dies, dass alle zur Auswahl stehenden User gezeigt werden. Dies ist normalerweise beim ersten Aufruf von `myuserdel.htm` der Fall. Nach Wahl eines Users (das heisst es existiert ein Wert in `username`) wird das Element `action` auf `confirm` gesetzt und das Bestätigungsformular wird angezeigt. Nach dieser Bestätigung wird der Benutzer gelöscht. Dies wird ausgelöst durch den `doit` als Wert im `action`-Element.

5.5 Konferenzdarstellung

5.5.1 Programm

Die Generierung des Konferenz-Programms beruht auf schon erklärten Mechanismen und beinhaltet keine neuen Ideen. Der Ablauf in der Pipeline sieht wie folgt aus:

```
<map:match pattern="schedule.html">
   <map:generate src="conf/schedule.xml"/>
   <map:transform src="conf/schedule.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:transform type="sql">
      <map:parameter name="use-connection" value="daconf"/>
   </map:transform>
   <map:transform src="conf/schedulea.xsl"/>
   <map:serialize type="html"/>
</map:match>
```

Da das Konferenzprogramm auch für nicht registrierte Benutzer zugänglich ist, braucht es keine Sessioninformationen. Das erste Dokument, `schedule.xml` ist ein fast leeres Dokument, es beinhaltet nur einen beliebigen Tag, der dazu da ist, dass das darauf folgende XSLT-File `schedule.xsl` auf ihn matchen kann.

Im XSLT-File werden die Informationen aus der Datenbank ausgelesen und so aufbereitet, dass das letzte File, `schedulea.xsl` die notwendigen Daten in HTML ausgeben kann.

```
<xsl:param name="parent"/>
<xsl:param name="view"/>
<xsl:param name="level"/>
```

Dieser Code-Teil und die Zeile `<map:parameter name=''use-request-parameters'' value=''true''/>` in der Pipeline, ermöglichen es, in der URL übergebene Variablen abzufangen und im XSLT-File zu verwenden. Die drei Parameter steuern das Navigieren durch das Programm und haben folgende Bedeutung:

view: Hat entweder den Wert "compact" oder "detail" und bestimmt die Ansicht der aktuellen Konferenzseite. Zwischen diesen Ansichten kann der Benutzer durch entsprechende Links auf der Seite hin und her wechseln. Hat der Benutzer keine Ansicht gewählt, so wird diese aus der Datenbank-Tabelle "design" selektiert.

parent: Dieser Wert übergibt die "parent_id" aus der Tabelle "event". Wenn der Benutzer auf einen Event klickt, so werden alle Events gesucht, die dieselbe "parent_id" haben, nämlich diejenige die identisch ist mit der ID des soeben angeklickten Events. Dies erscheint unlogisch, da man erwarten könnte, dass die ID des Events übergeben wird, der angezeigt werden soll. Dies hätte jedoch zur Folge, dass immer nur ein Event angezeigt werden würde, was nicht wünschenswert ist.

level: Dadurch, dass die einzelnen Events in der Datenbank nur einfach verknüpft sind (in einem Event-Eintrag ist nur ersichtlich, von welchem Elternteil er abstammt, die Kinder, falls vorhanden, müssen zuerst gesucht werden), muss dem Programm eine zusätzliche Tiefeninformation gegeben werden. Dies geschieht durch diesen Parameter. Immer wenn im Konferenzprogramm auf ein Kind-Event geklickt wird, so nimmt der Wert gegenüber dem aktuellen Event um eines zu, wird auf den Vaterteil geklickt (was zur Folge hat, dass man alle Events sieht, die dieselbe "parent_id" wie der Vater-Event haben) so nimmt der Level um eines ab.

Diese Parameter steuern die Navigation komplett. Erwähnenswert ist, dass in den entsprechenden SQL-Abfragen nicht nur alle Events mit der selben "parent_id" herausgelesen werden, sondern auch deren Vater-Event (dessen "event_id" ja denselben Wert hat wie die "parent_id" der Kinder).

Im File `schedulea.xsl` werden dann die Informationen dargestellt. Das Dokument ist eigentlich eine Ansammlung von "if-Abfragen" in XSLT. Informationen über einen Event (Room, Date,...) werden nur dann angezeigt, wenn sie vorhanden sind.

