TeamFound
Infrastrukturen zur Open Source Softwareentwicklung
Technische Universität Berlin

A. Bachmann, J. Heese, J. Kechel, M. Klink

WS 2005/2006

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Infrastrukturen zur Open Source Softwareentwicklung

Infrastrukturen zur Open Source Softwareentwicklung war eine Veranstaltung von Steffen Evers an der Technischen Universität Berlin im Wintersemester 2005/2006.

Der Inhalt dieser Veranstaltung wird auf der Projekt-Homepage wie folgt beschrieben:

Ein derartiger Entwicklungsprozess ist von der genutzten technischen Infrastruktur sehr abhängig. Wir werden daher verschiedene Werkzeuge und Plattformen untersuchen, weiterentwickeln und auch neue, unterstützende Systeme bauen.

1.2 TeamFound

TeamFound

Abbildung 1.1: TeamFound Logo

TeamFound ist wie man bereits aus dem Namen schliessen mag eine Suchmaschine für Teams. Der Vorteil einer eigenen Suchmaschine liegt ganz einfach darin, das nicht jedes Teammitglied eine herkömmliche Suchmaschine bemühen muss um Inhalte zu finden, die ein anderes Teammitglied bereits recherchiert hat. Andererseits ist es aber nicht gesagt, das die gesuchten Informationen wirklich schon in der eigenen Teamsuchmaschine vorhanden ist, daher suchen TeamFound-Clients immer auch in einer zweiten herkömmlichen Suchmaschine.

Bisherige Technik um Suchergebnisse zu verteilen integrieren sich dagegen sehr viel schlechter in einen vorhandenen Arbeitsprozess, zum Beispiel könnten einfache Bookmarks ausgetauscht werden, diese sind aber unter Umständen nicht von allen Teammitgliedern veränderbar, die Integration in einen Webbrowser ist hier quasi nicht vorhanden.

Eine weitere Möglichkeit bieten Kataloge[1] oder Verschlagwortungen wie del.icio.us [2], selbst wenn eine derartige Software in Form einer Teamsuchmaschine eingesetzt werden können, bieten sie eigentlich keine Integration in vorhandene Browser und damit würden vorhandene Arbeitsabläufe nachhaltig gestört.
TeamFound will - gerade durch die Integration in bekannte Browser - diese Integration einfacher möglich machen. Das Hinzufügen neuer Seiten sowie das Suchen im eigenen Index ist durch eine einfache Toolbar oder - genau wie bei bekannten Suchmaschinen - mittels einer Webseite möglich. Dabei können Seiten nicht nur in den Index eingefügt werden sondern auch in beliebig viele Kategorien einsortiert werden, welche wiederrum einzeln oder als Baum durchsucht werden können.

1.2.1 Komponenten

TeamFound besteht aus mehreren Komponenten, zentraler Bestandteil ist wie bei allen anderen vorgestellten Technologien ein Server, welcher Suchanfragen entgegennimmt, deren Ergebnisse übermittelt und sich außerdem um das herunterladen neuer Seiten und deren einfügen in den Index kümmert. Anders als bei anderen Technologien, ist die Client-Integration aber seit Beginn des projektes Teil der Planungen. TeamFound implementiert Clients unter anderem als Toolbar in bekannte Browser, natürlich ist es auch für ein geeignetes Webfrontend möglich TeamFound-Server zu benutzen, genauso wie auch andere Suchmaschinen meist über deren Internetseiten benutzt

[1] Jede größere Suchmaschine bietet eigene Kataloge. Das DMOZ-Projekt (http:/www.dmoz.org) stellt einen unabhängigen Versuch dar, das Internet zu kategorisieren

[2] http://del.icio.us

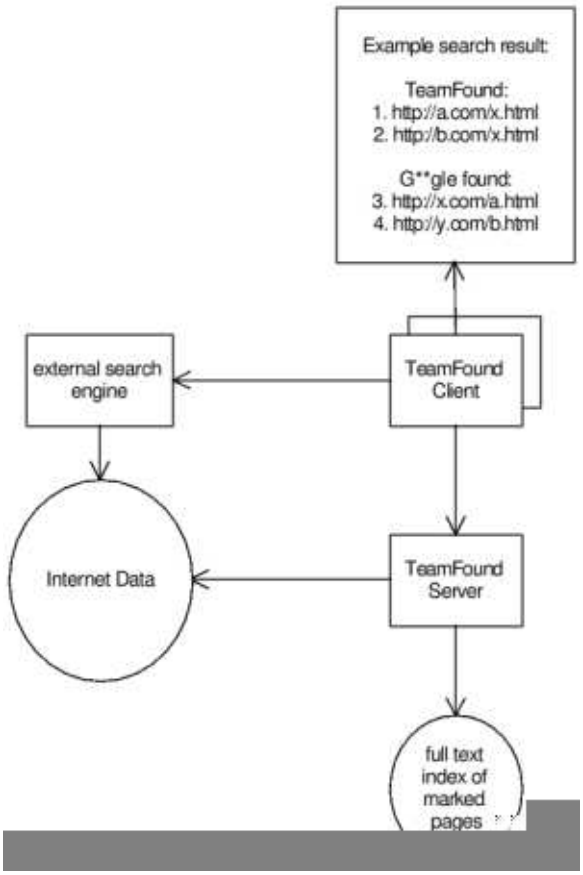

Abbildung 1.2: grobe Architektur Client - Server - Suchmaschiene

werden. Allerdings wurde bei der Clientkonzeption von TeamFound eher auf die Möglichkeiten von Browsererweiterungen eingegangen, als auf die Anforderungen von Webseiten.

Im Folgenden werden die einzelnen Teile von TeamFound detailliert dargestellt. Dabei handelt es sich zuerst um den TeamFound-Server und anschliessend um die Toolbars für den Firefox-Browser vom Mozilla-Projekt sowie der Integration in den Internet Explorer von Microsoft. Da diese Browser auf sehr unterschiedlicher Technik basieren werden die einzelnen Toolbars auch getrennt beschrieben.

1.2.2 Funktionsumfang

Suchanfragen

Suchanfragen an TeamFound werden wie in jeder anderen Volltextsuche mittels Schlüselwörter gestellt. Dabei hängt es von der Index-Implementierung ab, wie genau verschiedene Felder durchsucht oder Schlüsselwörter verknüpft werden können.

Kategorien

Jedes Dokument im Index kann in einer oder mehrerer Kategorien enthalten sein. Kategorien werden dabei als Baum dargestellt, es gibt also eine Wurzelkategorie sowie eine unbestimmte Anzahl weiterer Kategorien die jeweils eine Elternkategorie haben.

Suchanfragen an eine Kategorie findet nur Dokumente in dieser oder einer ihrer Unterkategorien. Das wirkliche Begrenzen der Suche auf genau eine Kategorie, ohne Unterkategorien ist in dieser TeamFound-Version noch nicht vorhanden.

