Web-Plattform zum Informationsaustausch über wissenschaftliche Veranstaltungen

Bachelorarbeit

von

Gerald Schenke

aus

Düsseldorf

vorgelegt am

Lehrstuhl für Rechnernetze und Kommunikationssysteme
Prof. Dr. Martin Mauve
Heinrich-Heine-Universität Düsseldorf

Juni 2006

Betreuer:
Dipl.-Inf. Björn Scheuermann
Dipl. Wirtsch.-Inf. Christian Lochert

Anmerkungen

In erster Linie möchte ich mich bei meinen beiden Betreuern Björn Scheuermann und Christian Lochert bedanken, die mir beim Verfassen dieser Arbeit immer wieder mit Tipps und Vorschlägen geholfen haben. Vielen Dank.

Ein Dank gilt auch meiner Familie und Freunden – besonders Christoph Pälmer und Michael Willigens – für die Anregungen und die Unterstützung während der letzten drei Monate.

Außerdem möchte ich mich noch bei den Leuten aus dem IRC-Channel #rubyonrails im Freenode-Netzwerk bedanken, die mir viele kleinere Fragen rund um Rails beantworten konnten und mir sehr weitergeholfen haben.

Als Letztes bedanke ich mich bei David Heinemeier Hansson und seinem Entwicklerteam für das Framework RubyOnRails, mit dem ich wohl nicht das letzte Mal gearbeitet habe.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Zu jeder wissenschaftlichen Konferenz wird in der Regel ein Call For Papers (CFP) herausgegeben, mit dem Arbeiten zu dem Thema der Konferenz gesucht werden. Bei einem solchen CFP handelt es sich um einen Text mit vielen Informationen zu der Veranstaltung wie zum Beispiel Titel, Zeitpunkt, Ort, Einreichungstermine und so weiter. Verbreitet werden solche CFPs über Emails und kleinere Listen im Internet ("CFP-Listings"). Beispiele sind die Seiten von Alex Slingerland (Sli06), Wenyu Jiang (Jia06) und Dirk Stroobandt (Str06). Da viele kleine CFP-Listen existieren, gestaltet sich die Suche nach einer passenden Konferenz für die eigene Arbeit meist schwierig. Hinzu kommt, dass kaum eine CFP-Liste über eine Suchfunktion verfügt.

Die Sammlung der Informationen über Konferenzen an einer zentralen Stelle bietet sich zwar an, jedoch ist es für eine Person zu aufwendig nebenbei die Daten zu sammeln und übersichtlich darzustellen. Es gibt eine kommerzielle Lösung (Pap06), allerdings sprechen die Preise für sich. So soll man als Einzelperson 75 US-Dollar im Jahr für 10 erlaubte Logins pro Monat bezahlen. Möchte eine Universität diesen Service nutzen, muss sie 2000 US-Dollar im Jahr dafür bezahlen. Hier setzt die Arbeit an und entwickelt eine kostenlose Web-Plattform auf der sich die Suchenden selbst mit Informationen versorgen und die gewünschte Konferenz leicht zu finden ist.

Diese Plattform steht und fällt mit der Benutzerbeteiligung, weshalb eine aktive Teilnahme einfach und intuitiv möglich sein sollte. Eine zukünftige Zusammenarbeit mit Betreibern anderer CFP-Listings ist gewünscht, wenn nicht sogar notwendig.

1.2 Problemstellung

In dieser Arbeit soll eine Web-Plattform zum Informationsaustausch über wissenschaftliche Veranstaltungen entwickelt werden. Es soll möglich sein, auf einer Website Veranstaltungen zu suchen und einzutragen. Mit Veranstaltungen sind nicht nur wissenschaftliche Konferenzen gemeint, sondern auch Sonderausgaben wissenschaftlicher Zeitschriften (“Journals”) – so genannte “special issues”.

Die Möglichkeit für Besucher, auf der Website Informationen über Konferenzen oder Journals einzutragen, ist dabei substanziell. Die Informationen sollen an einem Ort gesammelt werden, aber von den Benutzern selbst eingepflegt und aktuell gehalten werden. Besonders für Veranstalter bietet sich an, ihre Veranstaltung einzutragen, um auf sie aufmerksam zu machen.

Der Hauptzweck besteht darin, die Einreichungstermine (“submission deadlines”) für Paper, Poster, Artikel und Ähnliches zu einer Konferenz oder einem Journal eines bestimmten Themas leicht zu finden und weitere Informationen über die Konferenz oder das Journal zu erhalten.

1.3 Aufbau

Im zweiten Kapitel dieser Arbeit werden die Grundlagen behandelt. Dort werden die verwendeten Programmiersprachen und das verwendete Web-Development-Framework “Ruby On Rails” sowie angewandte Konzepte vorgestellt.

Nach diesem einleitenden Kapitel beschäftigt sich das dritte Kapitel mit der entwickelten Applikation. Als erstes wird dort die Struktur für die Datenspeicherung ermittelt und das gewählte Datenbankschema sowie die wichtigen Grundpfeiler ‘Display’ und ‘un-

vollständige Datumsangaben' vorgestellt. Bevor dann die einzelnen Bereiche der Plattform behandelt werden, ist noch eine kurze Darstellung des gewählten Layouts und dessen Umsetzung im Framework enthalten. Das Ende des dritten Kapitels befasst sich mit der Eingabe von CFPs und der Ausgabe mit Hilfe der Listen und Suchparameter.

Das vierte und letzte Kapitel fasst das Gelesene kurz zusammen, stellt mögliche Erweiterungen vor und enthält abschließend einen Ausblick auf die Möglichkeiten der Plattform.

Kapitel 2

Grundlagen

In den vergangen Jahren hat sich Web-Development stark verändert. Am Anfang standen statische HTML-Seiten, die mit Hilfe von CGI-Skripten, PHP, perl etc. langsam immer dynamischer wurden. Allerdings bestanden solche Websites oft aus einer Ansammlung von Skripten, die schwierig zu warten und zu verändern waren. Anwendungen zu schreiben ist sehr aufwendig, wenn man die Grundfunktionalitäten komplett neu programmieren muss oder die oben erwähnten Skripte anpassen und verbinden will. Daher entstanden mehr und mehr Web-Development-Werkzeuge und Frameworks, um auch kompliziertere Web-Applikationen möglich zu machen. Diese bieten dem Programmierer viele Hilfen und Vorlagen an, um seine Website schneller und einfacher zu programmieren. Mittlerweile haben solche Frameworks – das beste Beispiel ist Ruby On Rails – eine Qualität erreicht, dass die Programmierung von komplexen Web-Applikationen beinahe so schnell und bequem möglich ist, wie bei einer Desktop-Anwendung.

2.1 Ruby

Ruby wurde von Yukihiro Matsumoto entwickelt, erschien zum ersten Mal im Jahr 1995 in Japan und ist eine echt objektorientierte und interpretierte Programmiersprache. *Echt objektorientiert* heißt, dass alle Operationen auf einem Objekt als Ergebnis wieder ein Objekt liefern. Oft wird dies von Programmiersprachen, die sich objektorientiert nennen, nicht eingehalten (Tho05). Ihre Wurzeln hat die Sprache in Smalltalk, Perl und Python.

Ruby besitzt eine einfache Syntax, eine Fehlerbehandlung wie in Java oder Python, einen "true mark-and-sweep garbage collector", kann zur Laufzeit Methoden zu Klassen oder Instanzen hinzufügen und hat noch viele weitere Features (Rub06). In dieser Bachelorarbeit wird die aktuelle Version ruby-1.8.4 benutzt.

2.2 Rails

Ruby On Rails ist ein Framework um Web-Applikationen schnell und einfach zu erstellen. Es ist in der Programmiersprache Ruby geschrieben.
Im Juli 2004 wurde das von David Heinemeier Hansson entwickelte Framework Rails erstmals vorgestellt. Als Hauptmerkmal wird immer wieder vorangestellt: "Rails Is Agile" (THH05). Rails ist also beweglich. Um zu verstehen was mit dem Wort agil gemeint ist, sollte man sich die Hauptpunkte des Agile Manifesto (Agi01) – eine Definition von agiler Software-Entwicklung – ansehen, auf den sich die Entwickler hierbei beziehen. Hier heißt es:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Mit Rails kann man also demnach schnell, einfach, anpassungsfähig und bequem Web-Applikationen schreiben. Auf Änderungen kann schnell reagiert werden und komplizierte Prozesse und Programmabläufe sind überflüssig. Die Web-Applikation kann während der Entwicklung im Browser aufgerufen werden und so sehr leicht immer wieder kontrolliert und angepasst werden.
Aufwendige Konfigurationen werden vermieden und durch Konventionen ersetzt. Zum Beispiel werden Datenbanktabellen über Namenskonventionen Modellen in Ruby zugeordnet. Viele Frameworks benutzen hier Konfigurationsdateien um die Zuordnung herzustellen.

Rails besteht aus fünf Modulen: ActiveRecord, ActionPack, ActionMailer, ActionWebService und ActiveSupport.
Die wichtigsten Module sind ActiveRecord, welches die Verwaltung und Abstraktion der Datenbank mittels Modellen übernimmt, und die ActionPack-Bibliothek, die aus ActionController und ActionView besteht und für Bearbeitung der Anfragen und Ausgabe zuständig ist.

Beim Anlegen eines neuen Projektes wird ein komplettes Grundgerüst – eine Verzeichnisstruktur mit Rubycode und einer Grundkonfiguration – generiert, auf dem man dann seine Web-Applikation aufbaut. Dieses Grundgerüst ist nach dem Architekturmuster Model-View-Controller (MVC) aufgebaut.