5.5.2 myConf

Das Zusammenstellen der persönlichen Konferenzprogrammes bietet keine neuen Probleme: Das vorherige Generieren eines Programms wird mit dem Session-Management verknüpft[7]. Zusätzlich wird für jedes Event noch eine Zeile mit den Boxen "add to myPlan", "by sms" und "by email" hinzugefügt. Werden diese Boxen selektiert und wird anschliessend auf den "update"-Button geklickt, so wird im XSLT-File `myconf.xsl` noch ein INSERT-Befehl ausgeführt, der die selektierten Events in der entsprechenden Tabelle "user_event" abspeichert. Im Ausgabe-file `myconfa.xsl` werden dann schon bisher angeklickte Checkboxen angezeigt und eine Meldung, dass die Events gespeichert worden sind, erscheint.

Nicht selektiert werden können Events, die weder ein Datum noch eine Zeit besitzen. Sie werden nur mit dem Eltern-Event mitselektiert.

5.5.3 myPlan

Das persönliche Konferezprogramm benutzt ebenfalls nur schon bisher vorhandene Funktionen[8]. Zuerst werden in `myplan.xsl` die Events von der Datenbank ausgelesen, die der Benutzer selektiert hat. Die Ausgabe der Events wird in `myplana.xsl` praktisch mit den identischen Funktionen erzeugt wie in myConf und im normalen Programm. Die Events werden dabei chronologisch geordnet. Der einzige Unterschied ist, dass direkte Nachkommen von selektierten Events ebenfalls angezeigt werden. Im Gegensatz myConf können nur noch Events entfernt werden, dies geschieht durch die selben Funktionen wie in myConf.

[7] Hier wurde Code vom normalen Programm wiederverwendet.
[8] Hier wurde Code vom normalen Programm wiederverwendet.

Kapitel 6

Kritische Betrachtungen

6.1 Zielerreichung

Ein Vergleich mit den aufgestellten “Need to Have” und “Nice to Have” Listen zeigt, dass nicht alle “Need to Haves” und lange nicht alle “Nice to Haves” erfüllt werden konnten. Viele der noch nicht implementierten Funktionen sollten jedoch mit einem vernünftigen zusätzlichen Aufwand erstellt werden können.

So fehlen auf der “Need”-Seite das Überprüfen auf Überschneidungen genau so wie eine SMS/Email-Notifizierung kurz vor Beginn eines Events. Die diversen wünschenswerten Darstellungsarten wurden ebenfalls nicht implementiert.

In erster Linie sollten zuerst die Funktionen implementiert werden, die im Moment vom Webadministrator via phpMyAdmin ausgeführt werden müssen.

1. Erstellen und Löschen von eventTypes
2. Erstellen und Löschen von personTypes
3. Verknüpfen von personTypes mit eventTypes

Ebenfalls sehr lohnenswert wäre die Anpassung des CTTM an mehrere Ausgabemedien. Dies lässt sich dadurch bewerkstelligen, dass die XSLT-Transformationen am Ende jeder Pipeline andere Medien ausgeben. Eventuell ist es sinnvoll, sämtliche auszugebenden Seiten zuerst in allgemeinen XML-Zwischenformat zu speichern und dann erst von diesem aus die Tranformation in die gewünschten Medien zu bewerkstelligen. Aufgrund des knappen Zeitrahmens konzentrierten wir uns jedoch ausschliesslich auf HTML. Mit weiteren XSLT-Files (für jedes Ausgabemedium und Pipeline genau eines) lässt sich jedoch diese Erweiterung bewerkstelligen.

Wünschenswert wären komplettere Operationen am “Event-Baum”, zum Beispiel das Abschneiden und Verschieben von kompletten Ästen. Dazu müssen jedoch diverse SQL-Transformationen gemacht werden, welche auf weitaus komplexeren SQL-Abfragen beruhen als sie bis jetzt in diesem CTTM gebraucht worden sind. Wünschenswert wäre auch ein nachträgliches Ändern von Event-Typen. Wenn dies gemacht würde, so müsste

das CTTM sämtliche Events, die Aufgrund der Typen-Änderung nicht mehr gültig sind, anzeigen und vom Administrator verlangen, dass diese angepasst werden.

Die Implementierung eines multilingualen Supports hingegen dürfte einen beträchtlichen Mehraufwand mit sich ziehen. Dieser liese sich wohl nur bewerkstelligen, wenn ein Grossteil des CTTM neu geschrieben und dementsprechend angepasst werden würde.