Kapitel 2

Teamfound-Server

2.1 Vorüberlegungen

2.1.1 Anforderungen

In der Vorbereitung des Projektes wurde unter anderem auf die Verzweigung von Open Source-Projekten eingegangen: Es soll nicht jedesmal das Rad neu erfunden werden, eine wichtige Herausforderung im Open Source-Entwicklungsmuster ist es einerseits andere geeignete Projekte zu finden und diese dann effektiv in das eigene Projekt zu integrieren. In diesem Sinne war die erste Phase in der TeamFound-Entwicklung von Recherchearbeiten geprägt. Für den TeamFound-Server heisst das: Es muss eine Bibliothek gefunden werden, welche uns die Volltextindexierung von beliebigen Dokumenten möglich macht. Diese Bibliothek muss dann - sofern sie dieses nicht schon selber anbietet - geeignet ans Internet gebracht werden, einerseits um neue Dokumente herunterzuladen und in den Index aufzunehmen, andererseits um Suchanfragen entgegenzunehmen und deren Ergebnisse zurückzugeben. Diese Bibliothek sollte möglichst erweiterbar sein, um auch die Anforderungen von (relativ) einfachen Dokumentenkategorien umsetzen zu können. Für die Verwaltung der Kategorien würde eine Datenbank benötigt werden, da die Verknüpfung von Kategorien, sowie deren Details, wie Namen und Beschreibung nichts mit der Index-Bibliothek zu tun haben.

2.1.2 Plattform

Die Entscheidung für eine Plattform wurde implizit durch die Entscheidung für eine Bibliothek zum indexieren und durchsuchen von Dokumenten getroffen. Nachdem einige Tage nach derartigen Bibliotheken Ausschau gehalten

wurde, ist sehr schnell deutlich geworden, das Lucene [1] vom Apache-Projekt[2] die am weitest entwickelte Bibliothek ist. Ein weiterer Vorteil ist die aktive Community rund um Lucene, so wurde während unserer Entwicklung noch ein Sprung von Version 1.5 auf 1.9 mitgemacht. Andere Bibliotheken, zum Beispiel in Perl oder PHP boten keine ausreichende Dokumentation oder Erweiterbarkeit. Die ursprüngliche Lucene-Version ist in Java implementiert, ein weiteres Projekt bemüht sich um dessen Umsetzung in C und mittlerweile (Am Ende des Projekts hat Zend Technologies[3] einen PHP-Wrapper[4] für Lucene veröffentlicht.

Somit war die Wahl der Plattform auf Java gefallen, eine erste mehr als Proof-of-Concept anzusehende Version von Teamfound wurde mittels zwei Perl-CGI-Scripten und einem Systemaufruf der Lucene-Bibliothek umgesetzt. Dieses Vorgehen startet bei jeder Anfrage an den Webserver eine neue Java virtuel machine, was natürlich überhaupt nicht effektiv ist.

Daher fiel eine weitere Entscheidung, für eine Java-Servlet-Umsetzung, um die benötigte Umgebung für Anfragen an den Lucene-Index nicht für jedes Request neu zu schaffen. Somit besteht der Server aus einer 100%igen Java-Umgebung und es entfällt damit die Notwendigkeit für aufwändige und zeit-kostende Wrapper um mehrere Programmiersprachen zu verbinden.

2.2 Bibliotheken

2.2.1 Apache Lucene

Allgemein

Lucene ist - laut eigener Aussage - der harte Teil einer Suchmaschine. Diese Bibliothek erzeugt und durchsucht einen Volltextindex, kümmert sich aber weder darum wo die Daten herkommen, wie die Ergebnisse präsentiert werden oder wie die Suchanfragen vom Benutzer an den Index kommen. Der Index ist hingegen bietet viele Möglichkeiten die eigenen Daten zu analysieren und in sogenannten Dokumenten in den Index zu speichern. Eigene *Analyzer* sind mit wenig Aufwand umzusetzen (sofern die mitgelieferten nicht ausreichen), auch aufwändigere Parser sind möglich, wie es zB. für HTML-Seiten notwendig ist. Darüberhinaus kann jedes Dokument im Index beliebige *Felder* enthalten, ein Feld ist als Inhaltselement eines Dokuments

[1] www.lucene.de
[2] http://www.apache.org
[3] http://www.zend.com
[4] http://framework.zend.com/manual/en/zend.search.html

zu verstehen (Überschrift, Inhalt, Beschreibung). Dabei können völlig verschiedene Arten von Dokumenten in einem Index gespeichert werden sowie Suchanfragen direkt an ein oder mehrere Felder gestellt werden. Diese starke Erweiterbarkeit ist zum Beispiel bei der Implementierung von Dokumentenkategorien in TeamFound sehr wichtig gewesen (Siehe Kapitel 1.2.2)

Wichtige Komponenten

Document und Field Ein Lucene-`Document` ist die Basisspeichereinheit in einem von Lucene erstellten Index. Die Argumente bzw. Ergebnisse beim Einfügen und Suchen sind immer Lucene-`Documents`.

Diese Dokumente bestehen aus einer variablen Anzahl von Feldern (`Field`). Um ein Dokument zu erstellen kann man beliebig Felder definieren. Sinnvollerweise ist mindestens ein Inhaltsfeld definiert. Dieses wird in Token zerlegt und indiziert. Außerdem ist es zweckmäßig weitere Felder zu definieren, die nicht indiziert oder zerlegt, sondern als zusätzliche Information gespeichert werden. Ein eindeutiges Schlüsselwort oder eine URL sind typisch, da nur so eine Unterscheidung von Dokumenten möglich wird. Es können beliebig weitere Felder angelegt werden z.B. Author, Adresse, Zusammenfassung oder Kategorie.

Allerdings werden Felder generell uninterpretiert als Text abgelegt. Ausser Felder (wie das Inhaltsfeld), die zerlegt und indiziert werden. Diese sind textuell nicht wiederherstellbar, aber jetzt mithilfe von Anfragen durchsuchbar.

Analyzer Der `Analyzer` zerlegt zu indizierende Felder in Tokenstreams, damit diese dann ausgewertet werden können. Das kann beliebig komplex oder simple sein. Ein Analyzer kann z.B. speziell auf eine bestimmte Sprache ausgelegt sein um Artikel, Pronomen etc. direkt ausfiltern oder einfach anhand von *Whitespaces* Token bilden. Als Nutzer von Lucene bestehen hier beliebig Möglichkeiten selbst zu erweitern um eigene Ansprüche zu erfüllen.

Genauso ist es möglich verschiedene `Analyzer` zu verbinden um unterschiedliche Felder auf unterschiedliche Weise zu zerlegen.

Query und QueryParser Um nun indizierte Felder zu durchsuchen ist die Grundeinheit die `Query`. Der Name ist etwas irreführend, da es sich hierbei noch nicht um wirkliche Anfragen handelt. Eine `Query` besteht erstmal nur aus einem einzelnen Token. Nach diesem könnte gesucht werden. Lucene bietet nun verschiedenste Möglichkeiten solche Grundsteine in vielfältiger Art zu kombinieren um komplexere Anfragen zu bilden. Dies ist vorallem sinnvoll für Anfragen, die sich oft wiederholen und automatisiert gestellt werden.

