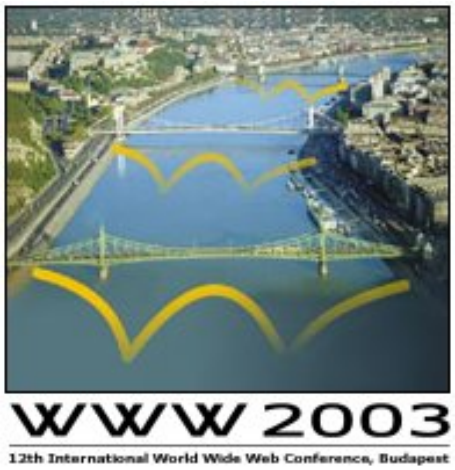
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prof. Bernhard Plattner
Institut TIK
Communications System Group
http://www.csg.ethz.ch/

Semesterarbeit

# Aufbau und Betrieb eines Konferenzplanungssystems für die WWW2003




Michael Feierabend, Thomas Hägi

Sommersemester 2003

Prof. Bernhard Plattner
Betreuer: Prof. Erik Wilde

# Zusammenfassung

In der Diplomarbeit zum Thema “Toolkit für Konferenzprogrammverwaltung und -personalisierung” (CTTM) [KW03] wurde ein Toolkit erstellt, das persönliche Konferenzplanungsprogramme basierend auf XML Technologie ermöglicht. Diese Arbeit konnte an der zwölften World Wide Web Konferenz 2003 in Budapest [Con03] als Poster präsentiert werden. Dieses Toolkit wurde nun in einer neuen Version auf die WWW2003 angewandt und eine Beispielsapplikation - das Konferenzprogramm der WWW2003 selber - an der WWW2003 vorgestellt.

In einem ersten Schritt wurde das bestehende Toolkit installiert und in der Analysephase Bekanntschaft mit dem Cocoon Projekt von Apace [Apa03b] gemacht. In der Implementationsphase wurde das Toolkit von Grund auf neu implementiert und auf die neuen Bedürfnisse angepasst. Der Weg der Neuimplementation war aufgrund des knappen Zeithorizontes die beste Lösung, um möglichst effizient arbeiten zu können.

Entstanden ist eine Lösung, die sämtliche Features des Toolkits beinhaltet und gewisse Ansätze weiter ausführte, wie zum Beispiel die Möglichkeit einer Benachrichtigung per Email über allfällige Veränderungen im Konferenzplan. Zudem wurde das User Interface Design vereinfacht und stärker vom eigentlichen Applikationscode getrennt. Während der Konferenz war der Planer im Internet am Laufen und konnte von den Besuchern der WWW2003 sowie auch an der Präsentation des Posters verwendet werden.

# Inhaltsverzeichnis

# Kapitel 1

# Aufgabenstellung

Die abgeschlossene Diplomarbeit “Toolkit für Konferenzprogrammverwaltung und -personalisierung” [KW03] soll weiterentwickelt und auf die zwölfte World Wide Web Konferenz 2003 (WWW2003) in Budapest angepasst werden. Das Programm dieser Konferenz soll in einer Beispielsapplikation den Besucher zugänglich gemacht werden. Die Besucher sollen sich ihr persönliches Konferenzprogramm zusammenstellen können. Bei Änderungen im Konferenzprogramm sollen die betroffenen Konferenzbesucher benachrichtigt werden.

Das bestehende, aus der Diplomarbeit hervorgegangenen Framework soll dazu termingerecht auf die WWW2003 analysiert und erweitert werden, wobei die verwendeten Technologien beibehalten werden müssen.


Abbildung 1.1: Die Website der WWW2003

# Kapitel 2

# Übersicht

Diese Dokumentation beschreibt die Entwicklung des Konferenzplaners[1] für die 12. internationale World Wide Web Konferenz in Budapest 2003 (WWW2003).

Als erstes wird die Installation des CTTM Frameworks[2] beschrieben. Das Framework wird für Entwicklung und Betrieb des Konferenzplaners benötigt. In der Analyse der vorhandenen Lösung der dieser Semesterarbeit vorausgegangenen Diplomarbeit wird abgeschätzt, inwiefern diese die gestellten Anforderungen erfüllt, und ob allenfalls Teile daraus übernommen werden können.

Die tatsächliche Implementation des Planers ist das Thema des sechsten Kapitels. Es werden Aufbau und dessen Umsetzung beschrieben. In Kapitel 6 werden auf spezielle während der Arbeit entstandene Herausforderungen und Problematiken eingegangen und in Kapitel 7 noch offene Aufgaben beschrieben.

## 2.1 Definitionen

**Definition 2.1.1** *Der Begriff **Toolkit** wird verwendet, um die Lösung der dieser Semesterarbeit vorausgegangen Diplomarbeit "Toolkit für Konferenzprogrammverwaltung und -personalisierung" [KW03] zu beschreiben.*

**Definition 2.1.2** *Die beiden Ausdrücke **CTTM** und **Planer** werden im folgenden verwendet, um den Konferenzplaner für die zwölfte internationale World Wide Web Konferenz in Budapest 2003 zu beschreiben.*

**Definition 2.1.3** ***WWW2003*** *beschreibt die zwölfte internationale World Wide Web Konferenz in Budapest 2003 [Con03] im späteren Verlaufe der Dokumentation.*

[1] Conference Time Table Management (CTTM)
[2] Apache Cocoon in Apache Tomcat mit MySQL Datenbank

**Definition 2.1.4 {*apache-root*}** *ist Platzhalter für das Installationsverzeichnis des Apache HTTP Webservers. Er muss durch dieses Verzeichnis ersetzt werden. Beispiel:* c:\apache2\

**Definition 2.1.5 {*tomcat-root*}** *wird als Platzhalter für das Installationsverzeichnis der Tomcat Installation verwendet und muss bei späteren Anweisungen durch das Verzeichnis ersetzt werden. Beispiel:* c:\sa\tomcat4\

**Definition 2.1.6 {*java-root*}** *ist Platzhalter für das Installationsverzeichnis des Java SDKs. Er muss durch dieses Verzeichnis ersetzt werden. Beispiel:* c:\sa\java\jdk\

**Definition 2.1.7 {*cocoon-root*}** *ist Platzhalter für das Installationsverzeichnis der Cocoon Installation in Tomcat. Er muss durch dieses Verzeichnis ersetzt werden. Beispiel:* c:\sa\tomcat4\webapps\cocoon\

# Kapitel 3

# Installation des Frameworks

Die Aufgabenstellung dieser Arbeit fordert einen Ausbau der bestehenden Lösung und sollte auf den bereits ausgewählten Technologien basieren. Die existierende Lösung verwendet den XML Application Server Cocoon von Apache [Apa03b]. Cocoon selbst benötigt als Laufzeitumgebung den Apache Tomcat [Apa03c] und ist komplett in Java implementiert.

Es stellt sich die Frage, ob direkt die in Apache Tomcat integrierte Rumpfversion des Apache Webservers benutzt werden soll. Wir haben uns aus Performancegründen entschieden, Tomcat in den leistungsfähigeren Apache Webserver zu integrieren. Es besteht jedoch auch weiterhin die Möglichkeit, nur Apache Tomcat zu verwenden.

Als Plattform können alle von Cocoon / Tomcat / Apache / MySQL unterstützten Betriebsystemen verwendet werden. Entwickelt und getestet wurde die Lösung in einer Windows 2000 Umgebung.

## 3.1 Java-SDK

Sowohl Tomcat als auch Cocoon benötigen das Java SDK [Sun03] von Sun in der Version 1.3 oder höher. Für die Entwicklung wurde Version 1.3.1 verwendet. Es wird vorausgesetzt, dass das Java SDK bereits installiert ist und entsprechende CLASSPATHS richtig gesetzt sind. Auf der Sun Java Website [Sun03] stehen entsprechende Downloads und Anleitungen zur Verfügung.

## 3.2 Apache HTTP Webserver

Es kann sowohl Version 1.3 als auch Version 2.0 des Apache Webservers [Apa03a] verwendet werden. Für CTTM wurde die Version 2.0 verwendet, um ein Funktionieren in zukünftigen Apache Webserver Versionen sicherstellen zu können. Die

detaillierten Installationsanleitungen für den Webserver liegen den jeweiligen Sourcen bei oder können online bei Apache eingesehen werden.

Im Folgenden wird angenommen, dass Apache auf dem lokalen Server ("localhost") auf Port "8080" läuft.

## 3.3 Apache Tomcat

Cocoon als Java Applikation benötigt Apache Tomcat [Apa03c]. Tomcat ist ein Java Servlet Engine und bietet die direkte Einbettung von Java Code in Webseiten (Java Server Pages, JSP) an.

Für CTTM wird die aktuellste Version von Apache Tomcat, Version 4.1.24, installiert. Nach der Installation gemäss den von Apache vorgegebenen Anleitungen muss Tomcat mit dem Apache Webserver verbunden werden. Apache Webserver soll Java Serverpage Requests an Tomcat weiterleiten. Dazu muss Tomcat entsprechend konfiguriert, sowie das Apache Webserver Modul *mod_jk* [Sha01] in den Apache Webserver integriert werden.

### 3.3.1 Konfiguration Tomcat

Es muss das File *workers.properties* im Verzeichnis {*tomcat-root*}*/conf/* mit folgendem Inhalt erstellt werden:[1]

```
workers.tomcat_home=<tomcat-root>
workers.java_home=<java-root>
ps= worker.list=ajp13
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
worker.ajp13.lbfactor=1
```

### 3.3.2 Konfiguration Apache Webserver

Das Modul *mod_jk* wird offiziell vom Apache Project unterhalten[2]. Für den Planer erwies sich die Version v1.2.0 als stabil. Im Folgenden wird die Installation in einer Windows-Umgebung beschrieben. Für Unix/Linux ist das Vorgehen analog.

[1] Sämtliche in der Dokumentation verwendeten Source-Listings sind zum einfachen Copy-Paste auf der CTTM Source Code CD im Verzeichnis */dist/* zu finden.

[2] *mod_jk* kann in der aktuellen Version von `http://jakarta.apache.org/builds/jakarta-tomcat-connectors/jk/` bezogen werden

Das Modul muss in *mod_jk.dll* umbenannt und nach {*apache-root*}*/modules/* kopiert werden.

Im Konfigurationsfile *apache-root/conf/httpd.conf* müssen folgende Zeilen angefügt werden:

```
LoadModule jk_module modules/mod_jk.dll

JkWorkersFile "<tomcat-root>/conf/workers.properties"
JkLogFile "<tomcat-root>/logs/mod_jk.log"
JkLogLevel info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "

JkMount /servlet/* ajp13
JkMount /examples/* ajp13
#To open as a web folder
JkMount /webdav ajp13
JkMount /webdav/* ajp13
JkMount /*.jsp ajp13
```

## 3.4 MySQL Datenbank

Die Verwaltung der Konferenzdaten übernimmt eine MySQL Datenbank [MyS03a]. CTTM baut auf der MySQL Version 4.0.12 auf. Da nur Standard-SQL verwendet wird, ist die Kompatibilität mit anderen SQL-92 unterstützenden Datenbanken gewährleistet.

Es kann eine bereits bestehende Installation von MySQL verwendet werden. Wird MySQL jedoch für CTTM neu installiert, empfiehlt es sich nach der standardmässigen Installation dem Datenbankuser *root* ein Passwort zu vergeben. Die Datenbank kann hierzu über die IP 127.0.0.1 auf dem Standard MySQL Port 3306 angesprochen werden.