Rails erweitert Ruby um viele Helfer für die Web-Entwicklung und HTML-, sowie JavaScript-Generatoren. Es ist die JavaScript-Bibliothek von script.aculo.us (Jav06d) enthalten, eine Sammlung von Scripten um Websites mit visuellen Effekten und neuen Funktionalitäten auszustatten, die bisher auf Websites nicht üblich sind. Dies beinhaltet zum Beispiel Ein- und Ausblenden von Objekten (“Fading”) oder Drag-And-Drop-Funktionalitäten für HTML-Elemente.

Des Weiteren setzt Rails auf das Prinzip “Don’t Repeat Yourself” (DRY), um redundante Codeblöcke zu vermeiden. Zu Beginn der Bachelorarbeit war die Version 1.1.0 aktuell, jedoch erschien am 9. April 2006 die Version 1.1.2, die einige Neuerungen brachte – zum Beispiel die Einführung von rjs um die Steuerung von Webseiten über AJAX noch einfacher zu gestalten. Die Version 1.1.2 wird in dieser Bachelorarbeit verwendet.

2.2.1 Model-View-Controller-Architektur

Unter einer Model-View-Controller-Architektur (MVC) – die deutsche Übersetzung lautet Modell-Präsentation-Steuerung (MPS) – versteht man die Aufteilung der Applikation in eben diese drei Komponenten. Jede Komponente hat dabei eine spezielle Aufgabe, so kümmert sich das Model um die Daten, der Controller um die Programmlogik und die View um die Darstellung. Im Folgenden werden die einzelnen Komponenten und die Umsetzung in Rails genauer vorgestellt, wie sie im Buch über Rails von Dave Thomas beschrieben ist (THH05).

Model

Das Model repräsentiert die persistenten Daten der Anwendung. Diese sind meistens in einer Datenbank wie zum Beispiel MySQL, DB2 oder Oracle abgespeichert.

In Rails ist es die Aufgabe von ActiveRecord, die Verbindung zur Datenbank herzustellen und die Klassen und Attribute in einem Model der richtigen Datenbanktabelle und Spalte zuzuordnen. Dies geschieht durch Namenskonventionen. Rails stellt die Verknüpfung über die Gleichheit der Namen mit Singular und Plural her, das heißt zu einem Model gibt es eine Tabelle in der Datenbank die den Namen des Models im Plural hat. Zum Beispiel würde Rails für das Model “order” die Datenbanktabelle “orders” erwarten. Rails kennt aber auch unregelmäßige Plurale, so gehört zum Model “person” die Tabelle “people”.

View

Diese Komponente ist ausschließlich für die Darstellung der Daten, die es von Controller oder Model bekommt, zuständig. Darum enthält die View keine oder so wenig wie möglich Programmlogik.

In Rails generiert ActionView für diesen Zweck HTML, XML oder JavaScript. Oft werden dafür Templates verwendet, die Ruby-Code enthalten. RHTML für HTML-Dateien, RXML für XML-Dokumente und RJS-Dateien, die seit Version 1.1.2 die einfache Generierung von JavaScript-Code durch Ruby-Code ermöglichen. So kann man viel Javascript-Code mit wenig Ruby-Code ausdrücken, indem man sich der Generatoren von Rails bedient.

Controller

Der Controller steuert den Ablauf (Workflow) der Applikation. Er verarbeitet Anfragen vom Benutzer und gibt dann das Ergebnis mit Hilfe von Views an den Benutzer zurück. Gegebenenfalls kommuniziert er mit den Models, um während der Verarbeitung Daten zu erhalten oder zu ändern.

In Rails werden alle Anfragen zuerst von einem Router abgefangen, der dann die korrekte Action im entsprechenden Controller ausführt. Eine Action ist nichts anderes als eine Methode in der entsprechenden Controller-Klasse. Anhand der URL wird also die auszuführende Methode ermittelt, aber es können auch zusätzlich noch Parameter übergeben werden.

Nach der Ausführung der Methode wird die zugehörige View zur Ausgabe aufgerufen – diese View hat dann den gleichen Namen wie die Methode. Durch den 'render'-Befehl kann eine beliebige andere View zur Ausgabe genutzt werden. Oft ist es sinnvoll, einen Redirect zu benutzen, um den Browser auf eine andere Action umzuleiten. Bei einem solchen HTTP-Redirect bekommt der Browser eine neue URL geliefert, die er dann aufruft. Soll zum Beispiel nach einem erfolgreichen Eintrag eines CFPs wieder die Startseite angezeigt werden, wäre ein Redirect die richtige Wahl.

Beispiel

Am Besten kann man den Workflow der Rails-Applikation anhand eines Beispieles verdeutlichen. Es handelt sich hier um einen tatsächlichen, aber vereinfachten Ablauf aus der Applikation. In der Abbildung 2.1 ist dies grafisch dargestellt.

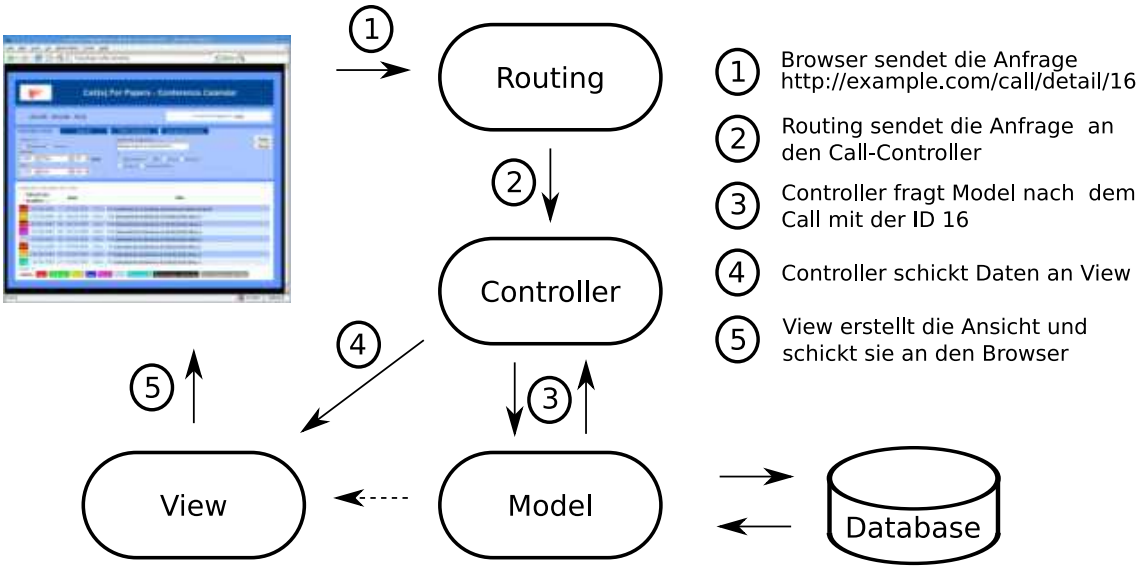

Abbildung 2.1: Die MVC-Architektur.

Möchte der Benutzer sich einen Termin zu einer Konferenz näher ansehen, so klickt er auf den Namen der Konferenz und schickt somit eine Anfrage an den Webserver mit der URL "*http://example.com/call/detail/16*" (1). Dieser Request wird vom Router in Rails abgefangen und anhand der Default-Route in /config/routes.rb (" *map.connect ':controller/:action/:id'* ") weiß Rails nun, dass in der Klasse "CallController" die Methode "detail" ausgeführt und als Argument eine Variable mit dem Namen "id" und dem Wert "16" übergeben werden muss (2). Der Controller fragt nun die Model-Klasse "Call" nach einem Eintrag mit dieser ID und speichert diesen in einem Array ab (3). Ist die Methode des Controllers beendet, ruft dieser die entsprechende View auf (4). Diese View hat Zugriff auf das Array des Controllers mit den Detail-Informationen zu dem entsprechenden Call und gibt diese eingebettet im Layout und eigenem HTML-Code aus (5).

2.2.2 andere Frameworks

Es gibt viele Web-Frameworks in vielen verschiedenen Sprachen. So gibt es in PHP zum Beispiel Frameworks wie symfony, CakePHP, Zend Framework, Agavi und PHP Trax, wobei letzteres eine direkte Übersetzung von Ruby on Rails nach PHP darstellt. Einen Überblick über diese fünf Frameworks bietet ein Artikel auf theweb20dev.com (PHP06).

Diese Frameworks sind relativ neu und orientieren sich größtenteils an Rails. Sie stellen eine Alternative dar, wenn man an die Programmiersprache PHP gebunden ist, was durchaus nicht unwahrscheinlich ist, da es nur wenige Provider gibt, die neben dem angebotenen Webspace auch die passende Umgebung für Ruby On Rails bereit stellen.

In Java programmierte Frameworks sind zum Beispiel Tapestry (Jav06b), Apache Struts (Jav06a) und WebWork (Jav06c). Für Anwendungen aus diesen Frameworks braucht man allerdings einen Application-Server wie JBoss oder Tomcat (Apa06). Es ist recht schwer herauszufinden, wo die Schwierigkeiten bei den einzelnen Frameworks liegen, da natürlich alle nur ihre Vorzüge anpreisen. Dies ist auch bei Rails nicht anders. Jedoch erfreut sich Rails großer Popularität und hat somit auch eine große, gern helfende Community. Um Rails ist ein regelrechter Hype entstanden. So gibt es auf der Rails-Website

ein Video (Rai06b), in dem ein Weblog in 15 Minuten erstellt wird. Wenn man sich gut mit Rails auskennt und seine Anforderungen anpasst, so ist dies tatsächlich möglich, sollte aber nicht überbewertet werden.