Um allerdings nach von Menschen formulierten Anfragen zu suchen, bietet Lucene eine Erleichterung. Der `QuerryParser` stellt eine einfache Syntax zur Verfügung. Diese wird ausgewertet und liefert die komplexe Anfrage für den Index. Die Syntax ist ähnlich der von Google oder anderer Suchseiten. (siehe Literaturverzeichnis LucQS)

2.2.2 HSQLDB

Die Entscheidung keine für Webapplikationen typische Datenbank wie MySQL einzusetzen ist leicht erklärt. Wie schon in Kapitel 2.1.2 beschrieben, sollte eine Java-Umgebung eingesetzt werden. Das schränkt erstmal nur auf Datenbanken ein, die JDBC kompatibel sind. Deshalb ist der Server auch leicht um einen weiteren Datenbanklayer zu erweitern, falls andere Datenbanken erwünscht sind.

HSQLDB wurde von uns ausgewählt, weil wir sie als kleine Java-Bibliothek einbinden können und es damit den Nutzern unseres Servers ersparen sich extra um einen Datenbankserver zu kümmern. HSQLDB kann als Server oder in Programme eingebettet laufen. In der derzeitigen Version nutzen wir den eingebetteten Modus, und haben somit eine kleine Datenbank für den TeamFound-Server allein. Die üblichen Riten wie Datenbanknutzer anlegen usw. können wir dadurch im Server selbst automatisiert regeln. HSQLDB hat sehr gute Ergebnisse in dem PolePosition benchmark (siehe Literaturverzeichnis PolPos) geliefert und unterstützt einen grossteil der ANSI-92 SQL, SQL 99 und 2003 Standards bis hin zu Transaktionen. Ausserdem wird HSQLDB in einigen bekannten Projekten(z.b. JBoss, Mathematica, OpenOffice) erfolgereich eingesetzt, was uns ausreichend von der Zuverlässigkeit überzeugt hat.

2.2.3 Weitere Bibliotheken

Zur Erzeugung der XML-Antworten (Siehe Kapitel 2.3.4) wird die JDOM-Bibliothek[5] von Jason Hunter benutzt. JDOM ist eine abgewandelte DOM-Implementierung[6] und bietet die Möglichkeit mit einfachen Java-Code sauberes XML zu erzeugen. JDOM wird ebenfalls benutzt um die XML-Dokumente in HTML zu wandeln, sofern der Client dieses wünscht.

[5] http://www.jdom.org
[6] Document Object Model

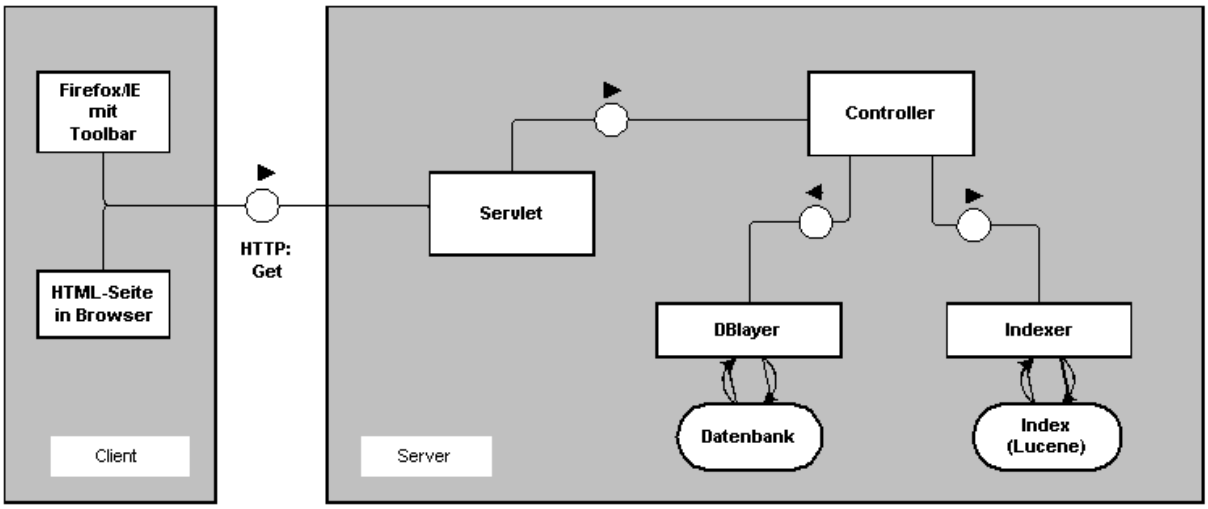

Abbildung 2.1: grobe ServerArchitektur

2.3 Architektur

In diesem Kapitel soll ein Einblick in den Aufbau des Servers gegeben werden. Die Abbildung 2.1 gibt einen groben Überblick über Komponenten des Servers. In den folgenden Sektionen werden dann einzelne Komponenten näher erläutert.

2.3.1 Servlet

Das Servlet ist der Einstiegspunkt für jegliche Anfragen an TeamFound. Dem Servlet kommt dabei die Rolle eines Wächters zu, welcher überprüft ob notwendige Parameter für die einzelnen Anfragen vorhanden sind. Ist eine Anfrage gültig, wird diese an den sogenannten Controller weitergereicht, welcher die eigentlichen TeamFound-Komponenten steuert. Dieser Controller gibt immer ein Response-Objekt zurück, welches vom Servlet dann zB. als XML oder HTML serialisiert und als Antwort zum Client geschickt wird.

2.3.2 Controller

Die Hauptaufgabe des Controllers besteht darin die Teilaufgaben, die sich aus einer Anfrage ergeben, koordiniert an Datenbank und Index zu stellen. Ausserdem sorgt er für den Download der zu indizierenden Dokumente.

Die Zugriffe auf den Index werden dabei durch eine Instanz eines Indexers erledigt. Um die Zugriffe der Indexer auf den Index zu steuern, besitzt

der Controller ein Semaphorobjekt. Dieses realisiert eine Leser-Schreiber-Kooperation der beauftragten Indexer.

Der Zugriff auf die Datenbank erfolgt über den DBLayer. Dieser stellt sämtliche Funktionalität zum lesen und schreiben der Datenbank zur Verfügung.

Wenn alle Teilaufgaben erfüllt sind (siehe Abläufe 2.5), generiert der Controller aus den erhaltenen Informationen eine Response und übergibt diese dem Servlet.

2.3.3 Indexer

Der Indexer ist für die Arbeit direkt am Lucene-Index verantwortlich. Er fügt Dokumente (siehe 2.2.1) in den Index ein oder entfernt sie wieder. Ausserdem führt er Suchanfragen auf dem Index aus. Die von uns definierte Dokumentenstruktur (siehe 2.4.2) ist bereits im Controller bekannt, somit kann der Indexer vorgefertigte Dokument erhalten und als Ergebnis liefern. Zur Indizierung und zur Suche benutzt der Indexer einen eigenen Analyzer (siehe 2.2.1), der wird hauptsächlich benötigt um das Kategorienfeld (siehe 2.4.2) korrekt zu zerlegen. Ausserdem gehört ein einfacher HTML-Parser zum Indexer, dieser soll den Text und einige zusätzliche Informationen (z.b. contenttype) aus den HTML-Seiten extrahieren. Weitere Parser z.b. für Postscript oder PDF-Dateien sind in zukünftigen Versionen denkbar.