6.2 Performance

Cocoon ist nicht gerade bekannt, schnell zu arbeiten. Deshalb stellt sich die Frage, wie die Performance verbessert werden kann.

Frontsidecache: Die Literatur schlägt vor, einen separaten Proxy-server als Frontsidecache zwischen Internet und Cocoon zu benutzen. Dieser beschleuingt dem Lesezugriff, jedoch nicht den Schreibzugriff. Da wir jedoch sehr viele Schreibzugriffe haben, fällt diese Methode ausser betracht.

Optimierungsmöglichkeiten in Cocoon: (siehe auch http://xml.apache.org/cocoon/performancetips.html): Grosse Files bremsen die Geschwindigkeit, komplizierte XSLT-Stylesheets ebenfalls. Dies ist bei unserem CTTM jedoch kein Problem. Ebenfalls sollte beim Gebrauch von CTTM das Logging im Cocoon abgestellt werden (natürlich kann dies für die Entwicklung und das Finden von Fehlern benutzt werden). Sämtliche Caches sollten aktiviert werden. Schliesslich empfiehlt sich bei einem regulären Betrieb Cocoon zu veranlassen, nichts ständig zu überprüfen, ob die Sitemap neu geladen werden muss[1].

XSP/XSLT: XSP-Dokumente werden langsamer verarbeitet als XSLT-Dokumente. In dieser Arbeit wurde deshalb wenn immer möglich mit XSLT gearbeitet, nur Probleme die nicht mit XSLT gelöst werden konnten, wurden in XSP implementiert

6.3 Verknüpfung mehrerer Funktionen

Diverse Funktionen sind im Moment getrennt implementiert, obwohl sie untereinander eine grosse Ähnlichkeit haben. Es wäre wünschenswert, wenn alle Funktionen, die das Session-Management benutzen, in der `my*.html`-Pipeline abgearbeitet werden könnten. Dies ist im Moment für das Editieren des eigenen Accounts nicht der Fall.

Ebenfalls ist es ineffizient, dass die Implementation der Account-Erstellung und der Account-Editierung mit verschiedenen Files in verschiedenen Pipelines bewerkstelligt wird. Eine Verknüpfung der beiden Funktionen wäre sicher effizienter.

[1] http://www.wyona.org/docs/xdocs/int-performance.html

6.4 Form-Validierung

Die Form-Validierung funktioniert im Moment nicht mit Umlauten und weiteren sprachspezifischen Buchstaben, selbst wenn im regulären Ausdruck der entsprechende Zeichensatz angegeben wird. Dies liegt vermutlich daran, dass die Auswertung der regulären Ausdrücke im XSP-Formvalidator nicht zu 100% implementiert ist.

6.5 Cocoon

Im Nachhinein betrachtet stellt sich die Frage, ob Cocoon die richtige Wahl für das CTTM war. Unserer Meinung nach ist dies der Fall. Allerdings ist Cocoon viel komplexer als angenommen. Um die vollen Stärken von Cocoon ausnutzen, ist weitaus mehr Aufwand erforderlich als es auf den ersten Blick schien. Vielleicht wäre mit einer traditionellen PHP/MySQL-Lösung in der selben Zeit mehr zu erreichen gewesen. Doch der gegenüber traditionellen Lösungen andere Aufbau von Cocoon sollte nicht zu schnell abgeschrieben werden, das CTTM liese sich auf jeden Fall weiter in Cocoon verbessern.

Zusammenfassung

Die Webseiten von bestehenden Konferenzen sind gemäss unseren Evaluationen sehr unflexibel aufgebaut. Wünschenswert Features wie zum Beispiel einen persönlichen Konferenzplan sind nicht vorhanden. Dies führt zur Idee eines eigenen Konferenz-Tools, welches ein Konferenzprogramm im World Wide Web benutzerdefiniert darstellt. Ein Konferenzteilnehmer kann sich auf dem Web einen Account mit seinen Angaben freischalten und so spezifisch seinen Konferenz-Stundenplan online zusammenstellen.

Mittels Analysen von ausgewählten Konferenzen haben wir uns für die Wahl von verschiedenen vorhanden Einträgen in möglichen Events entschlossen. Diese Einträge sind zum Beispiel Redner, Raum, Zeit, Datum,.... Ein Event wurde so allgemein wie möglich definiert. Aus einer vorgegeben Anzahl von Event-Eigenschaften können Event-Typen definiert werden. Diese definieren, welche Einträge dann ein Event eines bestimmten Typs haben muss.