2.3 Ajax

Ajax steht für "**A**synchronous **J**avaScript **a**nd **X**ML" und wurde früher nur als XMLHttpRequest bezeichnet (Six06). Zwar ist in jedem modernen Browser Ajax schon implementiert, aber an der Standardisierung von Ajax/XMLHttpRequest wird zur Zeit beim W3C (World Wide Web Consortium) noch gearbeitet (W3C06). Mittels dieser Technik kann der Browser Daten im Hintergrund mit dem Server austauschen und dann mit Hilfe von JavaScript die betrachtete Seite mit den neuen Informationen aktualisieren. Der Browser muss diese Technologie also unterstützen sowie die Ausführung von JavaScript erlauben, was bei nahezu allen aktuellen Browsern der Fall ist. Teilweise können sich die Implementationen von Ajax allerdings unterscheiden, so dass es in seltenen Fällen zu Kompatibilitätsproblemen kommen kann.

Da der Benutzer keine Seite neu lädt, sondern sich die Seite gemäß seinen Aktionen dynamisch verändert, entsteht der Eindruck er bediene eine Desktop-Anwendung. Solche Web-Applikationen bezeichnet man oft als *Web 2.0*. Obwohl Ajax schon seit etwa 1998 existiert, wird es zur Zeit als Neuentdeckung gefeiert. Im Grunde handelt es sich bei *Web 2.0* um einen Marketing-Begriff und es wird gerne alles so bezeichnet, was gerade neu und populär im Internet ist. Ein gutes Beispiel für eine Web 2.0 Anwendung ist der Online-Service "Google Maps" (Goo06a) bei dem die Karte beim zoomen und scrollen per Ajax im Hintergrund nachgeladen wird.

Da die entwickelte Web-Plattform an ein paar Stellen Ajax benutzt und die Seite mit Hilfe von JavaScript dynamisch geändert wird, ohne eine neue Seite zu laden, könnte man sie wahrscheinlich als eine Web 2.0 Anwendung klassifizieren. Da aber Ajax sparsam verwendet wird und sich das Gefühl einer Desktop-Anwendung nicht einstellt, sollte man sie nicht zum Web 2.0 zählen.

Kapitel 3

Die Web-Plattform

Dieses Kapitel befasst sich mit der Realisierung der Web-Plattform. Diese Plattform erfüllt die grundlegenden Anforderungen zum Austausch von Informationen über wissenschaftliche Veranstaltungen und kann beliebig erweitert werden. Die Anforderungen umfassen:

Eintragen Der Benutzer soll die Möglichkeit haben, Konferenzen und Journals mit ihren Einreichungsterminen "submission deadlines" einzutragen. Dabei sollten redundante Eintragungen vermieden werden und die Eingabemaske einfach und übersichtlich bleiben. Auch wenn die Informationen des Benutzers nicht komplett sind, sollte eine lückenhafte Eintragung möglich sein. Dies betrifft hauptsächlich ungenaue oder noch nicht verfügbare Einreichungstermine von früh angekündigten Konferenzen.

Suchen Des Weiteren muss der Benutzer nach Konferenzen und Journals suchen können. Dabei soll mindestens eine Eingrenzung des Zeitraums, eine Titel-Suche sowie eine Volltext-Suche in der Beschreibung ermöglicht werden. Eine variable Auflistung hätte den Vorteil, dass der Benutzer eine bevorzugte Ansicht auswählen kann in der die Termine aufgelistet werden.

Finden Hat der Benutzer einen Termin gefunden, erhält er alle verfügbaren Informationen zu der entsprechenden Konferenz oder des Journals auf einer Seite. Weitergehende

Informationen werden durch die Verlinkung der Veranstaltungs-Website oder mögliche Erweiterungen wie zum Beispiel ein Straßenkartenservice für den Veranstaltungsort angeboten.

Struktur Die Daten der Veranstaltungen müssen in einer Struktur abgespeichert sein, bei der die Eintragung von Daten unkompliziert ist (wenige Attribute), aber auch die Suche nach einzelnen Attributen möglich ist (viele Attribute). Die Lösung besteht hier darin, nur die wichtigsten Informationen als einzelne Attribute zu repräsentieren. Der eigentliche CFP mit allen Informationen wird als Beschreibungstext für die Veranstaltung abgespeichert, über den eine Volltext-Suche möglich ist.

3.1 Datenstruktur

Die Daten zu einer Veranstaltung kann man in mehrere Bereiche unterteilen wie in Abbildung 3.1 zu sehen ist. Detaillierte Informationen wie die Beschreibungen und Links sind für eine bessere Übersicht nicht in der Abbildung enthalten.

Kurzform	Titel	Sponsor	Typ		Start	Ende	Veranstaltungsort	Termin	Termin für
VANET	International Workshop on Vehicular...	ACM	c	3rd	2006-09-29	2006-09-29	Los Angeles, California, USA	2006-06-05	paper
VANET	International Workshop on Vehicular...	ACM	c	3rd	2006-09-29	2006-09-29	Los Angeles, California, USA	2006-07-10	notification
VANET	International Workshop on Vehicular...	ACM	c	3rd	2006-09-29	2006-09-29	Los Angeles, California, USA	2006-07-31	camera-ready
JWCN	Journal on Wireless Communications...	EURASIP	j		2007-10-00	2007-10-00		2006-07-01	paper
JWCN	Journal on Wireless Communications...	EURASIP	j		2007-10-00	2007-10-00		2006-11-01	notification
JWCN	Journal on Wireless Communications...	EURASIP	j		2007-10-00	2007-10-00		2007-02-01	camera-ready
VTC	Vehicular Technology Conference	IEEE	c	65th	2007-04-23	2007-04-25	Dublin, Ireland	2006-09-16	paper
VTC	Vehicular Technology Conference	IEEE	c	65th	2007-04-23	2007-04-25	Dublin, Ireland	2006-12-04	notification
VTC	Vehicular Technology Conference	IEEE	c	65th	2007-04-23	2007-04-25	Dublin, Ireland	2007-01-29	camera-ready
VTC	Vehicular Technology Conference	IEEE	c	66th	2007-10-01	2007-10-03	Baltimore, MD, USA	2007-02-00	paper
WINTECH	International Workshop on Wireless...	NULL	c	1st	2006-09-29	2006-09-29	Los Angeles, CA, USA	2006-06-19	paper

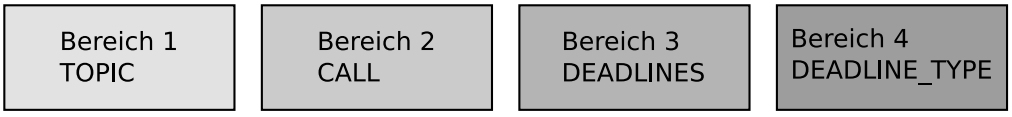

Abbildung 3.1: Unterteilung von CFP-Daten.

Schaut man sich die Daten von vielen verschiedenen CFPs an, so erkennt man als erstes, dass viele Konferenzen regelmäßig stattfinden und die CFPs zu einer solchen Veranstaltungsreihe einige Gemeinsamkeiten haben. Als Beispiel sollen hier die Einträge der Veranstaltungsreihe “VANET” aus der Abbildung 3.1 dienen. So beinhaltet der ers-

te Bereich Informationen zu der Veranstaltungsreihe wie die Kurzform (in der Abbildung ist dies “VANET”) und den Namen (“International Workshop on Vehicular Ad Hoc Networks”). Oft gibt es einen Link, einen Sponsor (“ACM”) und einen Themenschwerpunkt.

Einen zweiten Bereich stellen die Informationen zur Konferenz dar. Hier ist die Beschreibung einzuordnen zusammen mit dem Datum (“29 . September 2006”), dem Veranstaltungsort (“Los Angeles, California, USA”) und dem Link zu einer Veranstaltungsseite.

Da es zu einer Konferenz mehrere Termine gibt (“26. Mai 2006”, “10. Juli 2006” und “31. Juli 2006”), sind diese in einem dritten Bereich eingeordnet. Da es nicht nur Fristen für das Einreichen von Papern gibt, sondern auch zum Beispiel für Poster, Demos usw. gibt es noch einen vierten Bereich mit der Information, um welchen Termin es sich jeweils handelt (“Paper Submission Deadline”, “Notification of Acceptance” und “Camera-Ready Deadline”).

Man muss zwar berücksichtigen, dass neben Konferenzen auch Journals gespeichert werden sollen, da aber beides recht ähnlich ist, kann mit Hilfe von Single Table Inheritance (STI) eine gemeinsame Struktur genutzt werden. Das Rails-Buch (THH05) beschreibt STI als eine Technik mit der man Objekte, die viele gemeinsame Attribute besitzen, in einer Tabelle zusammenfasst. In dieser Tabelle werden nicht nur die gemeinsamen Attribute abgespeichert, sondern jedes Objekt ist mit allen seinen Attributen vertreten. Mit einem Zusatzattribut bestimmt man um welches Objekt es sich handelt und ignoriert beim Auslesen der Tabelle alle Attribute, die nicht zu diesem Objekt gehören. In diesem Fall entscheidet also ein zusätzliches Attribut, ob es sich um eine Konferenz oder ein Journal handelt. Handelt es sich beispielsweise um ein Journal, so macht das Attribut Veranstaltungsort aus dem zweiten Bereich keinen Sinn und wird ignoriert. Rails hat zwar eine Unterstützung für STI, aber da hier die Attribute zusätzlich auf mehrere Tabellen verteilt sind, kann man sie in diesem Fall leider nicht gebrauchen.

Die vier Bereiche sind als Objekte realisiert, die in Rails als Model-Klassen implementiert sind. Für jedes Model existiert in der Datenbank eine eigene Datenbanktabelle für die Speicherung der Daten.