### 3.4.1 Konfiguration MySQL

In einem ersten Schritt wird eine neue Datenbank erstellt und gleichzeitig mit der Datenstruktur und den Tabelleneinträgen gefüllt. Dazu wird ein SQL Skript verwendet, welches über den MySQL Kommandozeileninterpreter die Datenbank erstellt und initialisiert. Mit dem folgenden Befehl kann dieses Skript ausgeführt werden:

```
mysql -u root --password=<root-password> < mysqlscript.txt
```

*<root-password>* muss hierbei durch das entsprechende Passwort des MySQL *root* Users ersetzt werden. Nach dem Abarbeiten des Skripts sind alle Daten in der Datenbank vorhanden. Nun muss ein neuer Benutzer erstellt werden, mittels welchem CTTM auf die Datenbank zugreifen wird. Aus sicherheitstechnischen Gründen sollte nicht der Root Benutzer für diese Zwecke verwendet werden. Mit dem Befehl

```
GRANT ALL PRIVILEGES ON cttm.* TO <username>@localhost identified by '<password>'
```

kann ein Benutzer erstellt werden, der von der lokalen Maschine Zugriff auf die Datenbank *cttm* hat. *<username>* und *<password>* können dabei frei gewählt werden, müssen dann aber bei der Konfiguration des Cocoon MySQL Treibers entsprechend angepasst werden. Wird die Datenbank nicht auf demselben Rechner wie Cocoon installiert, muss *localhost* durch die entsprechende IP des Rechners ersetzt werden. Ein *%* erlaubt Zugriff unabhängig von der IP-Adresse des Clients.

## 3.5 Apache Cocoon

CTTM ist ein XML/XSP/XSLT Projekt basierend auf dem Apache Cocoon Java Servlet. Dieses liegt in der stabilen binary Version 2.0.4[3] vor. Ein Arbeiten mit der Betaversion Cocoon 2.1 ist aufgrund des Entwicklungstandes noch nicht zu empfehlen.

Cocoon wird zunächst in Tomcat installiert. Anschliessend muss der Cocoon MySQL Treiber konfiguriert sowie die Cocoon Sitemap für CTTM angepasst werden.

### 3.5.1 Installation von Cocoon

Aus der Cocoon Distribution [Apa03b] muss das Java Web Archive (WAR)[4] *cocoon.war* nach {*tomcat-root*}*/webapps/* kopiert werden. Durch Aufrufen der URL *http://localhost:8080/cocoon/* über einen Webbrowser wird das Archive durch Tomcat entpackt und installiert. Es existiert nun das Verzeichnis {*tomcat-root*}*/webapps/cocoon/*.

### 3.5.2 Installation der CTTM Applikation

Die CTTM Applikation[5] muss nach {*tomcat-root*}*/webapps/cocoon/* entpackt werden. Nach erfolgreichem Entpacken befindet sich der Konferenzplan für die WWW2003 in {*tomcat-root*}*/webapps/cocoon/cttm/www2003/*.

[3] Apache Cocoon 2.0.4:
`http://sunsite.cnlab-switch.ch/www/mirror/apache/dist/cocoon/BINARIES/`

[4] Ein Web Archive (WAR) repräsentiert eine Webapplikation und enthält somit deren benötigten Klassen, Bilder, usw. sowie einen Descriptor.

[5] Datei */dist/cttm.zip* auf der CTTM Source Code CD

### 3.5.3 Installation und Konfiguration des MySQL Treibers

Als Java MySQL-Treiber wird *j/connector* in der Version 3.0.7 [MyS03b] verwendet. Das entsprechende jar-File *mysql-connector-java-3.0.7-stable-bin.jar* muss dazu nach {*cocoon-root*}*/WEB-INF/lib/* kopiert werden.

Im Konfigurationsfile {*cocoon-root*}*/WEB-INF/web.xml* muss der MySQL Connector in der Sektion <*init-param*><*param-name*>*load-class*</*param-name*> (Zeile 112) geladen werden:

```
<init-param>
   <param-name>load-class</param-name>
   <param-value>
      <!-- For IBM WebSphere:
         com.ibm.servlet.classloader.Handler
      -->
      <!-- For Database Driver: -->
         org.hsqldb.jdbcDriver
      <!-- For parent ComponentManager sample:
         org.apache.cocoon.samples.parentcm.Configurator
      -->
   </param-value>
</init-param>
```

Die direkte logische Verbindung zur CTTM MySQL Datenbank wird anschliessend in {*cocoon-root*}*/WEB-INF/cocoon.xconf* definiert. In der Sektion <*datasources*> (Zeile 406) wird eine neue JDBC-Verbindung definiert. Dabei muss beachtet werden, dass die Werte für <*user*> und <*password*> mit den bei der MySQL Installation gewählten übereinstimmen. Gegebenenfalls müssen IP und Port angepasst werden

```
<datasources>
   <jdbc name="daconf">
   <pool-controller min="5" max="10"/>
      <auto-commit>true</auto-commit>
      <dburl>jdbc:mysql://127.0.0.1:3306/cttm</dburl>
      <user><user></user>
      <password><password></password>
      <driver>org.gjt.mm.mysql.Driver</driver>
   </jdbc>
   <jdbc...
```

### 3.5.4 Konfiguration der Cocoon Sitemap

CTTM benötigt einen zusätzlichen Transformer (*filewriter*), den dazu passenden Serializer sowie die Pipelines für die einzelnen Seitenaufrufe. Diese Einstellungen werden in der Cocoon Sitemap {*cocoon-root*}*/sitemap.xmap* vorgenommen.

#### 3.5.4.1 Hinzufügen des Transformers filewriter

In der Sitemap nach *<map:transformers default="xslt">* suchen und einen neuen TransformerEintrag erstellen:

```
<map:transformers default="xslt">
<!-- ================  CTTM  =========================== -->
   <map:transformer
         name="filewriter"
         logger="sitemap.transformer.tofile"
         src="org.apache.cocoon.transformation.SourceWritingTransformer"
   >
      <serializer>my-xml-serializer</serializer>
   </map:transformer>
<!-- ================  CTTM  =========================== -->
   <map:transformer...
```

#### 3.5.4.2 Hinzufügen des Serializers für filewriter

Passend zum Transformer muss ein XML Serializer für den *filewriter* erstellt werden. In der Sitemap nach *<map:serializers default="html">* suchen und einen neuen Eintrag erstellen:

```
<map:serializers default="html">
<!-- ================  CTTM  =========================== -->
   <!-- xml serializer for the file writer: remove enconding type etc...
      alternatively, you could define  my-xml-serializer (see definition
      of the filewriter transformer above...) for file writing purposes -->
   <map:serializer
         name="my-xml-serializer" mime-type="text/xml"
         logger="sitemap.serializer.myxml"
         src="org.apache.cocoon.serialization.XMLSerializer"
   >
      <!--<doctype-system>urn:hrc:doc</doctype-system>-->
      <encoding>utf-8</encoding>
      <omit-xml-declaration>yes</omit-xml-declaration>
      <indent>2</indent>
   </map:serializer>
<!-- ================  CTTM  =========================== -->
   <map:serializer...
```

#### 3.5.4.3 Hinzufügen der CTTM/WWW2003 Pipeline

In der Sitemap nach *<map:pipelines>* und dorthin die CTTM Pipeline für die WWW2003 Konferenz kopieren:

```
<map:pipelines>

<!-- ================  CTTM  ========================== -->
<map:pipeline>

<map:match pattern="cttm/www2003/*.html">
   <map:generate src="cttm/www2003/1.xml" type="serverpages"/>
   <map:transform src="cttm/www2003/1.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<map:match pattern="cttm/www2003/login/register.html">
   <map:act type="form-validator">
      <map:parameter name="descriptor"
            value="cttm/www2003/login/register_description.xml"/>
      <map:parameter name="validate-set" value="apply-form"/>
      <map:generate
            src="cttm/www2003/login/register_ok.xsp" type="serverpages"/>
      <map:transform src="cttm/www2003/login/register_confirm.xsl">
         <map:parameter name="view-source"
               value="cttm/www2003/login/register_ok.xsp"/>
      </map:transform>
      <map:serialize/>
   </map:act>
   <map:generate
         src="cttm/www2003/login/register_error.xsp" type="serverpages"/>
   <map:transform src="cttm/www2003/login/register_confirm.xsl">
      <map:parameter name="view-source"
            value="cttm/www2003/login/register_error.xsp"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<map:match pattern="cttm/www2003/myplan/index.html">
   <map:generate src="cttm/www2003/myPlan/index.xml" type="serverpages"/>
   <map:transform src="cttm/www2003/myPlan/index.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<map:match pattern="cttm/www2003/myplan/change.html">
   <map:generate src="cttm/www2003/myPlan/change.xml" type="serverpages"/>
```

```
   <map:transform src="cttm/www2003/myPlan/change.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<map:match pattern="cttm/www2003/*/*.html">
   <map:generate src="cttm/www2003/1/2.xml" type="serverpages"/>
   <map:transform src="cttm/www2003/1/2.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>

<!-- static stuff -->
<map:match pattern="cttm/www2003/img/*.gif">
   <map:read mime-type="image/gif" src="cttm/www2003/img/1.gif"/>
</map:match>
<map:match pattern="cttm/www2003/img/*.jpg">
   <map:read mime-type="image/jpg" src="cttm/www2003/img/1.jpg"/>
</map:match>
<map:match pattern="cttm/www2003/css/*.css">
   <map:read mime-type="text/css" src="cttm/www2003/css/1.css"/>
</map:match>
<map:match pattern="cttm/www2003/*">
   <map:redirect-to uri="index.html"/>
</map:match>

</map:pipeline>
<!-- ================  CTTM  ========================== -->
<map:pipeline...
```

## 3.6 Testen des Frameworks

Sind alle Teile des Frameworks installiert, kann die CTTM WWW2003 Applikation durch Aufrufen der URL *http://localhost:8080/cocoon/cttm/www2003/* gestartet werden. Die untenstehende Startseite sollte nun erscheinen.

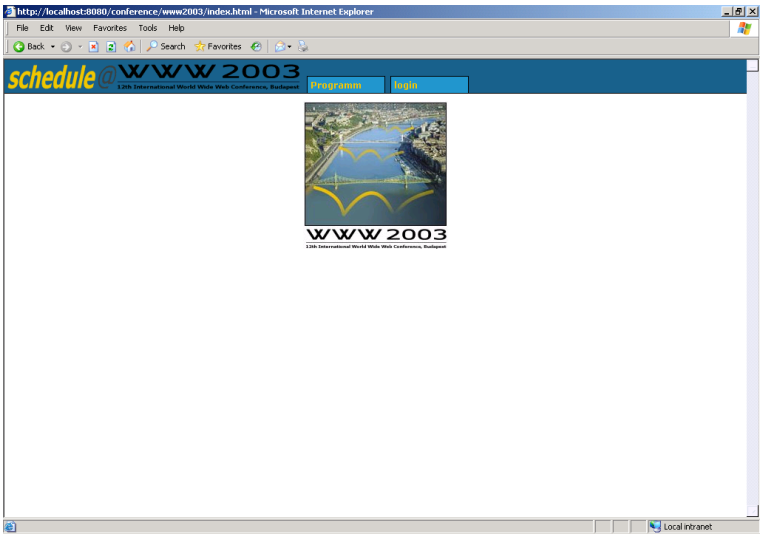



Abbildung 3.1: Einstiegsseite des Konferenzplaners für die WWW2003 in Budapest

# Kapitel 4

# Analyse der bestehenden Lösung

In einem ersten Schritt musste die bestehende CTTM Lösung aus der Diplomarbeit "Toolkit für Konferenzprogrammverwaltung und -personalisierung" [KW03] installiert werden. Es stellte sich heraus, dass in ihrer Lösung die Cocoon Installation direkt in die CTTM Applikation eingebunden ist. In einem ersten Schritt mussten deshalb die tatsächlichen Projektsourcen von der Cocoon-Installation getrennt werden. Dadurch wurde verständlich, welche Files relevant sind. Zudem wurde die Möglichkeit geschaffen, CTTM auf eine beliebige, eventuell neuere Variante von Cocoon zu portieren. Durch die Entkoppelung haben wir uns intensiver mit der Filestruktur und der Navigationslogik von CTTM auseinandersetzen müssen. Aus der Sitemap galt es die notwendige Pipelinestruktur zu extrahieren. Schliesslich konnte das Projekt erstmals im CTTM Framework gestartet und dessen Datenmodell, Funktionalität sowie Wiederverwendbarkeit analysiert werden.