Die ganze Arbeit ist eine Sammlung von XML-Dateien, welche unter Cocoon, einer Java-Servlet-Lösung, läuft. Diese XML-Dateien haben verschiedene Aufgaben: Sie führen entsprechende Transformationen durch und geben in unserer Arbeit mit XSLT-Transformationen HTML-Code aus. Mit Cocoon kann man mit wenigen Arbeitsschritten die Informationen auch auf alternativen Medien (zum Beispiel WML und PDF). Ferner werden auch andere Transformationen benutzt, beispielsweise um SQL-Datenbanken abzufragen, oder um XML-Files zu schreiben. Cocoon bietet den Vorteil, dass sogenannte Pipelines (welche durch URLs aufgerufen werden) mehrfache Transformationen hintereinander ausführen können. Die jeweils letzte Transformation führt die Informationen, welche in der Pipeline als XML zwischen den Transformationen weitergegeben werden, in ein oben erwähntes Ausgabeformat.

Das Resultat unserer Arbeit ist ein Konferenzschema für relationale Datenbanken (insbesondere MySQL und HSQLDB), die Entwicklung und Implementation der XSLT-Files, welche die Datenbankabfragen vornehmen, die Benutzeranfragen (HTML Formular-Antworten) verarbeiten und entsprechende HTML-Ausgaben machen. Ferner wurde noch ein Ansatz des Email- und SMS-Versandes für die Benutzer-Benachrichtigung implementiert.

Der Vorteil der Lösung mit Cocoon ist dessen Offenheit der Architektur in allen Belangen: Der Browser erhält simplen HTML-Code, auf der Serverseite ist die verarbeitete Information prinzipiell vom Layout oder vom Ausgabemedium unabhängig und es besteht kein Zwang auf eine bestimmte Serverhardware oder an ein bestimmtes Betriebssystem.

Neben der Herausforderung in der Verwendung der Technologien (XML, SQL, XSLT) war es interessant, einmal auf einer Plattform zu programmieren, die uns völlig unbe-

kannt war.

Leider hatten wir keine stattfindende Konferenz zur Verfügung, um unsere Lösung auszuprobieren. Durch die allgemeine Gestaltung unseres Konzeptes lassen sich unsere mitgelieferten Konferenz-Beispielseiten jedoch einfach an eine stattfindende Konferenz anpassen.

Literaturverzeichnis

[1] Bill Brodgen, Conrad D'Cruz, and Mark Gaither. *Cocoon 2 Programming: Web Publishing with XML and Java*. Sybex, first edition, 2002.

[2] Matthew Langham and Carsten Ziegeler. *Cocoon: Building XML Applications*. New Riders Publishing, first edition, July 2002.

Anhang A

Verwendete Hard- und Software

A.1 Server-Software

Unser Projekt wurde auf 3 Systemen mit 3 verschiedenen Konfigurationen ausgetestet:

A.1.1 Konfiguration 1

- AMD K6-2 500MHz CPU, 64 MB RAM, SuSE Linux 8.1, 120 GB HD, 7200 RPM
- Jakarta Tomcat 4.0.4
- Cocoon 2.0.3
- J2SE 1.3.1_04
- MySQL 3.23.52

A.1.2 Konfiguration 2

- Intel Pentium 1500 MHz CPU, 512 MB RAM, SuSE Linux 8.1, 40 GB HD, 7200 RPM
- Jakarta Tomcat 4.0.4
- Cocoon 2.0.3
- J2SE 1.3.1_04
- MySQL 3.23.52

A.1.3 Konfiguration 3

- SPARC Sun Solaris 8, Sun Blade 100,
- Jakarta Tomcat 4.1.18
- Cocoon 2.0.3
- externe Datenbankverbindung auf MysSQL 3.23.44 (SuSE Linux 7.3, Celeron 700 Mhz)
- Cocoon 2.0.3
- J2SE 1.3.01_01

A.2 Entwicklungsumgebungen

- Altova XMLSpy 5, MS Internet Explorer 6.0 und Mozilla 1.3a (unter Windows 2000 und XP)
- Kate, Konqueror und Mozilla 1.0.1 unter KDE 3
- PROTON (unter Windows XP)

Anhang B

Quellcodes

Alle XML- und XSLT-Dateien wurden abgelegt unter `/opt/jakarta/tomcat/webapps/conference/conf`, die Sitemap liegt jedoch in `/opt/jakarta/tomcat/webapps/conference/sitemap.xmap`. Es wurde darauf verzichtet, sämtlichen Quellcode hier einzufügen, einzig und allein die wichtige Sitemap findet sich hier.