Der erste Bereich wird durch das Objekt ‘topic’ repräsentiert, welches die Daten enthält,

die sich auch bei regelmäßigen Konferenzen nicht ändern. So beinhaltet es den Titel, die zugehörige Abkürzung und jeweils optional eine URL für eine allgemeine Website zu der Veranstaltungsreihe, einen Themenschwerpunkt und Sponsor. Dazu kommt noch das Attribut für STI, welches entscheidet, ob es sich um eine Konferenz oder ein Journal handelt und welche Attribute ausgeblendet werden müssen.

Das Objekt 'call' beinhaltet Informationen zu welchem 'topic' es gehört. Zudem wird gespeichert, um die wievielte Konferenz es sich handelt, wann sie beginnt und endet und wo sie stattfindet. Außerdem ist hier der komplette CFP enthalten und optional kann wieder eine URL zu einer Veranstaltungsseite gespeichert werden.

Der Einreichungstermin im Objekt 'deadline' enthält das Datum, Zeitpunkt, sowie die Information für welche Abgabe diese Frist gilt (paper, poster, demo...) und eine Verknüpfung mit dem zugehörigen 'call'.

Das Objekt 'deadline-type' speichert die Informationen aus dem vierten Bereich um welche Termine es sich handelt.

Es gibt aber auch zwei Model-Objekte ohne persistente Daten, die also keine Datenbank für die Speicherung brauchen. Ihre Daten existieren nur während des Besuches der Website in einer Session. Session-Daten können entweder auf dem Server oder beim User über Cookies gespeichert werden. Rails benutzt hier die Lösung über Cookies als Standard.

display Dieses Model speichert in seinen Attributen, wie und was der Benutzer in der CFP-Liste sehen möchte. Die Suchoptionen, die gewünschte Ordnung und verschiedene Optionen zur Ansicht der Liste sind hier enthalten.

inc_date Um unvollständige Datumsangaben verwenden zu können, wurde ein neuer Datentyp "IncDate" implementiert. Dieses Model enthält Methoden zum Definieren, Vergleichen, sowie Formatieren und Ausgeben von Datumsangaben, bei denen die Angabe des Tages, des Monats oder beides fehlt.

3.1.1 Datenbank

Die Objekte sind als Datenbanktabellen realisiert und enthalten die Attribute gemäß der oben definierten Daten. In der Namensgebung wurden einige Konventionen von Rails beachtet:

- Der Tabellenname ist jeweils der Plural des Model-Namens.
- Jede Tabelle besitzt das Schlüsselattribut 'id' mit Identifikationsnummern.
- Die Namen der Fremdschlüssel sind zusammengesetzt aus dem Model-Namen und '_id'.
- Besteht der Name des Models aus mehreren Worten (z.B. 'DeadlineType'), sind diese im Namen der Tabelle durch Unterstriche getrennt (z.B. 'deadline_types')

Mit Hilfe dieser Konventionen ist Rails in der Lage zu einem Model die entsprechende Datenbanktabelle zu finden und mit einer Zusatzinformation über die Beziehungen der Fremdschlüssel zueinander, auch die Zusammenhänge zwischen Models zu bestimmen. Von hier an übernimmt Rails die Kommunikation mit der Datenbank und im Programmablauf fragt man das Model nach Daten und nicht die Datenbank.

Die Datentypen werden automatisch erkannt (Mapping). So werden Attribute, die als VARCHAR oder TEXT in der Datenbank gespeichert sind, im Rails-Objekt zu Strings. INTEGER werden zu Fixnum (Integer werden in Ruby intern in binärer Form gespeichert und sind Objekte der Klasse Fixnum) und sowohl DATE als auch TIME werden zu ihren Repräsentationen in Ruby mit den gleichen Namen umgewandelt.

In der Datenbank werden die meisten Attribute als String repräsentiert und haben daher den Typen VARCHAR. Ausnahmen sind die Ids, bei denen es sich um INTEGER handelt, der CFP, der als TEXT abgespeichert wird, und falls eine Uhrzeit für eine Abgabefrist existiert, so wird diese als TIME gespeichert.

Zu beachten ist, dass Datumsangaben auch als VARCHAR abgespeichert werden und nicht als DATE. Da auch unvollständige Datumsangaben erfolgen dürfen, aber Rails den Datentyp DATE automatisch in den eigenen Datentyp Date umwandelt, welcher kein

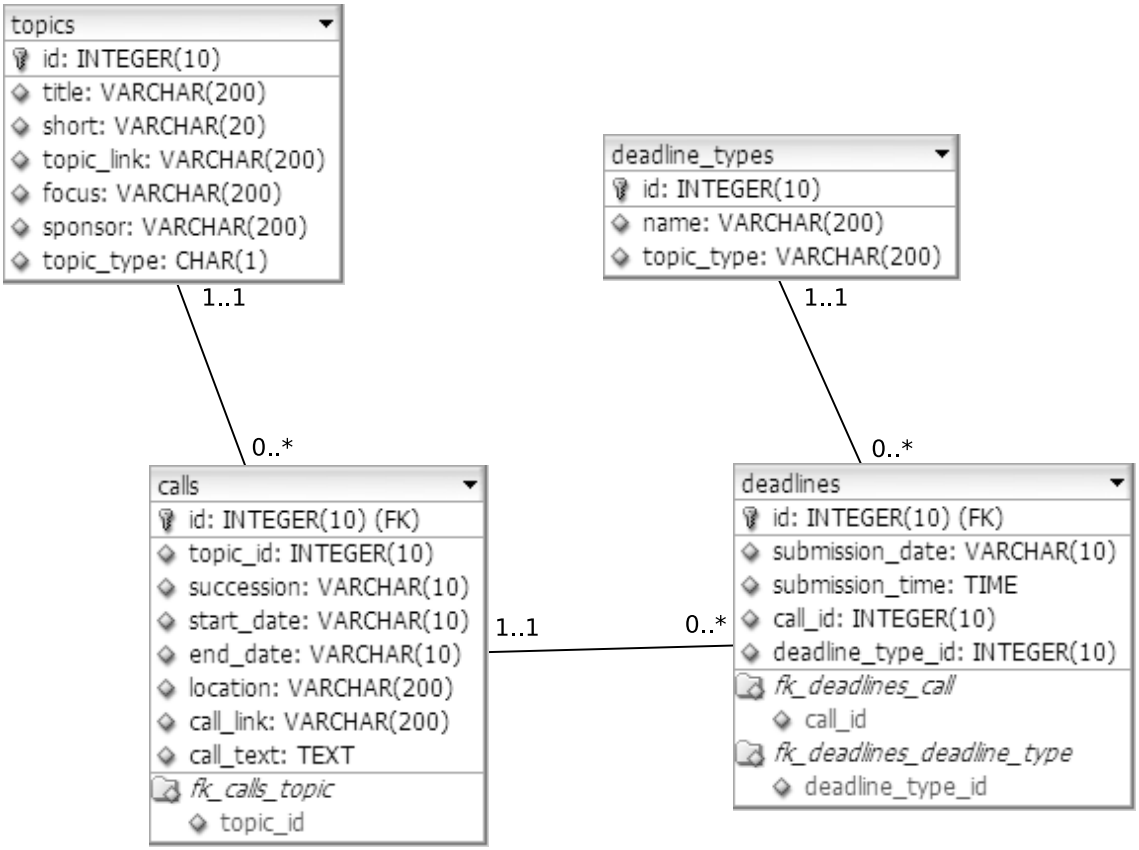

Abbildung 3.2: Das Datenbankschema.

Datum ohne Monat oder Tag zulässt, wird hier das Datum als String – also VARCHAR – abgespeichert.

Die Unterscheidung ob es sich um eine Konferenz oder ein Journal handelt, wird im Attribut topic_type der Tabelle topics getroffen. In diesem Attribut mit dem Datentyp CHAR der Länge Eins steht entweder ein “c” für conference oder ein “j” für journal.

In der Tabelle deadline_types gibt es ein Attribut mit gleichem Namen und Typ. Ist die Frist nur für Konferenzen üblich, erhält das Attribut den Wert “c”. Bei Fristen, die nur für Journals relevant sind, wird ein “j” abgespeichert und sind sie für beide üblich, bekommt das Attribut keinen Wert (“NULL”).

3.1.2 Display

Die Klasse “Display” speichert alle Einstellungen zur Ansicht und Suche.
Da man aus verschiedenen Listenformen wählen kann, wird in einem Attribut namens *@view* ein String mit dem Namen der ausgewählten Liste abgespeichert. Standardeinstellung ist ‘auto’ – die Listenform wird dann automatisch bestimmt. Genauso gibt es auch ein Attribut für die Sortierung der Liste: *@order*. Hier werden Paare mit Attributname und Richtung (“ASC” / “DESC”) angegeben. Es sind auch mehrere Paare möglich, so dass es weitere Sortierungen gibt, falls das erste Attribut gleich ist. Dies wird bisher nur intern benutzt und ist für den Benutzer nicht zugänglich. Als letztes Attribut der Ansichtsoptionen wird in *@per_page* die Anzahl der Ergebnisse pro Seite abgelegt.

Für die Suchoptionen werden bei der Initialisierung Hashes angelegt, die Optionen zu den Objekten deadlines, calls und topics speichern. Die Textsuche kann auf einen eigenen Hash zurückgreifen, in dem der zu suchende Text sowie die Angaben stehen, in welchen Attributen gesucht werden muss.