Die Anfragen werden zu einem Teil durch den QueryParser (siehe 2.2.1) und zum anderen vom Indexer generiert. Der Indexer erstellt dabei die Einschränkung welche Kategorien in der Anfrage gültig sind. Aus beiden Teilen wird die gesammte Anfrage gebildet und an den Index gestellt. Die Verarbeitung der Ergebnisse übernimmt der Controller.

2.3.4 Response

Antworten werden in TeamFound immer als XML generiert, diese XML-Repräsentation kann bevor sie zum Client geschickt wird, noch mittels XSLT in eine geeignete HTMLFassung gewandelt werden. Eine Basisklasse Response übernimmt dabei das Handling des eigentlichen JDOM-Document-Objektes, während die eigentlichen AntwortKlassen, wie SearchResponse oder AddPageResponse ihre eigentlichen Inhalte an die Standardelemente einer Antwort anhängen (Siehe dazu mehr in Kapitel 3 auf Seite 23, Protokoll).

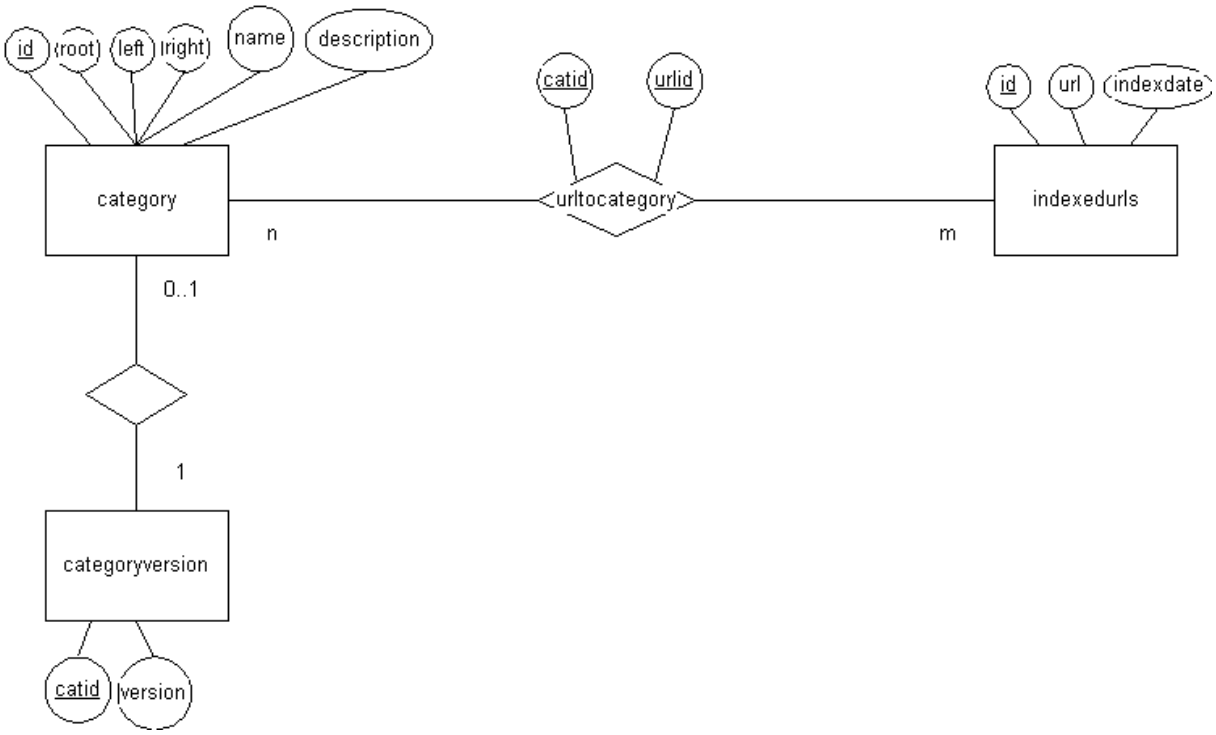

Abbildung 2.2: Datenbank-ERD

2.4 Datenmodell

In diesem Kapitel soll auf die Datenstrukturen in der Datenbank und im Index eingegangen werde.

Ausserdem soll erläutert werden, warum Daten auf diese Weise zwischen Index und Datenbank verteilt wurden und wo Zusammenhänge bestehen. Dabei wird das Hauptaugenmerk auf die Struktur der Kategorienrepräsentation gelegt. In späteren Versionen von Teamfound wird an dieser Stelle sicherlich auch das Problem von Nutzer- und Rechtemanagement zu klären sein.

2.4.1 Datenbank

Die Datenbank besteht zur Zeit aus 4 Tabellen (siehe Abbildung 2.2). Die wichtigste Tabelle des derzeitigen Standes ist die *category*-Tabelle. Sie beinhaltet die Kategoriebäume der Projekte eines Servers.

Um dieBäume in der Datenbank zu repräsentieren haben wir Verschachtelte Mengen (nested sets) eingesetzt. Dadurch können wir mit einfachen `select`-statements den Baum durchsuchen, allerdings ist das einfügen oder edititeren des Baumes aufwendig. Da wir Serverseitig hauptsächlich Such-

vorgänge zu bedienen haben, wollten wir eine rekursive Tabellenstruktur in der Datenbank vermeiden. Die Verschachtelten Mengen boten uns eine gute Variante dies zu umgehen. Genauere Informationen hierzu befinden sich im Kapitel 2.4.3 .

Die Tabelle *categoryversion* könnte prinzipiell zu jeder Kategorie eine Versionsnummer mitführen. Derzeit wird dies nur für Wurzelknoten getan. Damit hat jeder vom Server verwaltete Kategoriebaum eine Versionsnummer. Diese wird benutzt um die Aktualität der Bäume zwischen den Clients untereinander und mit dem Server sicherzustellen. Es ist durchaus vorstellbar Versionsnummern bei großen Projekten auch für Teilbäume zu verteilen, was derzeit aber nicht geplant ist.

Zu jeder Kategorie können beliebig viele Urls zugeordnet werden. Diese werden in der *indexedurl* Tabelle gehalten. Jede Url, die indiziert wurde, wird mit Datum in dieser Tabelle gespeichert. Somit kann schnell festgestellt werden, ob eine URL schon einmal hinzugefügt wurde und zu welchen Kategorien sie bisher zugeordnet war. Da der Index selbst auch doppelte Einträge zulässt, ist es wichtig vor dem Einfügen das vorhandensein überprüfen zu können.

Jede Url kann auch beliebig zu Kategorien zugeordnet werden, da wir dieselbe Url in verschiedenen Kategoriebäumen oder auch in verschiedenen Kategorien in ein und demselben Baum zulassen möchten. Die Nutzer von Teamfound entscheiden in welche Kategorie ein Document gehört. Unabhängig davon wäre eine Überprüfung, ob solch eine Zuordnung sinnvoll ist, auch kaum umzusetzbar.

Im Index wird allerdings zu jeder URL nur ein einziges Dokument (siehe 2.2.1) abgelegt. Wichtig ist dabei, dass eine Url, die einer Kategorie hinzugefügt wird, auch allen zugehörigen Elternkategorien zugeordnet wird. Das passiert in der Datenbank als auch im Dokument des Indexes. Dadurch werden bei einer Suchanfrage innerhalb einer Kategorie automatisch auch alle Urls geliefert, die Kindkategorien dieser Kategorie zugeordnet wurden.