## 4.1 Datenmodell

Die bisherige Lösung hat nur eine Demo-Konferenz implementiert. Es galt deshalb herauszufinden, inwieweit sich das Datenmodell für die Struktur der WWW2003 Konferenz überhaupt eignet. Zu prüfen war, ob die verwendeten Modellierungsarten den Anforderungen genügen oder ob sie allenfalls ergänzt und verändert werden müssen.

Im bestehenden Datenmodell gibt es drei Schwerpunkte:

- "Events" beschreiben die Konferenz an sich. Jeder Event hat Datum, Beschreibung, Links und weitere Meta-Informationen. Ein Event kann keinen, einen oder mehrere Subevents haben.

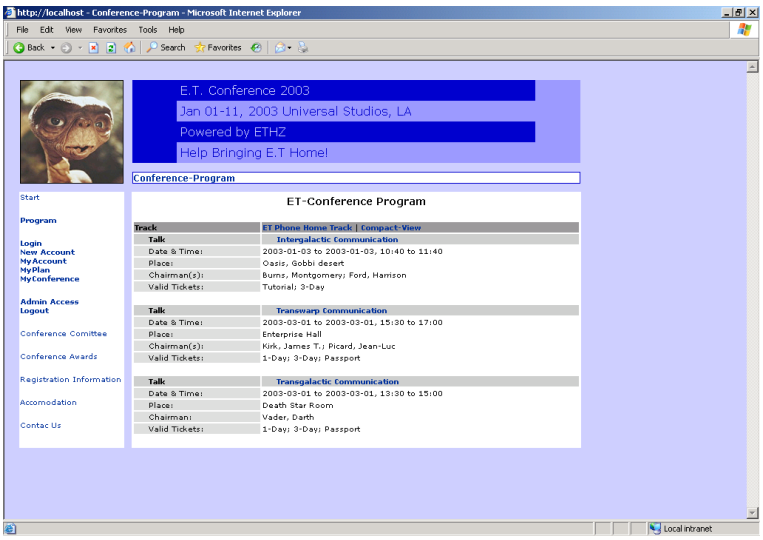



Abbildung 4.1: Die bestehende CTTM Beispielslösung

- "Persons" werden für Veranstalter und Präsentatoren der einzelnen Events verwendet. Eine "Person" beschreibt somit Name, Adresse und weitere personenspezifische Daten.
- "Users" enthalten die Daten über die registrierten Konferenz- bzw. Websitebesucher.

Das Modell wird durch "Event-Types" und Beziehungen der verschiedenen Schwerpunkte ergänzt. Die Modellierung erscheint solide und der Anwendung dienlich. Sie erlaubt die nötigen Relationen zwischen Events und Persons, aber auch Events und Users. Es waren noch einige mögliche Veränderungsvorschläge aufgetaucht. Diese zu evaluieren hätte jedoch mehr Zeit benötigt als zur Verfügung stand. In Kapitel 7 werden diese Ideen für mögliche Nachfolgeprojekte näher erläutert.

Für die nun vorliegende Version von CTTM wurde die Datenbank trotzdem neu erstellt. Die Namensgebung an einigen Orten der alten Datenbank ist nicht intuitiv und wurde deshalb im neuen Modell verbessert. Für die Personalisierung des Konferenzplaners mussten zudem den Event-Types diverse neue Attribute gegeben werden, welche vor allem im Seitenaufbau und in der Navigation durch das Konferenzprogramm ihre Anwendung finden.

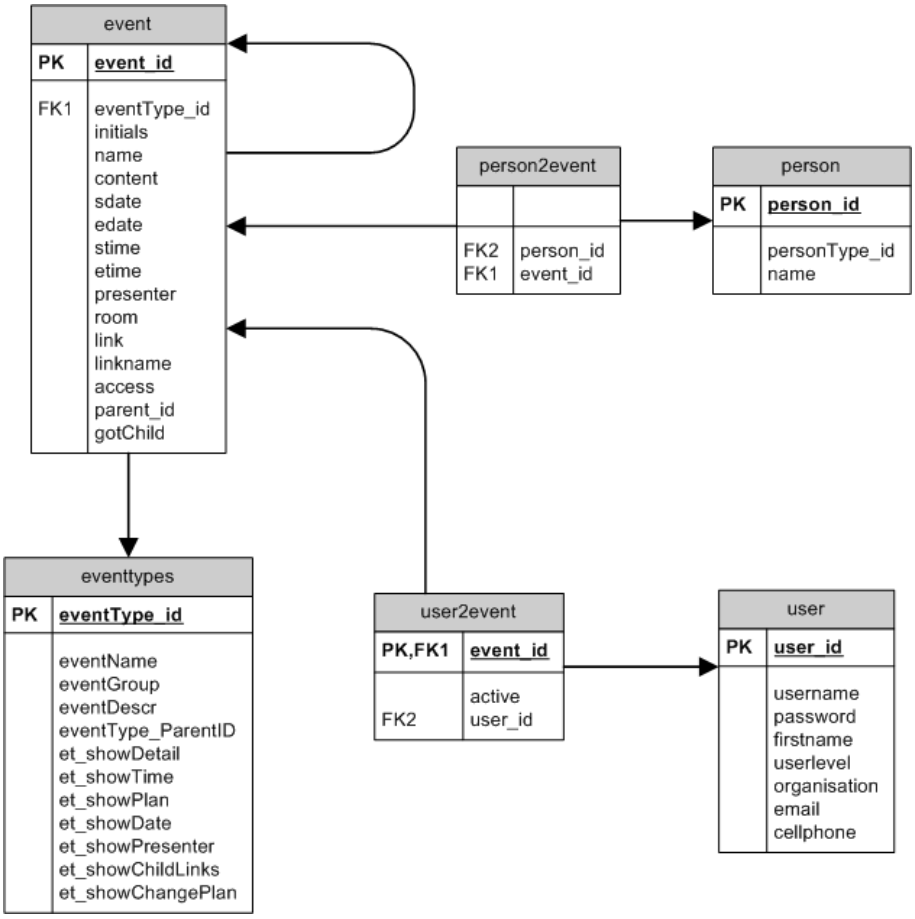



Abbildung 4.2: Das Datenmodell

## 4.2 Logischer Aufbau

In einem zweiten Schritt wurde der logische Aufbau der Seite genauer betrachtet. Neben der Navigation und dem Betrachten des eigentlichen Konferenzprogramms musste der Registrierung und Anmeldung von Benutzern sowie deren persönlicher Konferenzplan analysiert werden.

Grundsätzlich wird die Navigation in zwei Sparten aufgeteilt. Es handelt sich dabei einerseits um öffentliche Seiten, die jedem (auch unregistrierten) Benutzer der Seite zugänglich sind, andererseits die Bereiche, welche nur registrierten Besuchern zugänglich sind:

Neben den allgemeinen Informationen über Organisation, Anfahrt und Kontakt ist auch das eigentliche Konferenzprogramm für jedermann einsehbar. Das Programm ist hierarchisch nach dem zeitlichen Ablauf aufgebaut und bietet zwei verschiedene Ansichtsmöglichkeiten. Neben einem kompakten Anzeigemodus, in dem nur Veranstaltungsname, Ort, Zeit und Referent dargestellt werden, besteht die Möglichkeit einer Detailanzeige mit zusätzlichen vorhandenen Informationen wie Kurzbeschrieb und Links anzuzeigen. In der Datenbank ist für jeden Eventtypus festgelegt, welche Ansicht standardmässig dargestellt werden soll.

Die nur registrierten Benutzern zugängliche Seiten enthalten das personalisierte Konferenzprogramm. Der Benutzer kann sich aus dem kompletten Konferenzprogramm seine favorisierten Events aussuchen und in einen eigenen persönlichen Konferenzplan einfügen. So können bei Planänderungen die Benutzer gezielt informiert und beispielsweise während der Konferenz über die nächsten Veranstaltungsorte orientiert werden.

An der logischen Struktur des Programmaufbaus ist nichts zu bemängeln. Jede Konferenz ist in ihren Grundzügen hierarchisch aufgebaut, womit ihr Programm ohne grosse Probleme in die bestehende Struktur abzubilden sein wird. Weiter ist es sinnvoll und nötig zwischen einem öffentlichen und einem privaten Bereich zu trennen: Alle Besucher müssen die Möglichkeit haben, den kompletten Konferenzplan betrachten zu können, andererseits sollten registrierte Benützer ihre persönliche Kollektion an Events auswählen können.

Bei der bestehenden Lösung fehlt jedoch die Übersicht. Wenn ein registrierter Benutzer sich entschieden hat, einen Event in seinen persönlichen Konferenzplan einzufügen, so sollte dies auch im Hauptkonferenzplan ersichtlich sein.

Im Allgemeinen ist die wünschenswerte Grundfunktionalität eines Konferenzplaners vorhanden und kann mit Anpassungen für die Anwendung auf die WWW2003 Konferenz übernommen werden.

## 4.3 Erweiterte Funktionalität

In einem weiteren Schritt wurden zusätzliche Funktionalitäten wie Email- und SMS-Versand an registrierte Benutzer analysiert. Diesen Funktionen wurden im Aufbau bereits Plätze zugewiesen. Ein Benutzer kann angeben, ob er bei Änderungen per Email und/oder SMS informiert werden möchte. Die eigentliche Funktionalität aber wurde noch nicht implementiert. Ebenso wenig wurden mögliche, in diesem Zusammenhang auftretende Probleme evaluiert.

Für die WWW2003 ist nun die Funktionalität einer Benachrichtigung per Email integriert. Falls ein SMS Gateway zur Verfügung stünde, könnte auch die Benachrichtigung über Mobile Phones rasch eingebaut werden.

## 4.4 Wiederverwendbarkeit

Die bestehende Lösung wurde für eine Beispielkonferenz rund um E.T. den Ausserirdischen aufgebaut. Das Layout ist dabei rudimentär gehalten und musste für WWW2003 komplett erneuert werden. Es muss für künftige Konferenzen einfach und unkompliziert angepasst werden können. Um die Wiederverwendbarkeit der bestehenden Lösung diesbezüglich überblicken zu können, musste zuerst die Sitemap der Cocoon Anwendung genauer studiert werden.

Die Pipelinestruktur zum Erstellen einer Seite wurde in zwei verschiedene Kategorien unterteilt: Statische Seiten und dynamische Seiten, welche Abfragen auf die Datenbank enthalten.

Die statischen Seiten gehen in der Pipeline von einem allgemeinen XML File aus, welches ausschliesslich ein Tag <*conference*> enthält. Dieses XML File wird mittels eines XSLT Stylesheets mit Daten angereichert und basierend auf die inkludierten Funktionen in ein HTML File transformiert. Der grosse Teil der Layoutstruktur liegt in diesen eingebundenen Funktionen. Dadurch kann diese Struktur an einem einzelnen Ort verwaltet werden. Ändert man in dem importierten File das Layout, so werden sämtliche Files, die diese Datei importieren, diese Änderungen ebenfalls erfahren. Allerdings kann nicht das komplette Layout in diese Datei ausgelagert werden. Gewisse Teile müssen bereits im XSLT Stylesheet zu HTML formatiert werden und weiter übergeben werden.

In den dynamischen Seiten müssen zusätzlich Informationen aus der Datenbank gelesen werden. Hier wurde ein weiterer Schritt in der Pipeline eingebaut. Ausgegangen wird immer noch vom allgemeinen XML File mit nur einem Tag. Nun wird jedoch im ersten Schritt durch die XSL Transformation diese XML Datei zuerst mit sämtlichen Daten aus der Datenbank angereichert. Nach dieser ersten Transformation erhält man eine XML Datei, welche alle notwendigen Daten enthält.

Anschliessend werden durch weitere Transformationen analog der statischen Seiten eine HTML Seite erstellt.

Dies entspricht dem allgemeinen Vorgehen einer Cocoon Anwendung. Die Rohdaten werden durch eine oder mehrere Transformationen in das gewünschte Format gebracht, um schliesslich über den Serializer ausgegeben zu werden. Damit wird das Sammeln und Bereitstellen der Daten möglichst von der Darstellung getrennt. Leider ist es vor allem bei grossen Datenmengen nicht immer möglich, die Trennung komplett durchzuziehen; ein Mischen von Datenaufbereitung und Darstellung lässt sich nicht immer verhindern.