Um dem Controller diese Informationen überall zugänglich zu machen, ohne sie in einer Datenbank abzuspeichern, wird eine Instanz der Klasse in der Session – also im Cookie des Benutzers – gespeichert. Wie die folgenden Code-Zeilen zeigen, wird diese durch einen “before_filter” vor jeder Methode des CallControllers in die Variable *@display* abgelegt. So stehen dem Controller auch nach jedem neuen Request des Browsers trotzdem alle Angaben zur Ansicht ständig zur Verfügung. Falls in der Session noch kein display existiert, wird es angelegt.

```
class CallController < ApplicationController
  before_filter :find_display
...
private
  def find_display
    @display = (session[:display] ||= Display.new)
  end
end
```

Die wichtigsten Methoden der Klasse *Display* sind ‘update’ mit der man sämtliche Attribute mit neuen Parametern überschreiben kann, ‘generate_conditions’, ‘get_order’ und ‘get_view’. Fragt ein Controller zum Beispiel das Model *Deadline* nach Daten, so kann

er Bedingungen und Sortierung angeben. Diese bekommt er über die Methoden ‘generate_conditions’ und ‘get_order’ von der Variable *@display*.

Bei der Auswahl der Liste funktioniert es genauso, nur das diesmal die View direkt auf die Variable *@display* zugreift und den String für die Listenform ausliest. Um also weitere Listenformen hinzuzufügen braucht es nur die neue Liste und eine Auswahlmöglichkeit in der Searchbox. Da man nur darauf achten muss den richtigen String zu übergeben muss man weder im CallController noch im Display-Model etwas ändern.

3.1.3 unvollständige Datumsangaben

Oftmals werden in CFPs unvollständige Datumsangaben gemacht. Dies betrifft meistens das Erscheinungsdatum von Sonderausgaben zu einer Zeitschrift. Hier wird dann nur der Monat oder sogar nur das Jahr angegeben, da der genaue Termin der Veröffentlichung noch nicht bekannt ist. Dies kann auch bei Konferenzen vorkommen, wenn sie sehr früh angekündigt werden, es aber noch keinen festen Termin gibt.

Normalerweise ist das Abspeichern von einem Datum kein Problem. Es gibt extra einen Datentypen ‘Date’ in Ruby und viele Helfer in Rails, die den Umgang mit solchen Daten sehr einfach machen. Zum Beispiel gibt es gute Helfermethoden für die Formatierung von einem Datum oder das Generieren von Eingabefeldern für HTML-Formulare. Auch in SQL-Datenbanken gibt es einen Datentypen ‘Date’, der die Speicherung eines Datums problemlos möglich macht.

Doch alle diese Datentypen und Helfer sind unbrauchbar für unvollständige Datumsangaben. In Ruby ist es nicht möglich ein Datum ohne Tag zu speichern. Die Helfer zum Generieren von Eingabefeldern bieten keinerlei Optionen für unvollständige Eingaben. Wird in der Datenbank der Datentyp ‘Date’ verwendet, so speichert Rails die Variablen automatisch in seinem eigenem ‘Date’-Datentyp ab und wandelt ungültige Daten notfalls um. Um dies zu vermeiden, werden alle Datumsangaben als Zeichenketten abgespeichert. Diese Zeichenkette hat das Format “yyyy-mm-dd” und für einen fehlenden Tag oder Monat werden Nullen eingetragen. Der Oktober 2001 enspricht also “2001-10-00”. Da aber die Handhabung von Daten in Zeichenketten umständlich ist, wurde eine

eigene Klasse ‘IncDate’ zur Speicherung von unvollständigen Datumsangaben entworfen.

Die Klasse ‘IncDate’ speichert Jahr, Monat und Tag als Attribute intern ab. Ist kein Monat oder Tag gegeben, so erhält das Attribut den Wert ‘nil’ – dies ist der NULL-Wert in Ruby . Des Weiteren gibt es Methoden zum Vergleichen und Formatieren. So gibt es die Methode ‘strfincdate’, die an die Ruby-Methode ‘strftime’ angelehnt ist und mit der man das Datum durch Anweisungen frei formatieren und als Zeichenkette ausgeben lassen kann. Ist in der Anweisung die Ausgabe des Tages vorgesehen, obwohl kein Tag verfügbar ist, so führt dies nicht zu einem Fehler, sondern die Ausgabe wird dementsprechend angepasst.

Um aus der Zeichenkette in der Datenbank eine Variable des Typs ‘IncDate’ zu machen, bzw. eine solche Variable als Zeichenkette in der Datenbank abzulegen, bedarf es einer ‘Facade Column’ (THH05)[S284]. Dabei überschreibt man die Lese- und Schreib-Operationen des Models in Rails, um die notwendigen Konvertierungen zwischen Anwendung und Datenbank durchzuführen. Doch die direkte Konvertierung zwischen Zeichenkette und IncDate führt zu Schwierigkeiten bei HTML-Formularen.

Während die eingebauten Helfermethoden von Rails Eingabefelder generieren, die dann die Werte für das Jahr, Monat und Tag zu einer ‘Date’-Variable zusammenbauen können, so ist das mit selbst geschriebenen Helfermethoden nicht so leicht möglich. Die eingebauten Helfer greifen auf interne Methoden zu und auf diese hat man keine Zugriffsmöglichkeiten ohne den Rails-Code selbst zu ändern. Immerhin kann man aber Eingabefelder gruppieren und die einzelnen Werte in einem gemeinsamen Hash sammeln. Um die Eingabefelder wieder mit den richtigen Werten zu füllen – zum Beispiel wenn das Formular bei einer Fehleingabe neu geladen wird – braucht Rails diesen Hash. Aus diesem Grund konvertieren die Zugriffsoperationen des Models zwischen der Zeichenkette und einem Hash und nicht zwischen Zeichenkette und IncDate.

Bei der Eingabe werden also die Werte für Jahr, Monat und Tag in drei Eingabefeldern gesammelt, wobei auch die Möglichkeit besteht keinen Tag oder Monat anzugeben. Dann werden diese Werte in einem Hash zwischengespeichert, bevor sie für die interne Speicherung vom Model in eine Zeichenkette konvertiert werden. Um die Eingabefelder ohne weitere Konvertierungen mit bereits vorhandenen Werten zu füllen, wird auch bei der Ausgabe die Zeichenkette zuerst in einen Hash umgewandelt. Dank der Namens-

konventionen kann Rails den Hash den drei Eingabefeldern zuordnen. Für die normale Ausgabe eines Datums in der Liste oder der Detail-Ansicht wird der Hash weiter in ein IncDate konvertiert. Durch die oben beschriebene Methode ‘strfincdate’ ist dann eine problemlose Formatierung des Datums garantiert.

Für eine übersichtliche Ausgabe eines Zeitraums – in diesem Fall der Zeitraum einer Konferenz – gibt es zusätzliche Helfermethoden, die Start- und End-Datum einer Konferenz vergleichen und redundante Informationen ausblenden. Wenn zum Beispiel die Konferenz am 3. Mai anfängt und am 6. Mai aufhört, braucht man den Monat Mai nicht zweimal erwähnen und die Ausgabe “3. - 6. Mai” ist vollkommen ausreichend. Auch hier muss darauf geachtet werden, ob das Datum komplett oder nur teilweise gegeben ist.

3.2 Webdesign und Layout

Eine Website ansprechend zu gestalten und auf die Bedürfnisse der Anwendung, Benutzer und Webbrowser anzupassen, kann sehr anspruchsvoll sein. So findet man im Internet für viel Geld gestaltete Kunstwerke und professionelle Designs, aber auch die schrecklichsten Kuriositäten von Farb- und Text-Zusammenstellungen. Die hier vorgestellte Plattform will keinem dieser Extreme entsprechen und versucht ein schlichtes Design mit praktischer Funktionalität zu erreichen.

Es gibt eine HTML-Datei, die die Seite strukturiert, und eine CSS-Datei, die die definierten Absätze und Bereiche mit Eigenschaften für Position, Schriftgröße, Farbgebung usw. versieht. Mit Cascading Style Sheets (CSS) ist es möglich, durch Auswechseln der CSS-Datei das Design der Website völlig zu verändern ohne den HTML-Code zu bearbeiten.

Man unterscheidet beim Web-Design zwischen “fixed” und “liquid” Layouts. Bei einem “fixed layout” hat die Website unabhängig von der Größe des Browser-Fensters eine feste Breite. Dies hat aber Nachteile, wenn das Browser-Fenster größer oder kleiner als die Website ist. Diese Nachteile existieren beim “liquid layout” nicht. Hier passt sich die Breite der Website an das Browser-Fenster an. Da aber hier das Design übermäßig verzerrt werden kann, gibt es im W3C-Standard die Möglichkeit eine Minimal- und