Das Datum in der *indexedurl* Tabelle wird genutzt um Urls zu erkennen, welche schon lange im Index liegen. Diese werden dementsprechend neu indiziert.

2.4.2 Index und Dokumentstruktur

Die Bibliothek Lucene macht bestimmte Vorgaben wie die Textdokumente, die von Lucene indiziert werden sollen, zu erstellen sind. (siehe 2.2.1)

In diesem Abschnitt soll kurz der Aufbau eines von Teamfound erstelltem Dokuments beschrieben werden. Zur Hilfestellung lohnt es sich Abbildung 2.3 anzuschauen.

Jede Html-Seite, die dem Index hinzugefügt werden soll, muss erstmal serverseitig heruntergeladen werden. Danach wird der Text und weitere Informationen extrahiert. Dies alles, sowie die Kategorien zu denen die Seite zugeordnet wurde, wird nun in ein Lucene-Dokument verpackt.

Das erste Feld (siehe 2.2.1), welches ein Teamfounddokument enthält, ist ein Schlüsselwortfeld. In diesem wird die Url zu der dieses Dokument gehört gespeichert. Dieses Feld wird nicht durch den Teamfound-*Analyser* (siehe 2.2.1) zerlegt, aber durchsuchbar im Index abgespeichert. Die Url wird also zur eindeutigen Identifikation eines Dokumentes benötigt. Zusätzlich ist es möglich den Index nach Urls zu durchsuchen.

Als nächstes speichern wir eine sehr kurze Zusammenfassung. Diese wird weder zerlegt noch indiziert. Sie dient ausschließlich als nähere Beschreibung eines Suchergebnisses einer Anfrage.

Ein weiteres Feld ist der Titel der Seite. Er wird zerlegt und indiziert. Anfragen die Schlüsselwörter enthalten, die im Titel vorkommen, erhalten dadurch eine höhere Wertigkeit.

Das wichtigste Feld neben der Url enthält den textuellen Inhalt der Seite. Dieser wird in Token zerlegt und indiziert. Anfragen an den Index durchsuchen grundsätzlich dieses Feld und das Titelfeld.

Die Ergebnisse solcher Anfragen sollen nun durch Kategoriezugehörigkeit eingeschränkt werden. Dafür umfassen Teamfounddokumente ein weiteres Feld, dass die Kategorien enthält denen das Dokument zugeordnet wurde.

Die Anfrage über Inhalt und Titel wird also mit einer zweiten Suchanfrage über dem Kategorienfeld verknüpft und eingeschränkt. Um solche Anfragen zu ermöglichen muss jedes Token in einem Kategoriefeld eindeutig einer Kategorie zuzuordnen sein.

Die einfachste Lösung war die ID einer Kategorie aus der Datenbank als Token zu benutzen, und diese in dem Kategoriefeld zu speichern. Damit es möglich ist die verschiedenen IDs zu unterscheiden, trennen wir diese mit einem eindeutigen Charakter voneinander ab. Den Teamfound-*Analyser* haben wir dementsprechend angepasst, so dass er die Tokens dieses Feldes einfach anhand des Trenncharakters generiert. Auf diese Weise können wir die Anfragen leicht so erweitern, dass sie mithilfe von Kategorien eingeschränkt werden können.

2.4.3 Repräsentation der Kategorien

Da ein wichtiger Teil des derzeitigen Servers die Verwaltung der Kategorien betrifft, soll hier noch einmal etwas genauer auf diese eingegangen werden.

Wie schon in Kapitel 2.4.1 erwähnt wurde sind die Kategorien baumstrukturiert und werden in Form von verschachtelten Mengen in der Datenbank

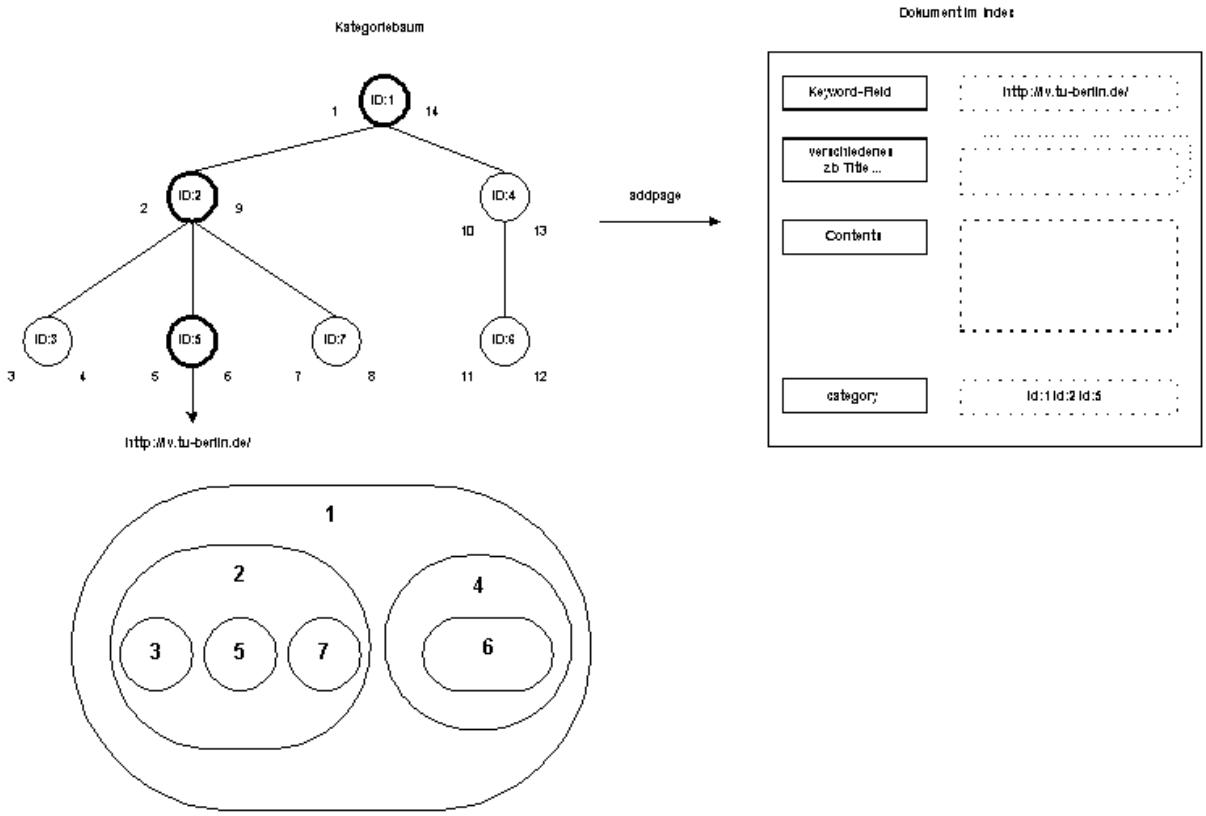

Abbildung 2.3: Repräsentation der Kategorien in DB und Index

abgelegt. Die grafische Darstellung eines solchen Baumes und einer solchen Menge kann man in Abbildung 2.3 sehen. Es ist leicht zu erkennen, dass alle Knoten des Baumes einen linken und einen rechten Wert aufweisen. Anhand dieser Werte kann die Mengen-Baumstruktur in der Datenbank nachgebaut und der Baum durchlaufen werden. Es ist leicht zu erkennen, dass der rechte Wert eines Knotens größer ist als alle Werte der Kindknoten und genauso ist der linke Wert kleiner. Es gelten noch einige andere nützliche Dinge, z.b. können alle Blätter des Baumes sehr leicht indentifiziert werden, da bei diesen die Differenz von linkem und rechtem Wert einfach 1 beträgt.