## 4.5 Zusammenfassung

Der Aufbau und die Umsetzung des ursprünglichen CTTM wurden gewissenhaft und sauber gemacht. Trotzdem müssen einerseits Teile im Datenmodell auf die konkreten Anforderungen des Konferenzplaners für die WWW2003 angepasst werden. Andererseits ist die Dateistruktur der gesamten Applikation sehr schwierig zu übernehmen: Administration, persönliches Konferenzprogramm und das Hauptprogramm wurden teils über mehrere Dateien verteilt, teils in einer einzigen implementiert.

Um in Anbetracht der knappen Zeit keine mühsame Auftrennung und kein erneutes Debugging des bestehenden Codes durchführen zu müssen, haben wir uns entschlossen, die Applikation von Grund auf neu zu erstellen und uns dabei in Teilen an den technischen Lösungen der bestehenden Applikation zu orientieren.

Generell war der Code der bestehenden Lösung wenig dokumentiert und erschwerte somit ein komplettes Nachvollziehen des Ablaufs zur Generierung der Seite. Die Implementation von Beginn weg neu zu machen ersparte daher mehr Zeit als das Umgestalten des vorhandenen Rohgerüstes gekostet hätte.

Da Cocoon generell schlecht dokumentiert ist und kaum Beispiele oder Anleitungen zu finden sind, war die vorhandene Lösung als Vorlage und Beispiel bei vielen Problemen aber äusserst nützlich. Gewisse Ansätze zur Implementation von Teilproblemen wurden somit schon präsentiert, während eine Suche nach einem passenden Beispiel im Internet allzu oft nicht von Erfolg gekrönt und viel zeitraubender gewesen wäre.

# Kapitel 5

# Implementation

Dieses Kapitel widmet sich der eigentlichen Implementation des Konferenzplaners für die WWW2003. Es wird der Aufbau des Planers erläutert und auf dessen technische Implementation eingegangen. Dabei werden verschiede Schwerpunkte berücksichtigt, welche interessante Erkenntnisse abdecken. Teilweise wird auf Änderungen im Bezug auf die bestehende Lösung aufmerksam gemacht. Das Hauptaugenmerk wird jedoch auf die Implementation der neuen Version von CTTM gelegt.

## 5.1 Aufbau

Der Aufbau der neuen Version lehnt sich stark an die bestehende an. Es sollen "öffentliche" und "private" Bereiche der Website erstellt werden. Während die öffentlichen Bereiche auch von nicht registrierten Besuchern eingesehen werden können, braucht es zur Nutzung der privaten Sektionen eine vorausgehende Benutzerregistration.

### 5.1.1 Allgemein zugängliche Bereiche

Im allgemein zugänglichen Bereich befindet sich die Einstiegsseite zum Konferenzplaner. Sie wurde schlicht gehalten und nur mit dem eigentlichen Logo der Konferenz besetzt. Ursprünglich war geplant, weitere Konferenzinformationen zu publizieren. Die Kommunikation mit den Ansprechpartnern von WWW2003 hat sich aber als schwierig erwiesen. Da CTTM in erster Linie als Beispielimplementation von XML/XSLT/XSP im Internet dienen sollte, wurde das Hauptaugenmerk deshalb auf den eigentlichen Konferenzplan gerichtet.

Der Konferenzplan präsentiert sich dem Besucher auf einer Übersichtseite. Der Besucher erhält einen Komplettüberblick über die gesamte Konferenz. Er kann - von dieser Übersichtseite ausgehend - die Detailprogramme der einzelnen Bereiche anwählen. Der Aufbau dieser Übersichtsseite ist stark von der Konferenz selbst abhängig. Deshalb werden die Daten vorerst in einem statischen File gespeichert.

Wird CTTM weiterentwickelt, so wäre eine dynamisch generierte Übersichtsseite denkbar.

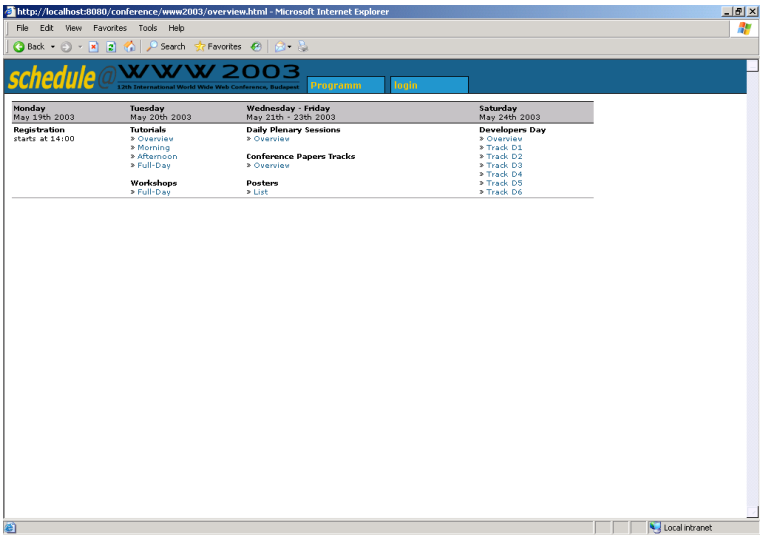



Abbildung 5.1: Übersicht über das komplette Konferenzprogramm

Wählt der Benutzer einen der verschiedenen Konferenzbereiche, beispielsweise “Tutorials”, aus, erhält er die genauen Tagespläne dieser Veranstaltung. Je nach Typ der Veranstaltung kann beliebig tief geschachtelt werden. So kann beispielsweise ein Tutorial aus mehreren kleineren Tutorials bestehen, welche wiederum in verschiedene kleinere Bereiche unterteilt sein können. Die Detailtiefe wird hierbei direkt in der Datenbank für den jeweiligen Eventtype bestimmt.

Jeder Event kann neben “Namen”, “Raum”, “Ort” und “Datum/Zeit” optional eine “Description”, einen “Link” und einen oder mehrere “Presenters” aufweisen.

Über eine Navigationszeile kann sich der Benutzer jederzeit über seinen momentanen Aufenthaltsort innerhalb der Website orientieren.

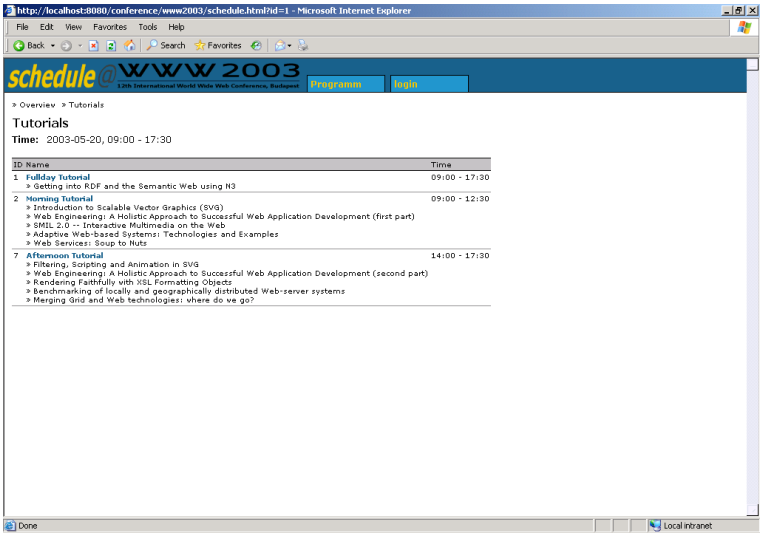



Abbildung 5.2: Übersicht der an der WWW2003 gehaltenen Tutorials

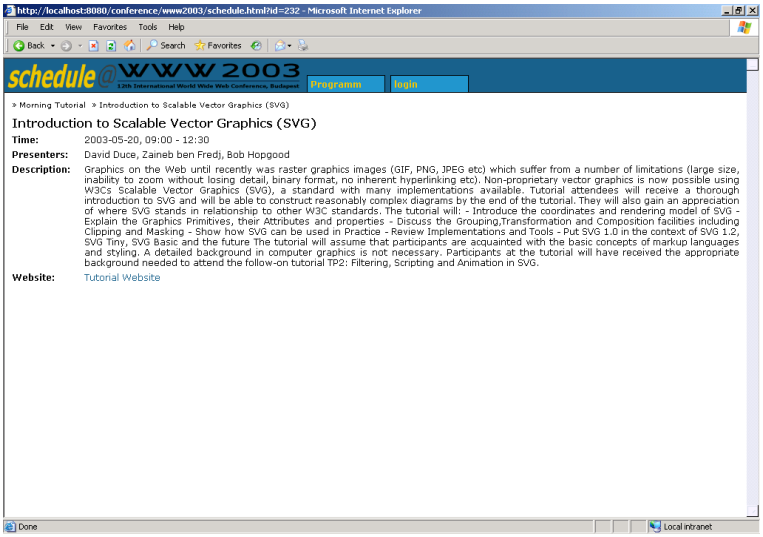



Abbildung 5.3: Detailansicht eines Tutorials

### 5.1.2 Bereiche für registrierte Benutzer

Registriert sich der Benutzer, kann er sich seinen persönlichen Konferenzplan zusammensetzen. Über eine Registration können sich neue Benutzer anmelden. Dazu können neben Namen, Passwort und Email auch weitere, optionale Benutzerangaben wie beispielsweise mobile Telefonnummer gemacht werden.

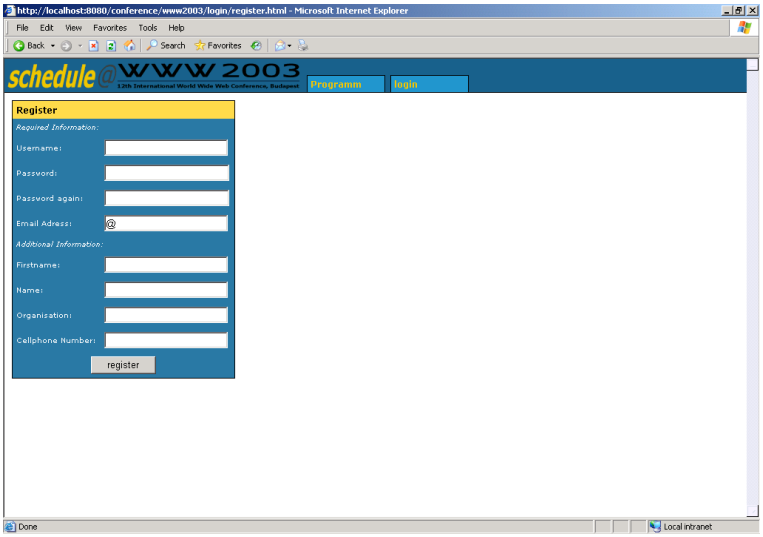



Abbildung 5.4: Registrationsseite im CTTM

Ist der Benutzer registriert, kann er sich mit seinen persönlichen Daten einloggen. Einmal eingeloggt, erscheint in der Navigation zusätzlich ein Link auf MyPlan, dem persönlichen Konferenzplan des Benutzers. Im eigentlichen Konferenzplan erscheinen nun bei jedem Event Buttons, über die der aktuelle Event in den persönlichen Plan eingefügt oder aus diesem entfernt werden kann. Um zusätzlich bereits ausgewählte Events graphisch hervorzuheben, werden die komplett Ausgewählten (alle Unterevents sind ebenfalls selektiert) im Konferenzplan durch ein helles Rot, teilweise ausgewählte mit Blau unterlegt.

Im MyPlan erhält der Benutzer schliesslich sämtliche ausgewählten Events zeitlich in der richtigen Abfolge aufgelistet. Über Links kann er direkt zur entsprechenden Detailseite im kompletten Konferenzplan wechseln.