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
 <head>
   <title><%= @content_for_page_topic %></title>
   <script type="text/javascript">
       // <![CDATA[
       <%= @content_for_page_scripts %>
       // ]]>
   </script>
 </head>
<body>
<div id="wrap">
  <div id="header"><img src="/images/logo.png" alt="CoCa" />
    <h1>
  <%= @page_title || "Call(s) For Papers - Conference Calendar
      "%>
    </h1>
  </div>
  <div id="navigation">
    <%= @content_for_navigation %>
  </div>
  <%= @content_for_more %>
  <div id="content">
    <%= @content_for_layout %>
  </div>
</div>
</body>
</html>
```

Abbildung 3.3: Die HTML-Datei für das Layout.

Maximal-Breite anzugeben. Dieser Standard wird von Microsofts Internet-Explorer (IE) allerdings nicht unterstützt. Um die verfügbare Fläche bestmöglich zu nutzen, ohne das Design bei zu kleinen Browser-Fenstern zu zerstören, verwendet die Plattform ein "liquid layout" mit einer Minimal-Breite. Da der IE nicht ignoriert werden kann, wird mit einem CSS-Befehl, den nur der IE interpretieren kann, eine feste Breite gesetzt – also ein "fixed layout".

In Rails gibt es ein Verzeichnis für Layout-Seiten, die die grobe Struktur festlegen. Meistens um die Seite in Bereiche aufzuteilen wie zum Beispiel Header, Footer, Navigation und Content. Abbildung 3.3 zeigt die Layout-Seite mit einem Header, in dem die Überschrift zu lesen ist, einer Navigation mit Links zu anderen Bereichen der Website und einem Content-Bereich in dem die eigentlichen Informationen erscheinen. Um die Bereiche mit Inhalt zu füllen gibt es den "content_for"-Mechanismus. Jede dargestellte View schreibt die Ausgabe automatisch in die Variable *@content_for_layout*. Zusätz-

lich kann man mit der Methode *content_for()* andere Bereiche wie hier zum Beispiel *@content_for_page_scripts* mit JavaScript oder *@content_for_navigation* mit entsprechendem HTML-Code füllen. Mit *content_for("more")* könnte man eine oder mehrere neue Zeilen zwischen Navigation und Content einfügen. Dies wird bei der Plattform benutzt, um bei der Listenansicht die Searchbox darzustellen.

3.3 Navigation

Die Navigation beschränkt sich auf einen Link für die Ausgabe der Calls in einer Liste "List Calls", einen weiteren Link für die Eingabe von Calls "Add Calls" und einen letzten für das notwendige Impressum "About". Die Navigationsleiste bleibt auf jeder Seite erhalten, um zu jeder Zeit zwischen den Bereichen wechseln zu können.

Die Links zeigen auf die Action im entsprechenden Controller. So ist für die Ausgabe der CallController zuständig und die Methode "list" wird ausgeführt. Der NewController regelt die Eingabe und hier wird die Methode "topic" ausgeführt. Der Link "About" spricht den IndexController an, der die leere Methode "about" ausführt, welche nur die entsprechende View darstellt. In diesem Fall wird nur reiner HTML-Code in den Content-Bereich des Layouts einfügt.

3.4 Eingabe

Um Informationen über wissenschaftliche Veranstaltungen anbieten zu können, müssen diese zuvor eingetragen werden. In diesem Abschnitt werden die Möglichkeiten zur Eingabe der Daten beschrieben.

Die Eingabe ist in drei Arbeitsschritte unterteilt. So hat der Benutzer relativ kurze Formulare vor sich und kann bei jedem Schritt sicher gehen, dass die eingetragenen Informationen gespeichert wurden und keine Fehler enthielten. Mögliche Eingabe-Fehler werden zwischen den einzelnen Schritten bemerkt und können so zeitnah verbessert werden. Außerdem bietet sich die Aufteilung in drei Schritte an, da der Benutzer drei Ob-

jekte angeben muss, die auf diesem Weg logisch voneinander getrennt werden. Bei den drei Objekten handelt es sich um das ‘topic’, welches aber bei einem größeren Datenbestand schon in der Datenbank existieren sollte, den ‘call’ und die einzelnen ‘deadlines’.

Rails bietet für HTML-Formulare Hilfen an. So wird bei einer richtigen Benennung der Eingabefelder das Attribut direkt dem korrektem Objekt zugeordnet. Man muss also nicht konfigurieren, welches Eingabefeld zu welchem Attribut gehört, sondern gibt Rails diese Informationen mittels Namenskonventionen.

Um zu überprüfen, ob der Benutzer einen Fehler gemacht hat, wurden Validierungsregeln aufgestellt. Beim Speichern werden diese Regeln überprüft. Verletzt ein Wert eines Attributs diese Regeln, so wird das Formular erneut geladen und eine Fehlermeldung ausgegeben. Solche Fehlermeldungen werden in Rails für das jeweilige Model gespeichert, bei dem der Fehler aufgetreten ist, und können so für jedes Model einzeln abgefragt werden. Zusätzlich wird das entsprechende Eingabefeld mit einem -Tag umgeben und kann so durch CSS kenntlich gemacht werden.

Die Fehlermeldungen für das Model *topic* werden dann nach einer fehlgeschlagenen Eintragung durch den Befehl “ *error_messages_for (‘topic’)* ” dem Benutzer angezeigt. Durch CSS wird die generierte Fehlermeldung an das Design angepasst.

Zusätzlich werden dem Benutzer – über das so genannte *flash* – kurze Nachrichten angezeigt. *Flash* ist ein Werkzeug um Informationen zwischen verschiedenen Actions zu transportieren. Im Falle dieser kurzen Nachrichten speichert man einen String ab, mit dem man dem Benutzer beispielsweise mitteilt, ob die Eintragung erfolgreich war.

Zum besseren Verständnis für die Eingabemaske und die Unterteilung in drei Schritte sollen die Daten aus Abbildung 3.4 eingetragen werden.

3.4.1 Schritt Eins – Die Veranstaltungsreihe

Im ersten Schritt der Eingabe werden die Attribute zum Model ‘topic’ abgefragt. Es wird also bestimmt, um welche Veranstaltungsreihe es sich handelt. In einem HTML-

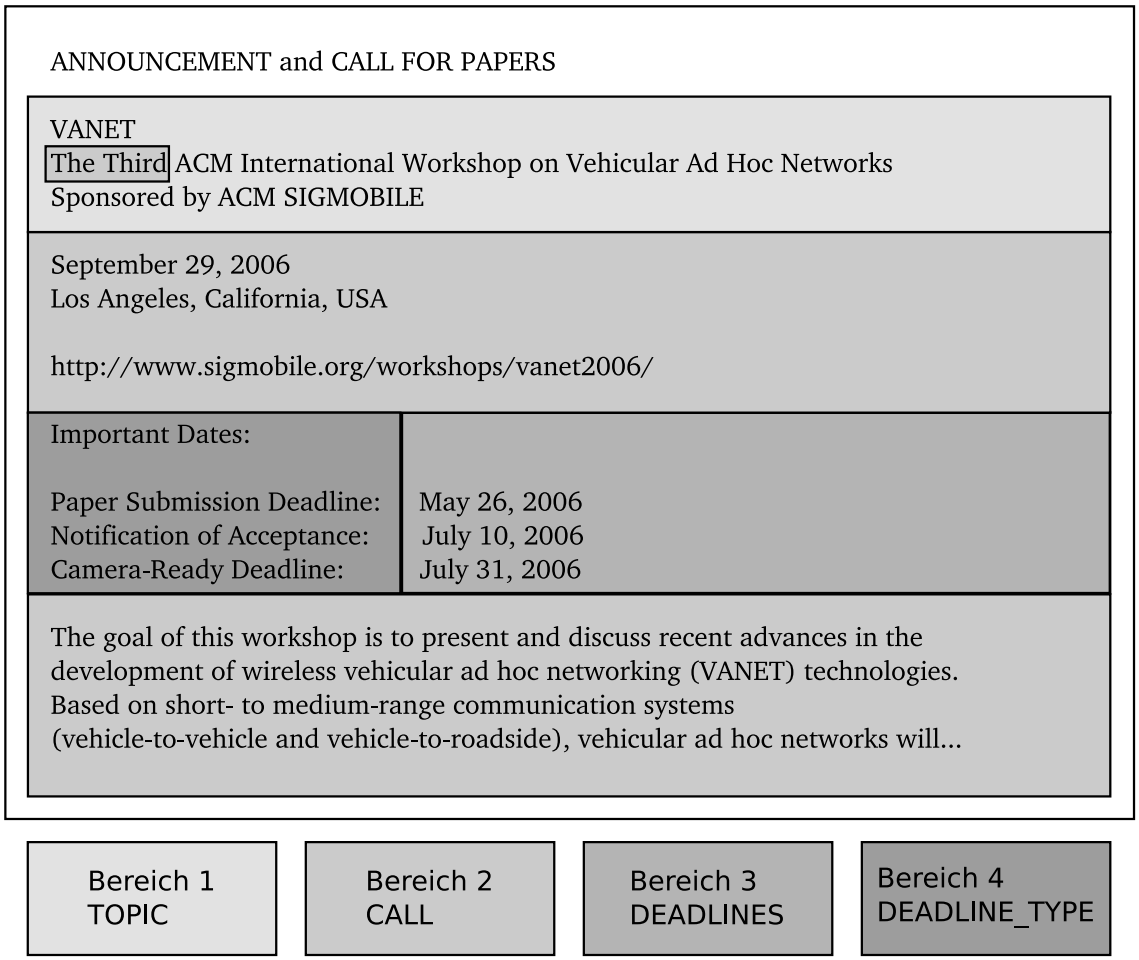

Abbildung 3.4: Eintragen eines CFP.

Formular trägt man ein, ob es sich um ein Journal oder eine Konferenz handelt, sowie den Titel und die Abkürzung. Optional kann der Benutzer einen Link, Sponsor und den Themenschwerpunkt angeben.

Man würde hier also die Daten aus dem ersten Bereich “TOPIC” eintragen. Der Link aus der Abbildung gehört nicht in diesen Bereich, da er sich nicht auf die gesamte Veranstaltungsreihe, sondern auf einen speziellen Workshop-Termin bezieht. Da die Veranstaltung bereits zum dritten Mal stattfindet, wird die Veranstaltungsreihe wahrscheinlich schon in der Datenbank enthalten sein.

Da es oft vorkommen kann, dass die Informationen schon in der Datenbank gespeichert sind, steht dem Benutzer neben dem Eingabeformular auch eine Suche zur Verfügung. Hier kann er nach ‘topics’ in der Datenbank suchen oder sie von einem Drop-Down-

Menü auswählen. Um die Eingabe nicht in noch mehr Schritte zu unterteilen, ist die Suche als “Live-Search” mit Ajax realisiert. Das Suchfeld wird nach Änderungen abgefragt und hat der Benutzer ein Suchwort eingegeben, dann schickt der Browser es im Hintergrund an den Server. Dieser antwortet mit einer Reihe von JavaScript-Befehlen in einer Datei, die die Seite je nach Ergebnis ändern. Gab es keinen Treffer, wird dies dem Benutzer natürlich mitgeteilt. Falls das Suchwort jedoch ein Ergebnis brachte, wird der erste Treffer im Formular eingetragen. Dem Benutzer wird dann angeboten, diesen Treffer zu übernehmen (‘Continue’), diesen Eintrag zu verändern (‘Update’) oder den Treffer zu verwerfen und doch einen neuen Eintrag zu erstellen (‘Insert’).

Für das Beispiel wird also im Suchfeld das Wort “VANET” eingetragen und nach kurzer Zeit wird das Formular automatisch mit Werten gefüllt. Es sollte kurz überprüft werden, ob alles korrekt ist, um dann mit einem Klick auf den Button ‘Continue’ zu Schritt zwei überzugehen.

Auf diesem Weg kann sich der Benutzer viel Schreibarbeit ersparen und redundante Eintragungen werden vermieden. Durch die Verwendung von Ajax ist die Suche unkompliziert und schnell. Veraltete Informationen auf den aktuellen Stand zu bringen, stellt keinen Umweg dar, sondern kann direkt auf dem Weg zu Schritt Zwei nebenbei geändert und gespeichert werden.