Wir haben diese Struktur ausgewählt, weil das durchsuchen sehr einfach und schnell geht. Der grosse Nachteil ist das Verändern eines solchen Baumes. Damit die Struktur in Takt bleibt muss prinzipiell jeder Knoten der "rechts" des bearbeiteten Knotens liegt angepasst werden. Dies wird bei sehr tiefen Bäumen entsprechend aufwendig.

Das sollte für unseren Anwendungsfall aber kein Problem sein. Die Kategorien werden von Nutzern der Übersichtlichkeit halber angelegt und sollten deshalb wohl nicht so tief und stark verzweigen. Ausserdem gehen wir davon aus, dass das verändern der Kategorien eher selten vorkommen wird. Wenn ein neues Projekt beginnt werden in der Anfangsphase von den Mitgliedern sicher einige Kategorien angelegt, diese werden danach aber auch nahezu unverändert bleiben.

Auf der anderen Seite muss bei sehr vielen Anfragen der Baum durchsucht werden. Allein das Anlegen einer neuen Kategorie hat zur Folge, das jeder Client den neuen Kategorienbaum vom Server erfragt. Da wir also den Baum sehr oft auslesen müssen ist diese Art der Speicherung vorteilhaft. Hinzukommt, das es sehr aufwendig ist einen solchen Baum oder Pfade eines Baumes in einem Index abzulegen. Deshalb speichern wir im Index auch nur eine einfache Aufzählung von Kategorien ab. (siehe Abbildung 2.3 und Kapitel 2.4.2) Diese enthält zwar den ganzen Pfad, dennoch lässt sich dieser nicht wiederherstellen, da ja mehrere Pfade in der Aufzählung enthalten sein können. Es entsteht dadurch aber kein Problem. Eine Suchanfrage an den Server benötigt trozdem keinen Datenbankzugriff, sondern kann mit einer Anfrage an den Index bearbeitet werden. (siehe Kapitel 2.5.2) Das lässt sich mit dem Zweck unserer Anwendung erklären. Wir stellen eine Suche anhand von Suchbegriffen bereit. Die Kategorien sind nur eine Einschränkung der Suchergebnisse. Je allgemeiner die Kategorie in einer Suchanfrage gewählt wird, desto mehr Ergebnisse werden gefunden. Das erklärt sich darin, das alle Dokumente, die einer Unterkategorie angehören, auch allen Elternkategorien zugeordnet wurden. Dieser einfache Trick erreicht genau unseren Zweck. Bei einer Suchanfrage wird ein Dokument in jeder Kategorie, die auf dem Pfad durch den Baum liegt gefunden und sie kann schnell ausgeführt werden.

2.5 Abläufe im Server

Wie Aufrufe vom Client im Server abgearbeitet werden, soll kurz in diesem Abschnitt behandelt werden. Wir haben drei verschiedene Serverdienste ausgewählt. Sie sollen einen Einblick in die Interna des Servers bieten und dadurch die Funktionsweise von Teamfound noch etwas besser verständlich machen.

2.5.1 AddpageRequest

Die wichtigsten Funktionen von Teamfound sind ohne Zweifel, dass hinzufügen und das Suchen. Deshalb soll als erstes ein Einblick in das Hinzufügen von Urls gegeben werden. Hierzu liefert die Abbildung 2.4 einen Überblick.

Der Aufruf des Clients wird in *Get*-Variablen in der Url codiert (siehe A.1) und vom Servlet (siehe Kapitel 2.3.1) entgegengenommen. Ist der Aufruf korekt, d.h. er enthält die hinzuzufügende Url und mindestens eine Kategorie, wird der Controller (siehe 2.3.2) mit entsprechendem Auftrag angestossen. Bei nicht korekten Aufrufen wird eine `ErrorResponse` erstellt.

Der Controller erstellt nun eine Verbindung zur Datenbank und einen Indexer (siehe Kapitel 2.3.3). Als nächstes überprüft er anhand der Datenbank ob die Url schon existiert. An dieser Stelle ergeben sich zwei Möglichkeiten. Sollte die Url noch nicht indiziert sein, so muss sie als erstes runtergeladen werden. Danach wird ein Dokument erstellt und durch den Indexer dem Index hinzugefügt. (siehe hierzu Kapitel 2.4.2) Als letztes muss noch die Datenbank angeglichen werden. (siehe Kapitel 2.4.1)

Die zweite Möglichkeit ist, dass die Url bereits indiziert wurde. Um ein Update der Url im Index durchzuführen, muss zuerst das entsprechende Dokument aus dem Index gelöscht werden. Dabei wird eine Kopie des Dokuments an den Controller geliefert. Dieser ersetzt nun das Kategorienfeld (siehe Kapitel 2.4.2), damit die neuen Kategorien auch enthalten sind. Danach wird das Dokument dem Index wieder hinzugefügt und die entsprechenden neuen Assoziationen zwischen Url und Kategorien werden in der Datenbank gesetzt.

Es gibt noch eine dritte Möglichkeit, die allerdings in Abbildung 2.4 nicht dargestellt ist. Die Url ist bereits hinzugefügt und es verändert sich auch bei den Kategorien nichts. Auch diese Möglichkeit wird anahnd der Datenbank überprüft. Sollte der Fall eintreten passiert gar nichts und die Erfolgsmeldung wird sofort an den Client gesendet. Es ist einfach vorstellbar, dass ein Benutzer ausversehen mehrmals das Hinzufügen auslöst. In solch einem Fall wollen wir nicht unnötig den Index blockieren. Allerdings gibt es auch hier eine Aussnahme. Sollte die Url ein gewisses "Alter" überschreiten wird sie