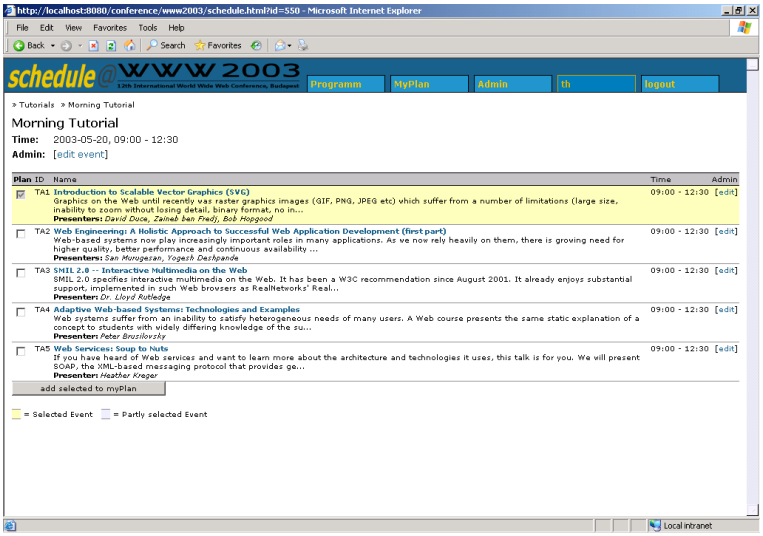



Abbildung 5.5: Öffentlicher Konferenzplan als eingeloggter Benutzer

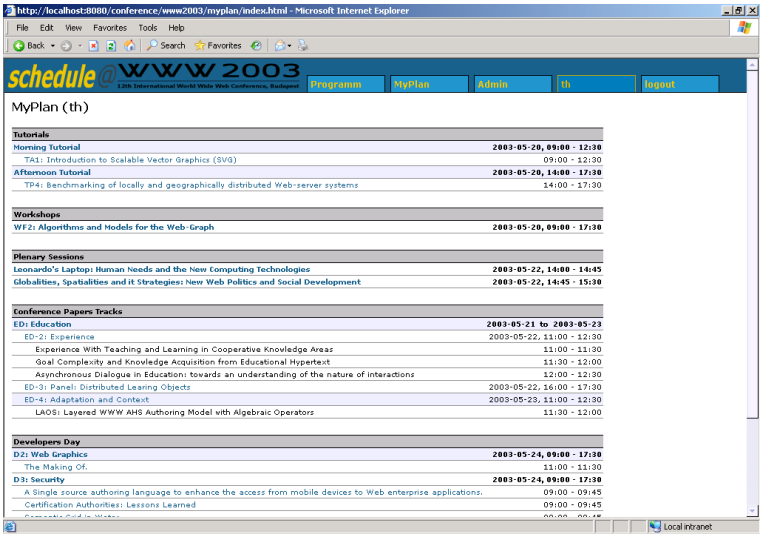



Abbildung 5.6: Persönlicher Konferenzplan als eingeloggter Benutzer

### 5.1.3 Administrationsbereich

Hat ein eingeloggter Benutzer Administrationsrechte, so kann er einerseits jeden Event direkt im Konferenzplan editieren, neue Unterevents hinzufügen oder löschen, sowie über das “Admin” Menu die registrierten Benutzer verwalten. Tätigt er Änderungen an einem Event, kann er optional diese automatisch per Email allen Benutzern mitteilen, die diesen Event in ihrem MyPlan eingefügt haben.

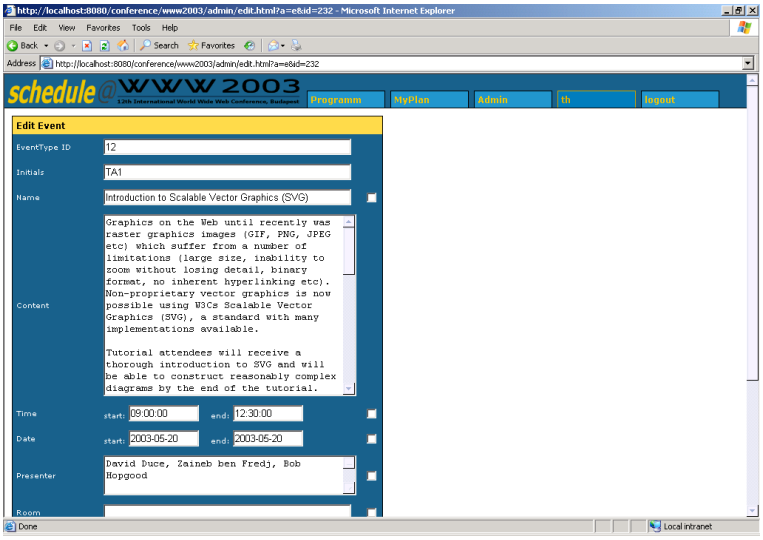



Abbildung 5.7: Editieransicht eines Events im Administrationsmodus

## 5.2 Technische Implementation

Dieses Kapitel beinhaltet die technischen Aspekte der Implementation. Es wird nicht mehr zwischen dem (inhaltlich) ungleichen Seitenaufbau unterschieden, sondern die verschieden technischen Anforderungen und Lösungen der einzelnen Seiten vorgestellt.

### 5.2.1 Session Management

Besonders wichtig für personalisierte Webseiten ist das Session Management. Der gesamte Zeitraum des Besuches auf der Konferenzplaner-Seite wird in einer Benutzersession zusammengefasst und so Informationen transparent von Seite zu Seite weitergegeben. Damit eine Besuchersession einem bestimmten Benutzer zugeordnet und dessen persönliche Daten geladen werden können, muss sich dieser über eine Login-Seite anmelden.

Technisch gesehen muss eine Variable von einer Seite zur nächsten mitgegeben werden, welche uns die notwendigen Informationen (Session-ID) über den Besucher mitteilt. Cocoon stellt dem Programmierer für diese Problematik ein *<session>* Objekt zur Verfügung. Dieses Objekt bedient sich einer XSP Bibliothek, welche die notwendige Funktionalität zur Verfügung stellt. Um auf dieses Objekt zurückgreifen zu können muss im Header des jeweiligen Files[1] der entsprechende Namespace deklariert werden:

```
<?xml version="1.0"?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xsp="http://apache.org/xsp"
   xmlns:xsp-lib="http://mf.edu/lib/xsp/cttm"
   xmlns:session="http://apache.org/xsp/session/2.0"
>
```

Damit kann das "Session" Objekt benutzt und darauf ein Session Management aufgebaut werden. Um eine möglichst einfache und auch optimal wartbare Entwicklung zu ermöglichen, wurden sämtliche Funktionen, die im Zusammenhang mit dem Session Management stehen, in ein separates File in Form eines XSL Stylesheets ausgelagert und die einzelnen Funktionen danach vom eigentlichen XSP File als Logicsheet aufgerufen.[2]

```
<?xml version="1.0"?>
<?xml-logicsheet href="cttm/www2003/lib/lib.xsl"?>
```

Sämtliche Funktionen der Datei *lib.xsl* sind in das aktuelle File eingebunden und können im späteren Verlauf aufgerufen werden. Zwei Funktionen sind von besonderer Bedeutung: Die Erste ist die Initialisierungsfunktion *xsp-lib:init*, welche prüft, ob ein Benutzer sich bereits in einer Session befindet und gegebenenfalls die Benutzerdaten zur Verfügung stellt.

```
<xsl:template match="xsp-lib:init">
   <!-- initialisation procedure -->
   <xsp:logic>
```

[1] Files, welche Session Management enthalten sollen, müssen in der Cocoon Pipeline als Serverpages (XSP) deklariert werden.

[2] Ein Logicsheet ist ein Filter in Form eines XSLT Stylesheets, welches benutzerdefinierten Markup in Programmcode übersetzen kann. Logicsheets ermöglichen die Trennung der Logik von Präsentation und Inhalt.[Bay02]

```
      <!-- set the session/user settings -->
      if ( <session:get-attribute name="user"/> != null )
         <user><session:get-attribute name="user"/></user>
         <userid><session:get-attribute name="userid"/></userid>
         <user_level><session:get-attribute name="user_level"/></user_level>
       else
         <notlogged/>

   </xsp:logic>
</xsl:template>
```

Der Aufruf der Funktion wird in der XML Datei aus dem Tag *<sess>* gemacht:

```
<xsp:page ...
   ...
   <sess>
      <xsp-lib:init/>
   </sess>
</xsp:page>
```

Nach erfolgreichem Login wird die Session-Variable *user* gesetzt. Wird nun diese von der *init* Funktion gefunden, wird das *<sess>* Tag mit den Userdaten *<user>* (Username), *<userid>* (eindeutige ID des Benutzers) und *<user_level>* (Mächtigkeit des Benutzers: Normal / Administrator) gefüllt:

```
<sess>
   <user>dret</user>
   <userid>123</userid>
   <user_level>10</user_level>
</sess>
```

Ist der Benutzer allerdings noch nicht eingeloggt, ist die Session-Variable *user* nicht gesetzt. Folglich wird das *<sess>* Tag nur mit *<notlogged/>* gefüllt. Der aktuelle Benutzer ist somit noch anonym und kann deshalb keinen persönlichen Konferenzplan benutzen.

Die zweite wichtige Funktion ist *xsp-lib:login*. Diese wird aufgerufen, wenn der Benutzer sich ein- oder ausloggen will. Ist der Benutzer beim Aufruf dieser Funktion bereits eingeloggt, so hat er den Menupunkt “Logout” aufgerufen. Folglich müssen vorher gesetzte Session-Variablen gelöscht und das Tag *<logout/>* ausgegeben werden. Ist der Benutzer noch nicht eingeloggt, wird nur das XML Tag *<login/>* zurückgegeben. Die eigentliche Loginprozedur wird in Kapitel 6.2.4 beschrieben.

```
<xsl:template match="xsp-lib:login">
   <!-- initialisation procedure -->
   <xsp:logic>
   <!-- set the session/user settings -->
      if ( <session:get-attribute name="user"/> != null )
         <user><session:get-attribute name="user"/></user>
         <session:remove-attribute name="user"/>
         <session:remove-attribute name="userid"/>
         <session:remove-attribute name="user_level"/>
         <logout/>
       else
         <login/>

   </xsp:logic>
</xsl:template>
```

Abschliessend muss darauf hingewiesen werden, dass es sich beim *session* Namespace um eine XSP Standardbibilothek handelt, die Funktionen wie *session:remove-attribute* oder auch *session:get-attribute* zur Verfügung stellt. Deshalb muss beim Pipelining in der Sitemap angegeben werden, dass es sich dabei um eine XSP Datei handelt, welche zuerst durch den XSP Preprocessor bearbeitet werden muss. Da sämtliche CTTM XML Dateien das Session Management verwenden, müssen diese zu Beginn eines Pipelinings als XSP Dateien deklariert werden. In der Sitemap stellt man dies folgendermassen sicher:

```
<map:match pattern="cttm/www2003/*.html">
   <map:generate src="cttm/www2003/1.xml" type="serverpages"/>
```

Das generierte XML File wird hier als Typ *serverpages* deklariert und somit zuerst noch durch den XSP Preprocessor abgearbeitet, welcher die Session-Funktionen ausführen kann.

### 5.2.2 Public Pages

Für alle Benutzer zugängliche, statische Seiten wie die Begrüssungs- und Konferenzübersichtseite wurden ebenfalls mit Session Management versehen. Ein nicht angemeldeter Benutzer merkt davon nichts, für eingeloggte Besucher jedoch können so weiter Menupunkte wie “MyPlan” angezeigt werden. Ursprünglich war geplant, neben den jetzt vorhandenen statischen Seiten weitere mit zusätzlichen Informationen über die Konferenz und deren Umfeld zu erstellen. Aufgrund der beschwerlichen Kommunikation mit den Organisatoren musste jedoch dem Konferenzplan Vorrang gegeben werden. Die Einbindung weiterer statischer Seiten ist jedoch weder eine konzeptionelle, noch eine programmiertechnische Herausforderung.