3.4.2 Schritt Zwei – Der Call for Papers

Nun geht es um die Attribute des Models ‘call’. Wurde im ersten Schritt eine Konferenz eingetragen, so wird man nun aufgefordert, das Datum (“September 29, 2006”) bzw. den Zeitraum der Konferenz und den CFP anzugeben. Anstatt den CFP kann man auch nur den beschreibenden Text “The goal of this ...” eintragen, um redundante Informationen in der Datenbank zu vermeiden. Des Weiteren kann der Benutzer Informationen, um die wievielte Konferenz dieser Art es sich handelt (“3rd”), einen Link (“http://www.sigmobile.org/workshops/vanet2006/”) und den Veranstaltungsort (“Los Angeles, California, USA”) hinzufügen. Handelt es sich allerdings um ein Journal, kann man nur ein Datum, einen Link und den CFP eintragen.

Auch hier gibt es die Möglichkeit schon in der Datenbank vorhandene Daten auszuwählen. Alle Calls zu der ausgewählten Veranstaltungsreihe sind aufgelistet und ein Call

kann durch einen Klick auf ‘select’ in das Formular eingetragen werden. Wie im ersten Schritt hat der Benutzer dann die Möglichkeit fortzufahren, den Call zu verändern oder doch einen neuen Call einzutragen.

3.4.3 Schritt Drei – Die Fristen

Als Letztes werden die Fristen eingetragen. Zu dem vorher ausgewählten oder eingetragenen Call können nun die Termine eingeben werden. Optional ist die Angabe einer Uhrzeit für den Einreichungstermin möglich.

Im Beispiel gibt es drei Termine, die man in den entsprechenden Feldern für ‘paper’ (“May 26, 2006”), ‘notification’ (“July 10, 2006”) und ‘camera-ready’ (“July 31, 2006”) einträgt.

Da hier mehrere Abgabefristen gleichzeitig abgefragt werden, wird die Namenskonvention von Rails für die Eingabefelder nicht mehr eingehalten. So müssen die eingetragenen Fristen in einer Schleife einzeln abgearbeitet und im Model abgespeichert werden. Dies ist aber für den Benutzer natürlich nicht sichtbar und er kann eine beliebige Auswahl an Abgabetypen eintragen, ohne einen weiteren Seitenaufbau in Kauf zu nehmen.

3.5 Ausgabe

In diesem Abschnitt wird kurz erklärt, wie der Benutzer an die gewünschten Informationen kommt und gezielt nach ihnen suchen kann. Anders als bei den vielen kleinen CFP-Listings kann man die Ergebnismenge vielfältig mit Filtern und Suchbegriffen beeinflussen und zwischen verschiedenen Listenformen wählen.

3.5.1 Die Liste

Die Liste der CFPs ist eigentlich eine Termin-Liste der Fristen oder eine Liste von Veranstaltungen, denn ein Benutzer der Plattform möchte entweder konkret einen Termin für eine Abgabe wissen oder Informationen zu einer Veranstaltung erfahren.

Daher gibt es nicht nur eine Liste, sondern man kann unter ‘Advanced Options’ zwischen drei Listen wählen. Die Option ‘One-Line-View’ ist die Standard-Liste und die beste Darstellung für die Auflistung von Fristen. Bei dieser Liste ist direkt ersichtlich, um welche Termin-Typen es sich handelt, wann die Frist abläuft und zu welcher Veranstaltung sie gehört.

Bei der zweiten Option ‘Tree-View’ liegt der Schwerpunkt auf Veranstaltungsreihen. So werden in einer Baumstruktur alle Veranstaltungen einer solchen Reihe unter dem gemeinsamen Titel aufgelistet und zu jeder Veranstaltung die Abgabefristen angegeben. Man erhält hier einen guten Überblick, welche Veranstaltung wann, wo und wie oft stattfand bzw. stattfinden wird.

Einen Kompromiss zwischen den beiden ersten Listen stellt die dritte Option ‘Mixed View’ zur Verfügung. Wie bei der zweiten Option sind hier die Veranstaltungen mit ihren Abgabefristen zu finden, jedoch diesmal nicht zu Veranstaltungsreihen zusammengefasst. Bei dieser Ansicht entspricht ein Listeneintrag am ehesten dem ursprünglichen CFP.

Alle diese Listen sind so genannte “shared partials”, das heißt *alle* Views können sie einbinden und zu einem *Teil* von sich machen. Welche Liste eingebunden werden soll, erfährt die View von dem Display-Objekt.

Je nach Suchkriterium sind aber nicht alle Listen sinnvoll, daher gibt es eine vierte Option ‘auto’, die automatisch anhand der eingegebenen Suchkriterien bestimmt, welche Ansicht am besten verwendet werden soll. Dies geschieht bei der Anfrage der View bei dem Display-Objekt, welche Liste sie einbinden soll. Ist die Option ‘auto’ aktiv, so entscheidet das Display-Objekt anhand der anderen gespeicherten Anzeige-Optionen, welche Liste die beste Darstellung bietet. Hat der Benutzer ein Suchwort eingegeben und möchte die Attribute Veranstaltungsort oder den Beschreibungstext durchsuchen, dann sucht er vermutlich eine Konferenz und die Veranstaltungsreihe sowie Termine sind zweitran-

gig. Für diese Suche eignet sich am Besten die Liste ‘Mixed View’. Werden aber diese Attribute nicht durchsucht, sondern zum Beispiel nur die Kurzform und der Titel, dann wird die Liste ‘Tree-View’ ausgewählt, um alle passenden Veranstaltungsreihen aufzulisten. Ohne die Stichwortsuche fällt die Wahl auf ‘One-Line-View’, da davon ausgegangen wird, dass die meisten Benutzer nach Terminen suchen. Gefällt einem Benutzer diese automatische Auswahl nicht, kann er jederzeit manuell eine Liste für seine Suche bestimmen.

3.5.2 Die Suche

Alle Einstellungen zur Suche von Veranstaltungen und Abgabefristen werden in der “Searchbox” getätigt. Diese ist zwischen Navigation und Ergebnisliste zu finden. Da das komplette Suchformular mit allen Einstellungsmöglichkeiten zu viel Platz beanspruchen würde, ist die Searchbox in drei Bereiche unterteilt, von denen immer nur einer sichtbar ist.

Im ersten Bereich “Search” kann man einstellen, ob Konferenzen, Journals oder beides gesucht werden soll. Des Weiteren kann ein Zeitraum angegeben werden, in dem die Veranstaltung stattfinden soll. Eine Textsuche steht hier ebenso zur Verfügung bei der man auswählt, in welchen Attributen gesucht wird. Unter “Filter Deadlines” kann ein Zeitraum für Fristen angegeben werden und es besteht die Möglichkeit Abgabetypen auszublenden. Welche Liste man verwenden möchte und wie viele Treffer auf einer Seite angezeigt werden sollen, ist unter “Advanced Options” festzulegen.

Damit die Informationen einer Suchanfrage nicht verloren gehen, werden alle Suchparameter im Display-Objekt abgespeichert. Erst wenn der Benutzer die Seite verlässt oder den Reset-Button in der Searchbox verwendet, gehen die Suchparameter verloren.

3.6 Usersystem

Über die Grundanforderungen hinaus wurde eine erste Erweiterung angelegt – das Usersystem. Es legt einen Grundstein für viele andere mögliche Erweiterungen, da die Platt-

form mit Hilfe des Usersystems den Benutzer wiedererkennen kann. Der Benutzer erhält die Möglichkeit sich auf der Plattform anzumelden. In der Email, die er daraufhin bekommt, findet er eine URL um seinen Account zu aktivieren. Dies verhindert eine automatische Anmeldung von Spambots.

Spambots suchen auf Websites nach HTML-Formularen und versuchen darüber Spam abzuschicken. Genau aus diesem Grund ist das Eintragen eines CFP mit der Einführung des Usersystems nicht mehr öffentlich, sondern nur noch für angemeldete Benutzer möglich. Möchte also ein Benutzer einen CFP eintragen, muss er sich zuerst anmelden und gegebenenfalls vorher einen Account erstellen. Dies stellt sicherlich eine kleine Hürde dar, aber auch einen wirksamen Schutz gegen Spam.

Für das Usersystem wurde die LoginEngine von James Adam verwendet (Rai06a). Engines sind kleine Rails-Applikationen, die eine bestimmte Funktionalität liefern wie in diesem Fall die Verwaltung von Benutzern. Der Code solcher Mini-Applikationen befindet sich im Plugins-Ordner von Rails und ist sauber getrennt vom Rest der Applikation. Controller, Models und Views der Engine lassen sich aber verwenden, wie alle anderen in der Applikation auch.