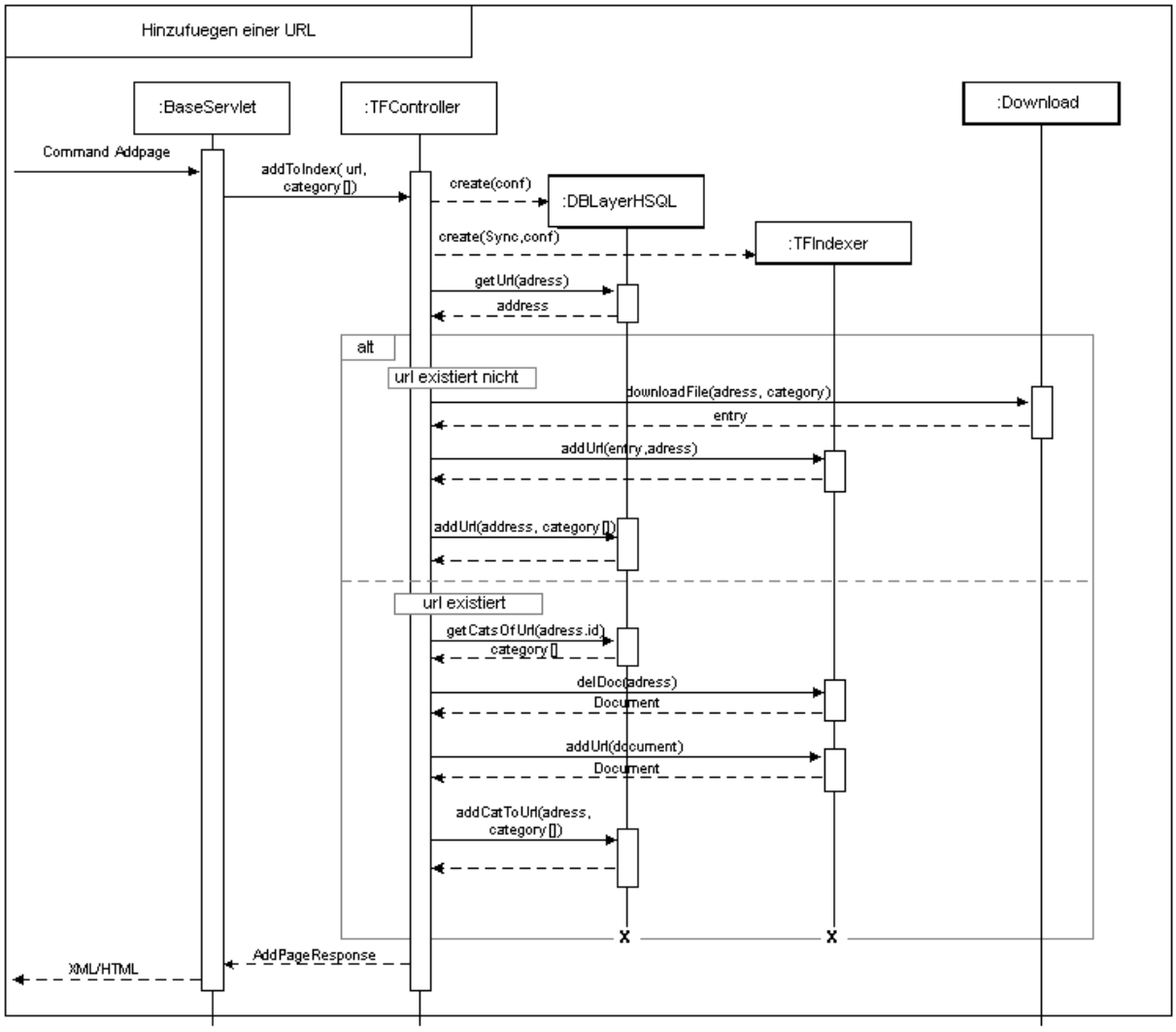

Abbildung 2.4: Indizieren einer neuen URL

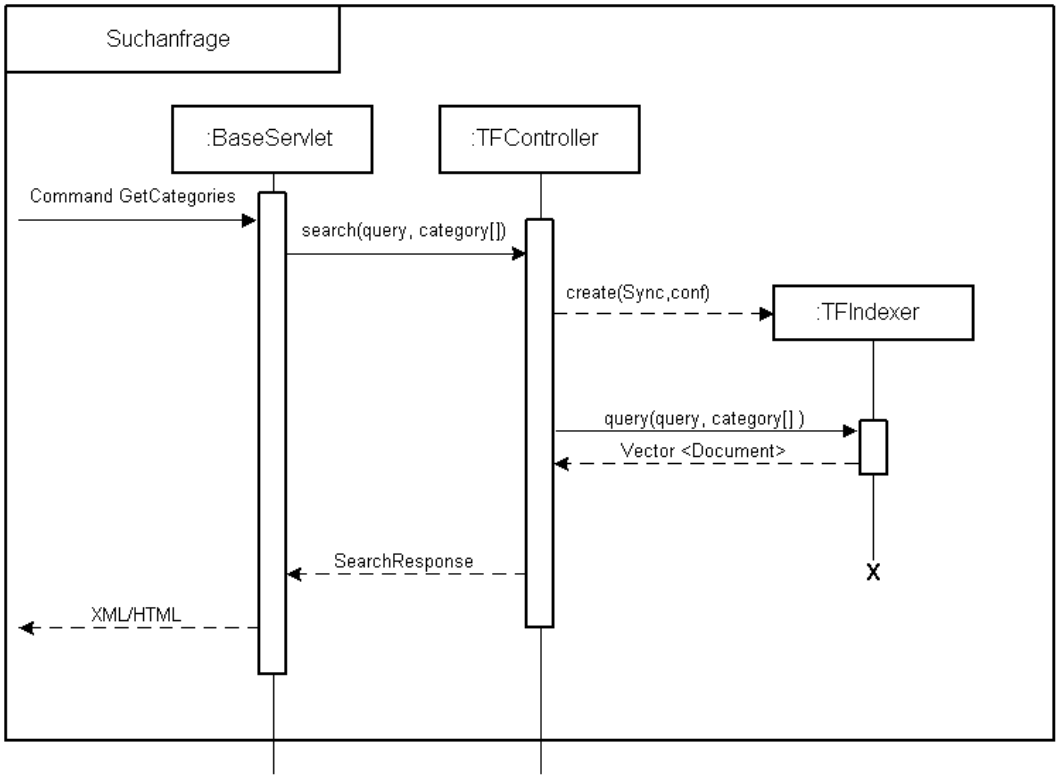

Abbildung 2.5: Suchanfrage an den Server

auf jeden Fall erneut heruntergeladen.

2.5.2 SearchRequest

Der am häufigsten benötigte Dienst des Servers ist die Suche nach Dokumenten. Wie ein Request grundsätzlich entgegengenommen wird ist bereits in Kapitel 2.5.1 beschrieben. Das Servlet (siehe Kapitel 2.3.1) erhellt den Request und ruft die entsprechende Funktion des Controllers (siehe 2.3.2) auf.