### 5.2.3 Konferenzprogramm

Der wichtigste Teil der öffentlichen Seiten ist das Konferenzprogramm. In CTTM ist es hierarchisch implementiert. Ausgehend von der Übersicht der gesamten Veranstaltung kann man die einzelnen Events anwählen. So werden auf der Ebene der einzelnen "Tracks" die dazugehörigen "Talks" mit Infos wie zum Beispiel einer Liste der "Presenters" aufgelistet. Zu jedem Event werden entsprechende Informationen angezeigt, die dem Besucher helfen sollen, sich seinen Plan zusammenzustellen.

In einem ersten Schritt wurde das Datenbankmodell erweitert und die für die WWW2003 Konferenz benötigten EventTypes "Tutorials", "Workshops", "Plenary Sessions", "Paper Tracks", "Posters", "Developers Day" und deren Subtypen definiert.

Der zweite Schritt zur Implementation des Konferenzprogramms sollte sich als erste Hürde herausstellen. Nachdem das Datenbankmodell feststand, musste es mit den eigentlichen Daten des offiziellen Programms gefüllt werden. Nach mehrmaligem Nachfragen wurde uns von den Veranstaltern schliesslich ein XML File mit den Daten zugesandt. Dieses war leider nicht wohlgeformt und voller sowohl struktureller als auch inhaltlicher Fehler. Nur mit grossem Aufwand und einem Behelfsscript in PHP war es möglich, diese Daten zu extrahieren und die Datenbank initial zu füllen.

#### 5.2.3.1 SQL Abfragen

Wählt ein Besucher aus der Übersicht einen Eventtyp aus, wird über die URL der Request Parameter *id* mit der Event-ID übermittelt. In *schedule.xml* werden daraufhin in mehreren SQL Abfragen die nötigen Daten aus der Datenbank gelesen. Dabei muss unterschieden werden, ob der Benutzer angemeldet ist oder nicht. Ist er angemeldet, müssen zusätzlich seine bereits im MyPlan befindlichen Events geladen werden, um ihm diese schliesslich visuell im kompletten Konferenzplan hervorzuheben. Im Folgenden werden die einzelnen Queries kurz beschrieben, beginnend mit denjenigen für anonyme Benutzer.

Mit der Query[3]

```
SELECT event.*, 0 as selected, 0 as active, eventTypes.*
FROM event, eventTypes
WHERE event.eventType_ID = eventTypes.eventType_ID
AND event.event_id = '<xsp-request:get-parameter name="id"/>'"
```

[3]Die Felder *selected* und *active* werden nur im Zusammenhang mit dem personalisierten Konferenzplan benutzt und werden deshalb für anonyme Benutzer auf *0* gesetzt.

werden alle Informationen zum aktuellen Event aus der Datenbank gelesen. Um eine hierarchische Navigation zu ermöglichen, müssen zudem alle Sub-Events sowie der Parent-Event des aktuellen Events ausgelesen werden. Die folgende Query liest alle Sub-Events aus der Datenbank:

```
SELECT event.*, 0 as selected, 0 as active, eventTypes.*,
   child.event_id, child.name, child.initials
FROM event
   LEFT JOIN event as child ON event.event_id = child.parent_id,
   eventTypes
WHERE event.eventType_Id = eventTypes.eventType_ID
AND event.parent_id = '<xsp-request:get-parameter name="id"/>'
ORDER BY event.initials, event.name, child.sdate, child.stime,
   child.initials, child.name
```

Ist der Benutzer angemeldet, werden zusätzlich seine selektierten Events markiert. Die Relation User - Event ist in der Tabelle *user2event* gespeichert. In den Queries wird nun geprüft, ob für den aktuellen Event einen Eintrag in dieser Tabelle existiert. Ist dies der Fall, wird die Column *selected* auf *1*, ansonsten auf *0* gesetzt. Zusätzlich wird geprüft, ob der Event komplett selektiert wurde, d.h. ob alle Sub-Events ebenfalls vom Benutzer in seinen MyPlan übernommen wurden. Die Column *active* ist *1*, falls alle Sub-Events im MyPlan vorhanden sind. Die Query

```
SELECT event.*, NOT ISNULL(user2event.user_id) as selected,
   user2event.active as active, eventTypes.*
FROM event
   LEFT JOIN eventTypes ON event.eventType_Id = eventTypes.eventType_ID
   LEFT JOIN user2event ON event.event_id=user2event.event_id
      AND user2event.user_id='<session:get-attribute name="userid"/>'
WHERE event.eventType_ID = eventTypes.eventType_ID
AND event.event_id = '<xsp-request:get-parameter name="id"/>'"
```

liest den aktuellen Event aus und setzt die beiden Columns *selected* und *active* gemäss dem MyPlan des aktuellen Benutzers. Analog für die weiteren benötigten Queries.

SQL Abfragen werden in Cocoon durch den XSP E-SQL Namespace [Coc02] realisiert. Dieser stellt zu SQL-92 kompatible Funktionen zur Verfügung, welche unter anderem die von CTTM benötigten Funktionen SELECT, INSERT und DELETE implementieren (siehe auch Kapitel 6.3). Die aus den Queries erhaltenen Daten werden als XML eingegliedert:

```
<esql:row-results>
   <curEvent>
      <event_id><esql:get-string column="event.event_id"/></event_id>
      <initials><esql:get-string column="event.initials"/></initials>
      <name><esql:get-string column="event.name"/></name>
      <content><esql:get-string column="event.content"/></content>
      <sdate><esql:get-string column="event.sdate"/></sdate>
      ...
      <children>
         <child>
            <xsp:attribute name="id">
               <esql:get-string column="event.event_id"/>
             </xsp:attribute>
             <event_id><esql:get-string column="event.event_id"/></event_id>
             <initials><esql:get-string column="event.initials"/></initials>
             <name><esql:get-string column="event.name"/></name>
             ...
         </child>
         ...
      </children>
      <parent>
         ...
      </parent>
   </curEvent>
</esql:row-results>
```

Damit stehen nun die Rohdaten aus der Datenbank zum aktuellen Event, dessen Parent und allen seinen Sub-Events für die weitere Verarbeitung zur Verfügung.

#### 5.2.3.2 XML Transformation nach HTML durch XSLT

Die in XML vorliegenden Informationen werden in einem zweiten Schritt per XSL Transformation in die eigentliche HTML Seitendarstellung überführt. So wird beispielsweise in *schedule_layout.xsl* mittels des Tags <*parent_id*> und dem entsprechenden *name*> Tag folgendermassen eine Menuführung zusammengesetzt: Aus

```
<tr><td class="nav" colspan="2">
<p>&raquo;
   <xsl:choose>
      <xsl:when test="(parent_id != 0)">
         <a>
            <xsl:attribute name="href">schedule.html?id=
               <xsl:value-of select="parent_id"/>
            </xsl:attribute>
            <xsl:value-of select="parent/name"/>
         </a>
      </xsl:when>
      <xsl:otherwise>
         <a href="overview.html">Overview</a>
      </xsl:otherwise>
```

```
   </xsl:choose>
    &raquo; <xsl:value-of select="name"/>
</p>
</td></tr>
```

wird beispielsweise für den aktuellen Event mit der ID 232 der folgende HTML Code erzeugt:

```
<tr><td class="nav" colspan="2">
   <p>
     &raquo; <a href="schedule.html?id=23">Morning Tutorial</a> 
     &raquo; Introduction to Scalable Vector Graphics (SVG)
   </p>
</td></tr>
```

Analog wird die gesamte XML Struktur durch das XSL *schedule_layout.xsl* in HTML überführt. Dabei wird zur Darstellung zusätzlich CSS verwendet, um das XSL möglichst schlank zu halten.

### 5.2.4 Registration und Login

Der zunächst anonyme Besucher der Seite kann sich durch ein Login anmelden. Ist er noch nicht registriert, kann durch eine einfache Registrationsmaske ein neuer Benutzeraccount erstellt werden.

#### 5.2.4.1 Registration

Der neue Benutzer kann sich auf der Registrationsseite neu registrieren. Die vom Benutzer getätigten obligatorischen und freiwilligen Angaben werden mittels der von Cocoon bereitgestellten Form Validator Funktion auf ihre Korrektheit überprüft. Der Form Validator kann die Eingabefelder nach regulären Ausdrücken überprüfen. Somit können ohne grossen programmiertechnischen Aufwand Emailadressen oder Telefonnummern auf ihre syntaktische Korrektheit überprüft werden. Für weitere Informationen zur technischen Implementierung eines Form Validators wird hier auf die Dokumentation der bestehenden Lösung verwiesen.

Sind die Eingaben syntaktisch korrekt und der gewählte Loginname noch nicht in der Datenbank vorhanden, werden die Informationen in die Datenbank geschrieben und der Besucher kann sich zukünftig auf der Seite einloggen.

#### 5.2.4.2 Login

Beim Login wird ein Formular bereitgestellt, welches den Benutzernamen sowie das Passwort verlangt. Diese Werte werden anschliessend in der Datenbank gesucht und

bei einer positiven Übereinstimmung die Einträge für *user*, *userid* und *user_level* in das Session Objekt getätigt.

```
<sess>
   <xsp:logic>
      <!--   check if login is valid -->
      <esql:connection>
         <esql:pool>saconf</esql:pool>
         <esql:execute-query>
            <esql:query>
               SELECT * FROM user
               WHERE username='<xsp-request:get-parameter name="user"/>'
               AND password='<xsp-request:get-parameter name="password"/>'
            </esql:query>
            <esql:results>
               <esql:row-results>
                  <session:set-attribute name="user">
                     <xsp-request:get-parameter name="user"/>
                  </session:set-attribute>
                  <session:set-attribute name="userid">
                     <esql:get-string column="user_id"/>
                  </session:set-attribute>
                  <session:set-attribute name="user_level">
                     <esql:get-string column="userlevel"/>
                  </session:set-attribute>
               </esql:row-results>
            </esql:results>
            <esql:no-results>
               <notloggedin/>
            </esql:no-results>
         </esql:execute-query>
      </esql:connection>
   </xsp:logic>
   <xsp-lib:init/>
</sess>
```

Bei erfolgreicher Anmeldung ist das Session Objekt mit den entsprechenden Einträgen gesetzt. In der Folge kann wie in Abschnitt 5.2.1 beschrieben die Gestaltung der Seite angepasst und der persönliche Konferenzplan über "MyPlan" zugänglich gemacht werden.

### 5.2.5 Persönliches Konferenzprogramm

Im persönlichen Konferenzprogramm des angemeldeten Benutzers befinden sich die vom Benutzer ausgewählten Event. Er kann sich so aus dem vielfältigen Angebot der Konferenz diejenigen Veranstaltungen heraussuchen, die ihn interessieren. Ist der Benutzer angemeldet, kann er den aktuellen Event des Konferenzprogramms in seinen MyPlan hinzufügen oder diesen aus dem MyPlan entfernen, falls er bereits früher ausgewählt wurde.

#### 5.2.5.1 Übersicht

Das persönliche Konferenzprogramm ist eine Zusammenstellung der vom Benutzer ausgewählten Events aus der kompletten Konferenz. Auf dieser Seite werden dem Besucher die Events nach Kategorien und Zeit geordnet präsentiert.

Mit Hilfe der Übersichtsseite soll dem Benutzer geholfen werden, sich auf einer Konferenz mit Unmengen von unterschiedlichsten Events zurechtzufinden. Alle in der Datenbank vorhanden Einträge werden nach Eventtypen und Datum sortiert in einer übersichtlichen Zusammenfassung präsentiert.

#### 5.2.5.2 Einfügen und Entfernen

Bei jedem Einfügen eines Events wird ein Eintrag für den entsprechenden Benutzer und den selektierten Event in der Tabelle *user2event* erstellt. Zusätzlich wird bei allen hierarchisch über dem Event liegenden Oberevents nachgeführt, ob deren Sub-Events alle selektiert sind. Dies ist nötig um dem Benutzer visuell aufzuzeigen, ob ein Event komplett oder nur teilweise ausgewählt ist. Der Benutzer hat zudem die Möglichkeit, alle Sub-Events ebenfalls automatisch hinzufügen zu lassen. Dazu müssen zusätzlich rekursiv für alle Sub-Events die Einträge in *user2event* erstellt werden.