B.1 Sitemap

Ein grosser Teil der Sitemap wurde von Cocoon vorgegeben und musste kaum angepasst werden. Für ausführlichere Informationen wird auf die komplette Sitemap verwiesen. Im folgenden File wird nur der von uns modifizierte Teil angegeben.

```
<?xml version="1.0" encoding="UTF-8"?>
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
<map:components>
   <map:generators default="file">
      ...
   </map:generators>
   <map:transformers default="xslt">
      ...
   </map:transformers>
   <map:serializers default="html">
      ...
   </map:serializers>
   <map:matchers default="wildcard">
      ...
   </map:matchers>
   <map:selectors default="browser">
      ...
   </map:selectors>
   <map:actions>
      ...
   </map:actions>
</map:components>

<map:pipelines>
   <map:pipeline>
      ...
      <!-- Here comes our conference pipeline -->
      <map:match pattern="myaccount.html">
      <!-- edit a Account -->
         <map:act type="form-validator">
            <map:parameter name="descriptor" value="conf/apply.descriptor.xml"/>
            <map:parameter name="validate-set" value="apply-form"/>
            <map:generate src="conf/myaccount.OK.xsp" type="serverpages"/>
            <map:transform src="conf/myaccount.confirm.xsl">
               <map:parameter name="view-source" value="conf/myaccount.OK.xsp"/>
            </map:transform>
```

```
      <map:serialize/>
   </map:act>
   <map:generate src="conf/myaccount.ERROR.xsp" type="serverpages">
      <map:parameter name="realm" value="{1}"/>
   </map:generate>
   <map:transform src="conf/myaccount.confirm.xsl">
      <map:parameter name="view-source" value="conf/myaccount.ERROR.xsp"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<!-- generate the standard program -->
<map:match pattern="schedule.html">
   <map:generate src="conf/schedule.xml"/>
   <map:transform src="conf/schedule.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:transform type="sql">
      <map:parameter name="use-connection" value="daconf"/>
   </map:transform>
   <map:transform src="conf/schedulea.xsl"/>
   <map:serialize type="html"/>
 </map:match>
<!-- edit a user account -->
<map:match pattern="myuseracc.html">
   <map:act type="form-validator">
      <map:parameter name="descriptor" value="conf/apply.descriptor.xml"/>
      <map:parameter name="validate-set" value="apply-form"/>
      <map:generate src="conf/myaccount.OK.xsp" type="serverpages"/>
      <map:transform src="conf/myaccount.confirm.xsl">
         <map:parameter name="view-source" value="conf/myaccount.OK.xsp"/>
      </map:transform>
      <map:serialize/>
   </map:act>
   <map:generate src="conf/myaccount.ERROR.xsp" type="serverpages">
      <map:parameter name="realm" value="{1}"/>
   </map:generate>
   <map:transform src="conf/myaccount.confirm.xsl">
      <map:parameter name="view-source" value="conf/myaccount.ERROR.xsp"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<!-- all URL starting with "my" use session-management -->
<map:match pattern="my*.html*">

   <!-- check if login needed -->
   <map:generate src="conf/login.xsp" type="serverpages">
      <map:parameter name="realm" value="{1}"/>
   </map:generate>
   <map:select type="parameter">
      <map:parameter name="parameter-selector-test" value="{1}"/>
      <map:when test="admin3">
          <!-- write down the event's changes here -->
         <map:transform src="conf/my{1}.xsl">
            <map:parameter name="use-request-parameters" value="true"/>
         </map:transform>
         <!-- change db entries -->
         <map:transform type="sql">
            <map:parameter name="use-connection" value="daconf"/>
         </map:transform>
         <!-- get the users and their informations on this event -->
         <map:transform src="conf/my{1}u.xsl"/>
         <map:transform type="sql">
            <map:parameter name="use-connection" value="daconf"/>
         </map:transform>
         <!-- report users on changes -> write to xml file  -->
         <!--check if done, if so: notify users by mail/sms -->