Die verwendete LoginEngine wurde für die Plattform um eine Loginbox erweitert, die in der Navigationsleiste eingebaut wurde und den Benutzer über Ajax anmelden kann. Auf diesem Weg ist für die Anmeldung kein weiterer Seitenaufbau nötig.

Kapitel 4

Zusammenfassung und Ausblick

4.1 Zusammenfassung

Um die geforderte Plattform zu implementieren, wurde zunächst ein passendes Web-Development-Framework ausgesucht. Die Wahl fiel hier auf Ruby On Rails, da es viele Vorteile bietet. Für dynamische Websites wird fast überall PHP benutzt, jedoch stellte sich die Programmiersprache Ruby und das exzellente Framework Rails als wesentlich geeigneter und produktiver heraus. Zudem ist der Einsatz von Ajax und JavaScript durch die Unterstützung von Rails leicht zu handhaben.

Damit die Plattform die Daten von wissenschaftlichen Veranstaltungen geeignet ablegen kann, wurden CFPs auf ihre Struktur hin untersucht. Schließlich wurde ein Datenbankschema mit vier Tabellen entwickelt, welches einerseits ein relationales Schema ist und andererseits eine abgewandelte Form von STI verwendet, um neben Konferenzen auch Journals speichern zu können.

Neben der Struktur für die Veranstaltungsdaten wurden auch die implementierten Objekte ‘Display’ und ‘IncDate’ vorgestellt. Mit ‘IncDate’ ist ein Datentyp entstanden, der die Verwendung von unvollständigen Datumsangaben möglich macht. Das Objekt ‘Display’ speichert alle Angaben zur Darstellung und bietet so eine individuell anpassungsfähige Anzeige, die während des gesamten Besuchs der Website erhalten bleibt.

Auf diesen Grundlagen aufbauend wurde dann die Gestaltung der Plattform sowie die Möglichkeiten der Eingabe und Ausgabe vorgestellt.

4.2 Erweiterungen

Für die oben vorgestellte Web-Plattform sind viele Erweiterungen denkbar. Über die Anforderungen hinaus kann man dem Benutzer weitergehende Features bieten, die das Benutzen der Plattform attraktiver machen. Allerdings sollte die Website nicht mit Features überladen werden, die dann niemand nutzen würde. Um die Plattform zu einem Erfolg zu machen, sollte man besonders auf die Wünsche und Vorschläge der Benutzer eingehen.

4.2.1 intelligente Eingabe

Um die Eingabe von CFPs weiter zu erleichtern, wäre eine ‘intelligente Eingabe’ als Erweiterung sinnvoll. Diese wurde aber im Rahmen der Arbeit nicht implementiert und soll hier nur als Idee für eine gute Erweiterung vorgestellt werden.

Mit einer ‘intelligenten Eingabe’ kann das System halbautomatisch erkennen, was der Benutzer eintragen möchte. Die Idee ist, dass viele CFPs einen ähnlichen Aufbau haben und das System durch eine Analyse des CFP Vorschläge zum Eintragen der Daten macht. So stehen die Angaben zur Veranstaltung am Anfang des CFP, gefolgt von den Fristen, die als ‘important dates’ kenntlich gemacht sind. Die Einreichungstermine sind eigentlich immer in einzelnen Zeilen untergebracht mit den Namen des Termins gefolgt von dem Datum. Danach folgt die Beschreibung der Veranstaltung. Durch diese automatische Erkennung bei Angabe eines CFP kann viel Arbeit bei der Eingabe erspart werden.

Denkbar ist auch, dass man dem System eine Email mit dem CFP schickt und als Antwort eine Internetadresse erhält, unter der der Benutzer ein Formular mit den vorgeschlagenen Eintragungen im besten Fall nur noch bestätigen muss.

4.2.2 weitere Ideen

Weitere Ideen sind zum Beispiel die Möglichkeit für die Benutzer Tags (Stichworte) zu Veranstaltungen einzutragen und darin suchen zu können.
Das Abspeichern von Suchparametern könnte sich als nützlich erweisen, um sie bei einem späteren Seitenaufruf wieder zu verwenden. Außerdem würden bestimmt einige Benutzer einen Email-Service nutzen, bei dem sie benachrichtigt werden, wenn ihre abgespeicherten Suchparameter neue Treffer liefern.
Um den Veranstaltungsort schnell zu lokalisieren gibt es zwar schon einen Link auf GoogleMaps, aber es wäre möglich, durch die API von GoogleMaps (Goo06b) oder eines anderen Kartenabieters wie Yahoo (Yah06) eine interaktive Karte vom Veranstaltungsort direkt bei den Detailinformationen anzubieten. Eine Übersichtskarte, auf der alle eingetragenen Konferenzen eingezeichnet sind, ist ebenfalls möglich.

4.3 Ausblick

Die Web-Plattform zum Informationsaustausch über wissenschaftliche Veranstaltungen ist bereits im produktivem Betrieb (Sch06) und enthält schon ein paar Informationen zu kommenden Konferenzen und Journals. Bei wachsendem Datenbestand kann die Plattform zu einer guten Alternative zu den kleinen CFP-Listings und der kommerziellen Variante werden. Die Möglichkeiten zur Erweiterung sind vielfältig und können die Attraktivität noch steigern.

Der nächste Schritt ist es nun, die Plattform bekannt zu machen, in dem man die Betreiber der anderen CFP-Listings kontaktiert und sie auf die Plattform aufmerksam macht. Laut eigenen Angaben hat das CFP-Listing von Alex Slingerland *täglich* fast 4000 Hits und es werden Tag für Tag um die 700 Visits gezählt (Sta06). Das Ziel der Plattform wird sein, diese Zahlen ebenfalls zu erreichen.

Literaturverzeichnis

Agi01 *Manifesto for Agile Software Development*. Version: 2001. http://agilemanifesto.org/, Abruf: 18. Jun. 2006

Apa06 *Apache Tomcat - Applikation Server*. Version: 1999-2006. http://tomcat.apache.org/, Abruf: 18. Jun. 2006

Goo06a GOOGLE INC. (Hrsg.): *Google Maps*. Version: 2006. http://maps.google.com/, Abruf: 18. Jun. 2006

Goo06b GOOGLE INC. (Hrsg.): *Google Maps API*. Version: 2006. http://www.google.com/apis/maps/, Abruf: 18. Jun. 2006

Jav06a *Apache Struts*. Version: 2000-2006. http://struts.apache.org/, Abruf: 18. Jun. 2006

Jav06b *Tapestry Project*. Version: 2006. http://jakarta.apache.org/tapestry/, Abruf: 18. Jun. 2006

Jav06c *WebWork*. Version: 2000-2006. http://www.opensymphony.com/webwork/, Abruf: 18. Jun. 2006

Jav06d *JavaScript Library, Web 2.0 Style*. Version: 2006. http://script.aculo.us, Abruf: 18. Jun. 2006

Jia06 JIANG, Wenyu: *Call For Papers in Networking Research*. Version: 2006. http://www1.cs.columbia.edu/~wenyu/research/cfp/, Abruf: 18. Jun. 2006

Pap06 *PapersInvited*. Version: 2006. http://www.papersinvited.com, Abruf: 18. Jun. 2006

PHP06 *5 Next Generation PHP Frameworks*. Version: 2006. http://www.theweb20dev.com/wordpress/2006/05/03/5-next-generation-php-frameworks/, Abruf: 18. Jun. 2006

Rai06a *LoginEngine*. Version: 2006. http://rails-engines.org/login_engine, Abruf: 18. Jun. 2006

Rai06b *Creating a weblog in 15 minutes*. Version: 2006. http://www.rubyonrails.org/screencasts, Abruf: 18. Jun. 2006

Rub06 *The Object-Oriented Scripting Language Ruby*. Version: 2001-2006. http://www.ruby-lang.org/, Abruf: 18. Jun. 2006

Sch06 SCHENKE, Gerald: *Web-Plattform zum Informationsaustausch über wissenschaftliche Veranstaltungen*. Version: 2006. http://cocacn.cs.uni-duesseldorf.de, Abruf: 18. Jun. 2006

Six06 SIXTUS, Mario: Das Netz erfindet sich neu - Programmieren mit Ajax. In: *c't* 05 (2006), Mai, S. 152–159.

Sli06 SLINGERLAND, Alex: *Call(s) for Papers on wireless and mobile network control*. Version: 2006. http://dutetvg.et.tudelft.nl/~alex/CFP/, Abruf: 18. Jun. 2006

Sta06 *Usage Statistics for the WMC Group's CFP pages*. Version: 2006. http://dutetvg.et.tudelft.nl/~alex/stats-CFP/, Abruf: 18. Jun. 2006

Str06 STROOBANDT, Dirk: *Call for Papers: conferences*. Version: 2006. http://www.elis.ugent.be/~dstr/CFP_conferentie.html, Abruf: 18. Jun. 2006

THH05 THOMAS, Dave; HEINEMEIER HANSSON, David: *Agile Web Development with Rails*. The Progmatic Programmers, 2005

Tho05 THOMAS, Dave: *Programming Ruby*. The Progmatic Programmers, 2005

W3C06 *The XMLHttpRequest Object - W3C Working Draft 05 April 2006*. Version: 2006. `http://www.w3.org/TR/XMLHttpRequest/`, Abruf: 18. Jun. 2006

Yah06 YAHOO INC. (Hrsg.): *Yahoo Maps API*. Version: 2006. `http://developer.yahoo.com/maps/`, Abruf: 18. Jun. 2006

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 27. Juni 2006 Gerald Schenke