Beim Entfernen eines Events aus dem MyPlan wird analaog vorgegangen. In diesem Fall muss der Selektionsstatus aller Oberevents neu gesetzt werden.

Die Lösung über Attribute wurde aus Performancegründen gewählt. Müssten bei jeder Ansicht des Konferenzplaners die Selektionsstati der Events neu berechnet werden, wären massiv mehr Datenbankanfragen von Nöten. Es kann davon ausgegangen werden, dass eine View einer Seite des Konferenzprogramms häufiger auftritt, als das Einfügen eines Events in den persönlichen MyPlan.

### 5.2.6 Persönliches Datenblatt

Der Benutzer kann seine bei der Registrierung eingegeben persönlichen Daten einsehen und gegebenenfalls korrigieren oder ergänzen. Dazu wurde eine weitere persönliche Seite eingerichtet, welche durch Klick auf den Loginnamen im Menu aufgerufen werden kann.

Das persönliche Datenblatt ist analog zur Registrierseite aufgebaut. Der einzige Unterschied besteht darin, dass die einzelnen Eingabefelder bereits zu Beginn mit den vorhandenen Daten aus der Datenbank gefüllt werden. Danach wird wieder nach demselben Schema vorgegangen. Die eingegebenen Daten werden mittels Formvalidator auf ihre Gültigkeit getestet und bei erfolgreichem Check in die Datenbank geschrieben.

### 5.2.7 Administration Sites

Für die Wartung des Konferenzplaners wurde ein Administrationsbereich eingerichtet. Dieser ermöglicht Administratoren über Editiermasken Events und Benutzer zu editieren sowie Email Notifications über Änderungen zu verschicken. Um Zugang zu den Administrationsseiten zu haben, muss der Benutzer einen Userlevel höher 10 aufweisen. Es erscheint ein zusätzlicher Menupunkt “Admin”.

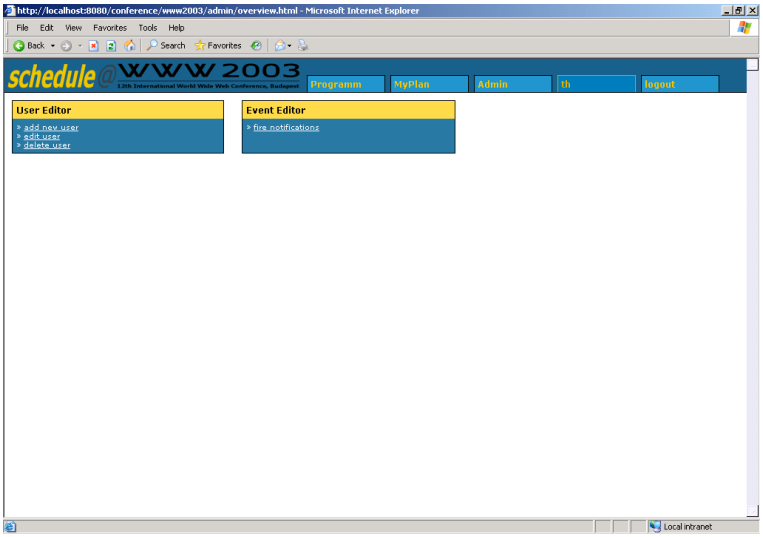



Abbildung 5.8: Administrations Interface des CTTM

#### 5.2.7.1 Editieren von Events

Hat der Benutzer Administrationsrechte, erscheint im Konferenzplan beim Betrachten der einzelnen Events ein Button “Edit”. Mittels dessen gelangt der Benutzer zur Editieransicht des aktuellen Events. Sämtliche Event-Daten können verändert und anschliessend gespeichert werden. Die Felder “Name”, “Time”, “Date”, “Presenter” und/oder “Room” haben zusätzlich eine “notify” Checkbox. Wird diese angewählt, werden sämtliche Benutzer, die diesen Event in ihrem persönlichen Konferenzplaner haben, per Email über die Änderungen benachrichtigt. Wertet der Administrator die vorgenommenen Änderungen als nicht gravierend, kann die Checkbox leer und die Benutzer unbenachrichtigt gelassen werden. Das tatsächliche Versenden der Emails muss, wie im nächsten Abschnitt erklärt, manuell gestartet werden.

#### 5.2.7.2 Versenden von Änderungshinweisen per Email

Wird CTTM rege benutzt und führen viele Benutzer ihren persönlichen Konferenzplan, so kann das automatische Versenden der Änderungshinweise per Email

einige Zeit beanspruchen. Aus diesem Grund werden die Emails nicht direkt nach dem Ändern der Events verschickt. Erst durch Anwählen des Eintrages "fire notifications" im Administrationsinterface wird das tatsächliche Versenden ausgelöst. Dadurch kann der Administrator einige Events nacheinander editieren, ohne jeweils auf das Ende des Emailversandes warten zu müssen.

In der Tabelle *tonotify* werden beim Editieren die Benutzer vermerkt, denen Hinweise versandt werden müssen. Beim Betätigen des "fire notifications" Eintrags werden sämtliche Einträge in dieser Tabelle abgearbeitet und jeweils an die Emailadresse die dazugehörige Nachricht geschickt.

Zum Verschicken von Emails verwendet der Konferenzplaner eine Sendmail Bibliothek. Nach dem Hinzufügen der Bibliothek in den Namespace der Applikation kann auf einfache Weise eine Email versandt werden. Die Bibliothek benötigt einzig einen für CTTM offenen SMTP Server, über welchen die Emails[4] verschickt werden.

```
<sendmail:send-mail>
   <sendmail:charset>ISO-8859-1</sendmail:charset>
   <sendmail:from>ceo@w3c.org</sendmail:from>
   <sendmail:to><esql:get-string column="email"/></sendmail:to>
   <sendmail:subject>Change of Conference plan</sendmail:subject>
   <sendmail:smtphost>mail.ethz.ch</sendmail:smtphost>
   <sendmail:body>
      <esql:get-string column="message"/>
   </sendmail:body>
</sendmail:send-mail>
```

Nach dem Versenden der Nachrichten werden die Einträge aus der Tabelle *tonotify* gelöscht.

#### 5.2.7.3 Editieren von Benutzern

Die Editieransicht für registrierte Benutzer ist identisch mit derjenigen, die die einzelnen Benutzer selbst besitzen. Als einziger Unterschied wird hier das Feld "Userlevel" angezeigt, um einen Benutzer in ein bestimmtes Profil einteilen zu können. Normale Benutzer haben Userlevel 1, Administratoren Level 10. Beim Löschen eines Benutzers werden die betreffenden Einträge aus den Tabellen *user*, *user2event* und *tonotify* aus der Datenbank gelöscht.

### 5.2.8 Sitemap und Pipeline

Die Begriffe "Sitemap" und "Pipeline" und ihre Bedeutung werden ausführlich in der Dokumentation zur früheren CTTM Version erklärt. Für detaillierte Informationen

[4] Tangieren mehrere verschiedene Änderungen den Benutzer, so erhält er in der aktuellen Version auch mehrere Emails zugesandt.

wird hier deshalb auf diese verwiesen[5]. In diesem Kapitel soll allerdings der Aufbau und das Pipelining des Konferenzplaners veranschaulicht werden.

In der bestehenden Sitemap wurde eine neue Pipeline zum Abarbeiten der Anfragen für den Konferenzplaner eingefügt. Es wurde versucht, die Pipelines der einzelnen Dokumente so einheitlich wie möglich zu gestalten. Der normale Ablauf einer Transformation zu einem HTML Dokument sieht wie folgt aus:

```
<map:match pattern="cttm/www2003/*.html">
   <map:generate src="cttm/www2003/1.xml" type="serverpages"/>
   <map:transform src="cttm/www2003/1.xsl">
      <map:parameter name="use-request-parameters" value="true"/>
   </map:transform>
   <map:serialize type="html"/>
</map:match>
```

Ausgangspunkt ist das Generieren eines XML Files des Typs *serverpages*. Dadurch wird der XSP Preprocessor aktiviert, welcher das Ausführen von XSP ermöglicht. In einem zweiten Schritt transformiert die Pipeline das XML Dokument über XSL Transformations zu einer XHTML Seite, welche schliesslich im dritten Schritt mit dem Typ *html* serialisiert wird. Um Daten, Transformation und Layout so gut wie möglich voneinander trennen zu können, wurde ähnlich zur früheren Lösung das Layout der HTML Seite in eine weitere Datei ausgelagert. Für die Generierung des HTML Dokuments sind also nicht nur die Dateien *beispiel.xml* sowie *beispiel.xsl* verantwortlich. Das eigentliche Layout mit CSS, Page Title, Menu und ähnlichem wurde in die Datei *layout/header.xsl* ausgelagert. Diese Datei bietet eine Template-Funktion *make_page* an, welche vom XSL File aufgerufen wird und als Parameter *page_content* den anzuzeigenden und vorher generierten Inhalt erhalten muss.

```
<xsl:import href="layout/header.xsl"/>
...
<xsl:variable name="page_content">
   <table border="0" cellpadding="0" cellspacing="0" align="center">
      <tr><td>
         <img src="/conference/www2003/img/welcome.jpg" border="0"/>
      </td></tr>
   </table>
</xsl:variable>
...
<xsl:call-template name="make_page">
   <xsl:with-param name="login" select="$login"/>
   <xsl:with-param name="user" select="$user"/>
   <xsl:with-param name="page_content" select="$page_content"/>
</xsl:call-template>
...
```

[5] [KW03] Kapitel 1.2.4.6 (Seite 12ff.)

Oft erwies es sich als schwierig, die Transformation der XML Daten vom eigentlichen Transformieren des Inhaltes nach HTML zu trennen. Da beispielsweise die Darstellungsweise der Event-Auflistung teilweise über die in der Datenbank definierten EventTypes bestimmt wird, müssen aufgrund der Übersichtlichkeit bereits beim Transformieren der XML Daten kleine HTML Teile eingeflochten werden. So oft wie möglich wurden aber darstellungsspezifische Informationen in eine Datei *beispiel_layout.xsl* ausgelagert.

Ausnahmen beim Pipelining stellen Seiten dar, welche den Cocoon eigenen Formvalidierungsmechanismus benötigen. Die genaue Funktionsweise dieses speziellen Pipelinings wurde bereits in der Dokumentation der bestehenden Lösung beschrieben[6].

[6] [KW03] Kapitel 5.4.1 (Seite 62ff.)

# Kapitel 6

# Herausforderungen

Obwohl die neue CTTM Version auf der vorausgegangenen aufbauen konnte, tauchten im Verlaufe der Arbeit einige Schwierigkeiten und unerwartete Hindernisse auf. In diesem Kapitel werden diese Punkte näher erläutert, um einer allfälligen Arbeit zu einem ähnlichen Thema das neu gewonnene Wissen weitergeben zu können.

## 6.1 Konferenzbeginn

Das Hauptproblem stellte der Zeitdruck dar. Schon vor Beginn der Arbeit stand fest, dass nur rund fünf bis sechs Wochen zur Verfügung stehen würden, um CTTM auf die WWW2003 realisieren zu können. Die Anmeldung des Projekts bei der WWW2003 war bereits getätigt; es sollte in einem Poster präsentiert werden und den Konferenzbesuchern gleichzeitig als einsatzfähiger Planer ihrer besuchten Veranstaltungen zur Verfügung stehen.

Die Aufgabenstellung der Semesterarbeit wurde dahingehend angepasst, dass zuerst die Grundfunktionalität implementiert und diese später je nach Zeitbudget mit verschiedenen Zusatzfunktionen ergänzt werden sollte. Tatsächlich reichte die Zeit nicht aus, um den kompletten Satz an Features implementieren zu können. Allerdings wurden die wichtigsten Funktionen implementiert, so dass weitere Features mit relativ geringem Aufwand in das CTTM eingefügt werden können.