         <map:transform src="conf/my{1}r.xsl"/>
            <map:transform type="filewriter"/>
         <map:transform src="conf/my{1}a.xsl"/>
         <map:serialize type="html"/>
      </map:when>
```

```
      <!-- Choose the settings of an event to be changed  -->
      <map:when test="admin2">
         <!-- chosse all the events settings here to the assigned event -->
         <map:transform src="conf/my{1}.xsl">
            <map:parameter name="use-request-parameters" value="true"/>
         </map:transform>
         <map:transform type="sql">
            <map:parameter name="use-connection" value="daconf"/>
         </map:transform>
         <map:transform src="conf/my{1}a.xsl"/>
         <map:serialize type="html"/>
      </map:when>

      <!-- eg myPlan, myConf -->
      <!-- 2nd stop: chosse all the events settings here to the assigned event -->
      <map:otherwise>
         <map:transform src="conf/my{1}.xsl">
            <map:parameter name="use-request-parameters" value="true"/>
         </map:transform>
         <!-- show all events here -->
         <map:transform type="sql">
            <map:parameter name="use-connection" value="daconf"/>
         </map:transform>
         <map:transform src="conf/my{1}a.xsl"/>
         <map:serialize type="html"/>
      </map:otherwise>
   </map:select>

   <!--check if done, and report answer in html -->
   <!-- show form with event settings-->
   <!-- show all events here in HTML -->
   <!-- show login stuff: form or failed login message or just the content.. -->
   <map:transform src="conf/my{1}a.xsl"/>
   <map:serialize type="html"/>
</map:match>

<!-- create a account -->
<map:match pattern="apply.html">
   <map:act type="form-validator">
      <map:parameter name="descriptor" value="conf/apply.descriptor.xml"/>
      <map:parameter name="validate-set" value="apply-form"/>
      <map:generate src="conf/apply.OK.xsp" type="serverpages"/>
      <map:transform src="conf/apply.confirm.xsl">
         <map:parameter name="view-source" value="conf/apply.OK.xsp"/>
      </map:transform>
      <map:serialize/>
   </map:act>
   <map:generate src="conf/apply.ERROR.xsp" type="serverpages"/>
   <map:transform src="conf/apply.confirm.xsl">
      <map:parameter name="view-source" value="conf/apply.ERROR.xsp"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<map:match pattern="sendmail.html">
   <map:generate src="conf/sendmail.xsp" type="serverpages"/>
   <map:serialize type="xml"/>
</map:match>

<map:match pattern="logout.html">
   <!-- map:act if resource-exists: {1}.xsp ...-->
   <map:generate src="conf/logout.xsp" type="serverpages"/>
   <map:transform src="conf/logout.xsl"/>
   <map:serialize type="html"/>
</map:match>

<map:match pattern="*.html">
   <map:generate src="conf/conference.xml"/>
   <map:transform src="conf/{1}.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:serialize/>
</map:match>
```

```
      <!-- doing some static stuff -->
      <map:match pattern="*.gif">
         <map:read mime-type="image/gif" src="conf/{1}.gif"/>
      </map:match>
      <map:match pattern="*.jpg">
         <map:read mime-type="image/jpg" src="conf/{1}.jpg"/>
      </map:match>
      <map:match pattern="*.css">
         <map:read mime-type="text/css" src="conf/{1}.css"/>
      </map:match>
      <map:match pattern="*">
         <map:redirect-to uri="index.html"/>
      </map:match>
   <map:pipeline>
<map:pipelines>
```

B.2 Übersicht der anderen XML-Dateien

Login/Session-Management: login.xsp, logout.xsp, logout.xsl

Administrierung: myadmin.xsl, myadmina.xsl, myadmin1.xsl, myadmin1a.xsl, myadmin2.xsl, myadmin2a.xsl, myadmin3.xsl, myadmin3r.xsl, myadmin3u.xsl, myadmin3a.xsl

Account-Erstellung: apply.descriptor.xml, apply.ERROR.xsp, apply.OK.xsp, apply.confirm.xsl

Account-Editierung: myaccount.ERROR.xsp, myaccount.OK.xsp, myaccount.confirm.xsl

Standard-Konferenplan: schedule.xml, schedule.xsl, schedulea.xsl

Konferenzplan-Auswahl: myplan.xsl, myplana.xsl

persönlicher Konferenzplan: myconf.xsl, myconfa.xsl