## 6.2 Informationsfluss

Ein sehr projektspezifisches Problem war direkt von Beginn weg die Kommunikation mit den Ansprechpartnern der WWW2003. Da das Projekt nur etwas über einem Monat vor Konferenzbeginn lanciert wurde, waren die Veranstalter selber schon zu sehr mit der Organisation beschäftigt; unsere Wünsche und Bedürfnisse waren nicht immer von erster Priorität. Es war beschwerlich nur schon die allerwichtigsten Informationen wie die kompletten Daten des Konferenzprogramms an sich von den Veranstaltern zu bekommen. Die schliesslich mit einiger Verspätung gelieferten

Daten waren zudem in einem schlechten Zustand, sodass das Parsen und Einlesen zusätzlich erschwert wurde. Aus diesem Grund musste viel Zeit in das Konvertieren der Daten gesteckt werden. Auf den Ausbau von allgemeinen Informationen über die WWW2003 wurde schliesslich mangels Informationen ganz verzichtet.

## 6.3 Dynamische SQL Abfragen in Cocoon/XSP

SQL Abfragen werden in Cocoon über den XSP E-SQL Namespace realisiert. Dieser muss zu Beginn der XSP Seite deklariert werden:

```
<?xml version="1.0"?>
<?xml-logicsheet href="cttm/www2003/lib/lib.xsl"?>
<xsp:page language="java"
   xmlns:xsp="http://apache.org/xsp"
   xmlns:xsp-lib="http://mf.edu/lib/xsp/cttm"
   xmlns:esql="http://apache.org/cocoon/SQL/v2"
   xmlns:session="http://apache.org/xsp/session/2.0"
   xmlns:xsp-request="http://apache.org/xsp/request/2.0"
>
```

So kann nun über den *esql*-Präfix auf die E-SQL Funktionalität zugegriffen werden:

```
<esql:connection>
   <esql:pool>saconf</esql:pool>
   <esql:execute-query>
   <esql:query>
      SELECT * FROM user
      WHERE username='<xsp-request:get-parameter name="user"/>'
         AND password='<xsp-request:get-parameter name="password"/>'
   </esql:query>
   <esql:results>
      <esql:row-results>
      ...
```

Will man nun aber dynamisch zusammengesetzte Queries verwenden, kann man das SQL Statement über *<xsp:logic>* und *<xsp:expr>* zusammenfügen:

```
...
<xsp:logic>
   if (<esql:get-boolean column="et_showDate"/>)
      querySort = "sdate, stime, ";
    else if (<esql:get-boolean column="et_showTime"/>)
      querySort = "stime, ";
```

```
    else
     querySort = "";

</xsp:logic>
<children>
<esql:execute-query>
   <esql:query>
      <xsp:expr>
         "SELECT * FROM events WHERE event.parent_id = '" +
         <xsp-request:get-parameter name="id"/> +
         "' ORDER BY " + querySort +
         "event.initials, event.name"
      </xsp:expr>
   </esql:query>
   <esql:results>
      <esql:row-results>
      ...
```

## 6.4 Rekursive Datenbankabfragen in Cocoon/XSP

Bei der Selektion bzw. Deselektion eines Events müssen im *myPlan/change.xml* rekursiv alle dessen Vater- und Kinder-Events ebenfalls behandelt werden. Hier stösst allerdings die XSP Implementation von Cocoon an ihre Grenzen. So können zwar Funktionen definiert und ausgeführt werden, innerhalb dieser aber keinerlei Cocoon XML verwendet oder generiert werden. Damit wird rekursives Programmieren erheblich erschwert. Um dennoch alle nötigen Events auswählen zu können, musste erst eine lineare String-Liste aller involvierten Events über Schleifen generiert werden. Über temporäre String-Variablen konnte eine Pseudo-Rekursion erreicht werden:

In einem ersten Query werden alle Children eines Events selektiert, bearbeitet und deren ID in einem String konkateniert. Dieser String wird wiederum in einer Schleife abgearbeitet: Für jede ID im String wird über ein weiteres Query dessen Children gesucht, bearbeitet und deren ID an den temporären String angehängt. Dies wird solange ausgeführt, bis dieser temporäre String leer ist.

Analog können alle Parents gefunden werden.

## 6.5 Informationsknappheit im Internet

Obwohl Cocoon nun schon seit mehreren Jahren entwickelt wird und bereits in der zweiten Version vorliegt, sind kaum brauchbare Anleitungen, SDKs und Beispiele erhältlich. Die vorhandenen Cocoon Projekte beschränken sich auf Grundlagenforschung in engen Gebieten. Obwohl die Entwicklung und Performance von Cocoon

seit Version 2.0 deutlich verbessert wurde, kommt es scheinbar nur selten zu einem tatsächlichen Einsatz und hat dementsprechend nur eine kleine Benutzergruppe. Selbst die Standarddokumentation von Cocoon weist noch zahlreiche offene Fragen und Unklarheiten auf. Da Cocoon zudem keine Debugging-Umgebung aufweist, war die Problem- und Lösungssuche streckenweise äusserst mühsam.

# Kapitel 7

# Offene Probleme

Nach beendeter Arbeit sind einige Punkte noch nicht in der entstandenen Lösung integriert. Hierbei handelt es sich um kleinere Aspekte der Aufgabenstellung, welche aus Zeitgründen nicht mehr realisiert werden konnten.

## 7.1 Datenmodell Probleme/Verbesserungen

Die Modellierung der Datenbank aus der bestehenden Lösung ist zwar funktional und wurde grösstenteils für die neue Version von CTTM so übernommen, allerdings gibt es Möglichkeiten, das Modell zu verbessern, um dadurch Vorteile beim Unterhalten und Wartung der Datenbank und der Konferenz an sich erhalten zu können:

Zurzeit ist jeder Event als einzelner Eintrag in die Datenbank eingefügt. Die einzige Verlinkung der Events untereinander besteht vom *parent_id* Wert zum Vaterevent sowie zum *eventtype*. Auf Attribute wie Zeit und Ort sind die einzelnen Einträge völlig autonom. Eine verbesserte Modellierung könnte es erlauben, mehrere Events zu gruppieren. So könnten beispielsweise alle Events, die im Saal A an einem Morgen hintereinander stattfinden zu einer Gruppe zusammengefasst werden. Würde sich nun der Beginn eines Events beispielsweise durch Überziehen der Zeit des Vorgängerevents verschieben, würden die Startzeiten aller nachfolgender Events automatisch um diese Zeit nach hinten verschoben, ohne dass jedes einzelne Event angepasst werden müsste.

## 7.2 SMS Integration

Geplant war nicht nur die Änderungsnotifikation per Email, sondern generell ein Ausbau auf mobile Dienste. Per SMS könnten kurzfristige Änderungen den entsprechenden Teilnehmer auf ihre Mobiltelefone geschickt werden. Denkbar wäre auch eine generelle Erinnerung an bevorstehende Events und an ihre Durchführungsorte ("Ihr nächster Event beginnt um 13:30 im Raum D").

Da im Prinzip das Versenden von SMS analog zu demjenigen der Emails gesehen werden kann, könnte zumindest die Änderungsnotifikation leicht auf SMS ausgeweitet werden. Spezielle Beachtung muss dann jedoch den limitierten Meldungsgrössen pro SMS sowie eventuellen Konflikten bei unterschiedlichen Standards und Sprachen gewisser Länder gewidmet werden.

## 7.3 Optimierung des User Interface Designs

Dem User Interface Design des CTTM wurde nur geringe Beachtung geschenkt und es ist deshalb nicht ausgereift. Um eine komplette Lösung vorzeigen zu können, muss es überarbeitet werden. Das vorhandene Layout basiert auf den Standards HTML und CSS und kann nach Bedarf angepasst werden.

## 7.4 Optimierung des persönlichen Konferenzprogramms

Das persönliche Konferenzprogramm beinhaltet viele Optimierungsmöglichkeiten. Es könnte in einzelne Tage unterteilt werden; Such- und Sortierfunktionen fehlen. Ebenfalls denkbar wären verschiedene Ansichtsarten des Programms, beispielsweise per WAP oder Export nach PDF / Email.

## 7.5 Erweiterung des Administrationsinterfaces

Für ein produktives Arbeiten muss das Administrationsinterface ausgebaut werden. Die Funktionen zum Erstellen neuer Events sind nicht komplett implementiert. Änderungen an Event-Types müssen zurzeit direkt in der Datenbank getätigt werden.

# Kapitel 8

# Schlusswort

Die vorliegende Semesterarbeit ermöglichte uns einen umfassenden Einblick in eine aufstrebende Technologie. Dass das Produkt schliesslich auch tatsächlich an der WWW2003 Verwendung fand und zudem in einem Poster präsentiert wurde, machte die Arbeit sehr interessant.

Bei der Erweiterung des in der vorangegangen Diplomarbeit entstandenen Toolkits konnten wir von der Vorarbeit der Diplomanden profitieren. Besonders zu Beginn war es hilfreich auf eine schon existierende Lösung zurückgreifen zu können, obschon kaum Codefragmente direkt in den CTTM einfliessen konnten.

Als schwierig gestaltete sich die Kommunikation zu den Veranstaltern der WWW2003. Wichtigste Voraussetzung für diese Arbeit war natürlich, dass uns die benötigten Informationen zur WWW2003 zur Verfügung standen. Wichtigste Information war das Konferenzprogramm, aber auch weitere Informationen über die Veranstalter oder die Konferenz, hätten wir gerne in den CTTM eingebaut. Schliesslich konnten wir glücklich sein, ein Konferenzprogramm in mehr oder minder gutem Zustand zu erhalten. Auf weitere Infos mussten wir leider verzichten.

Trotz Zeitdruck mussten wir uns ausführlich mit den grundlegenden Mechanismen von Cocoon befassen. Da gute Referenzbeispiele fehlten und die Cocoon-Dokumentation mangelhaft war, musste viel Zeit mit selbstständigem Experimentieren verbracht werden. Es gelang uns jedoch die anfänglichen Probleme überwinden und rechzeitig auf Konferenzstart den Planer aufzuschalten.

Herzlichen Dank gebürt unserem Betreuer Prof. Erik Wilde, der uns über die gesamte Dauer der Arbeit unterstützte. Die Zusammenarbeit war sehr angenehm und inspirierend.

# Literaturverzeichnis


[Apa03a] Apache. *The Apache HTTP Server Project*. The Apache Software Foundation, 1992-2003. `http://httpd.apache.org/`.

[Apa03b] Apache. *The Apache Cocoon Project*. The Apache Software Foundation, 1999-2003. `http://cocoon.apache.org/`.

[Apa03c] Apache. *The Jakarta Site: Apache Tomcat*. The Apache Software Foundation, 1999-2003. `http://jakarta.apache.org/tomcat/`.

[Bay02] Thomas Bayer. *Cocoon Logicsheet Tutorial*. Orientation in Objects GmbH, 2002. `http://www.oio.de/public/xml/logicsheet-tutorial.htm`.

[Coc02] Cocoon. *ESQL Taglib*. The Apache Software Foundation, 2002. `http://cocoon.apache.org/2.0/userdocs/xsp/esql.html`.

[Con03] World Wide Web Consortium. *The Twelfth International World Wide Web Conference 2003*. World Wide Web Consortium, 2003. `http://www2003.org/`.

[KW03] Beat Krähenmann and Martin Waldburger. *Toolkit für Konferenzprogrammverwaltung und -personalisierung*. ETH Zürich, 2003. Diplomarbeit.

[MyS03a] MySQL. *MySQL: The World's Most Popular Open Source Database*. MySQL AB, 1995-2003. `http://www.mysql.com/`.

[MyS03b] MySQL. *MySQL Connector/J*. MySQL AB, 2003. `http://www.mysql.com/products/connector-j/`.

[Sha01] Gal Shachor. *Working with mod_jk*. The Apache Software Foundation, 2001. `http://jakarta.apache.org/tomcat/tomcat-3.3-doc/mod_jk-howto.html`.

[Sun03] Sun. *The Java 2 Platform, Standard Edition*. Sun Microsystems Inc., 1995-2003. `http://java.sun.com/j2se/`.