Eine Übersicht zum Ablauf einer Suchanfrage biete die Abbildung 2.5. Die Suchanfrage enthält zuerst einen vom User eingegebenen Text, der in der Lucene *QuerySyntax* formuliert ist. (siehe `QueryParser` in Kapitel 2.2.1) Zusätzlich wird ein Array von Kategorie-IDs übergeben. Die Kategorien wurden vom Nutzer am Client ausgewählt. Da alle Clients immer ihre Kategoriebäume aktuell halten, sind die IDs direkt verwendbar. Es gibt zur Zeit keine serverseitige Überprüfung der übergebenen Kategorien. Das bedeutet, wenn veraltete und somit "falsche" IDs in der Anfrage stehen, werden ent-

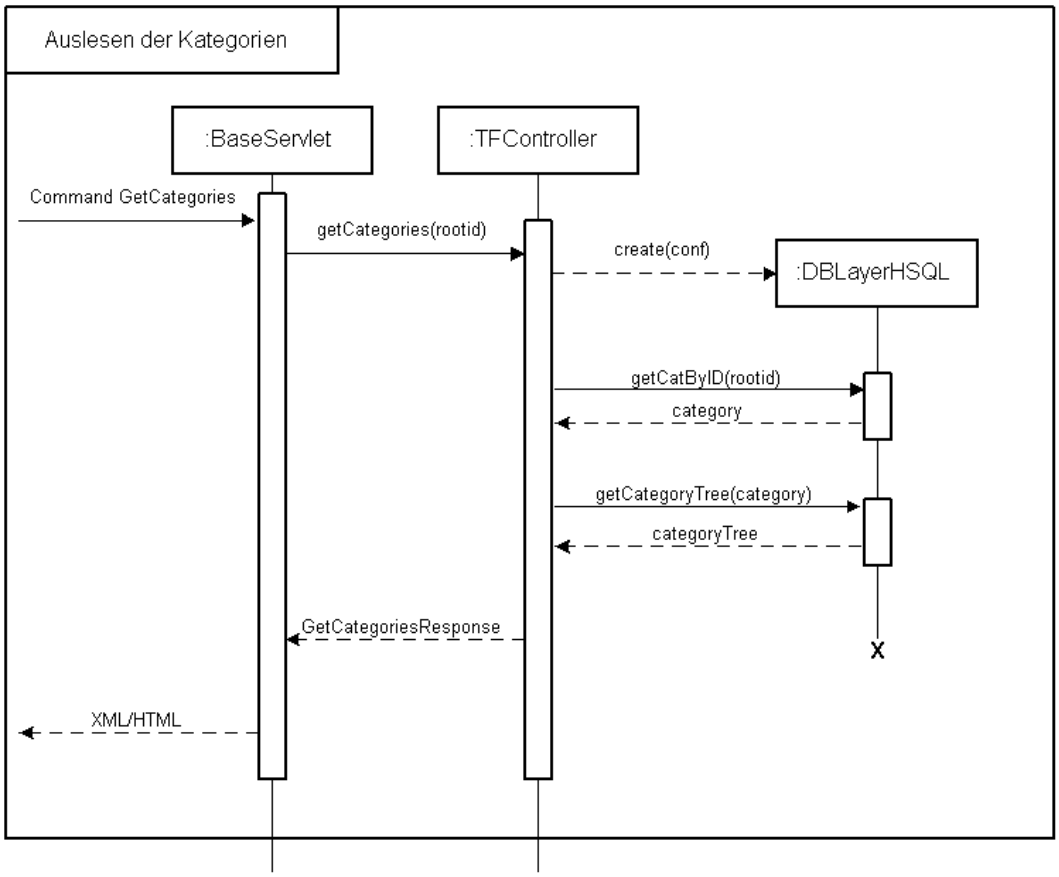

Abbildung 2.6: Abfragen der Kategorien

weder falsche oder keine Dokumente gefunden.

Die weitere Bearbeitung benötigt nur eine Anfrage an den Index. Hierzu wird ein Indexer (siehe Kapitel 2.3.3) erstellt. Dieser baut anhand der Parameter eine *Query* (Kapitel 2.2.1), die dann durch Lucene bearbeitet wird. Genauere Informationen zum Erstellen der *Query* befinden sich in Kapitel 2.4.2. Die Dokumente (Kapitel 2.2.1) , die der Index als Antwort generiert, werden durch den Controller ausgewertet und eine *Response* wird erstellt. Diese kann dann die HTML- oder XML-Antwort generieren, welche an den Client gesendet wird.

2.5.3 GetCategoriesRequest

In den vorangegangenen Kapiteln wurden Kategorien schon ziemlich ausgiebig erläutert. An dieser Stelle soll kurz erklärt werden wie die Kategorien ausgelesen und an den Client gesendet werden. Als Übersicht hierzu dient

die Abbildung 2.6.

Für diese Aufgabe muss der Controller nur zwei Datenbankabfragen machen und kann dann die entsprechende Response füllen. Als erstes werden dafür die vollständigen Informationen über die Kategorie, deren Unterbaum wir erstellen sollen, ausgelesen. Dies kann prinzipiell eine beliebige Kategorie sein, aber da wir zum derzeitigen Stand nur die Version von gesammten Bäumen speichern (siehe 2.4.1), ist derzeit auch nur eine Abfrage des gesammten Baumes sinnvoll. Die Repräsentation des Baumes in der Datenbank ermöglicht (siehe 2.4.3) es uns den gewünschten Baum mit einer etwas komplexeren Anfrage auszulesen. Die Anfrage:

```
SELECT a.id, a.left, a.right, a.name,
a.beschreibung,count(*) AS level FROM category AS a,
category AS b WHERE a.root\_id = 1 AND b.root\_id = 1
AND a.left BETWEEN b.left AND b.right GROUP BY a.id,
a.left, a.right, a.name, a.beschreibung ORDER BY a.left
```

ergibt bei dem Kategoriebaum aus Abbildung 2.3 folgende Antwort:

id	left	right	name	description	level
1	1	14	root	start des baumes	1
2	2	9	Kat2	...	2
3	3	4	Kat3	...	3
5	5	6	Kat5	...	3
7	7	8	Kat7	...	3
4	10	13	Kat4	...	2
6	11	12	Kat6	...	3

Diese kann leicht ausgwertet und eine Response erstellt werden.

2.6 Mögliche Verbesserungen im derzeitigen Server

1. Update indizierter Seiten ohne den Index lahmzulegen.

   Da lesender und schreibender Zugriff exklusiv voneinander getrennt ablaufen, kann ein Update zu einer langen Blockierung des Servers führen. Es gibt eine Möglichkeit dies zu umgehen. Lucene kann verschiedene Segmente anlegen wenn der Index vergrössert wird. Es ist möglich den lesenden Zugriff auf ein "altes" Segment zu gewähren, während ein neues Segment mit den aktualisierten Dokumenten angelegt wird. Der lesende Zugriff muss auf diese Weise nur blockiert werden, wenn die Segmente wieder Zusammengeführt werden.

2. Indizierung anhand der Sprache. Man könnte bei Webseiten den Analyzer anhand der gelieferten Sprachinformation aus dem HTTP-Header wählen und somit verhindern unnütze Worte wie z.B. Artikel zu indizieren.

3. Bessere Implementierung der Download Instanz. Wir nutzen zur Zeit eine eigene Klasse, die angeforderte Adressen herunterläd. Deshalb gibt es z.B. Probleme mit Frames. Die Implementierung muss verbessert werden oder es muss eine geeignete Bibliothek gefunden werden.

4. Finden einer geeigneten Lösung für gleiche Seiten mit unterschiedlicher URL.

   Man betrachte zum Beispiel die Adressen `http://xzy.de` und `http://xzy.de/index.html`. Diese sind wahrscheinlich genau dieselbe Seite. Sollten solche Adressen an den Server übergeben werden, würde er diese als zwei verschiedene Seiten behandeln. Auf diese Weise können redundante Dokumente entstehen.

   Ein Ansatz dies zu verhindern wäre es Hashwerte zu indizierten Seiten zu erstellen und zu speichern. Anhand von vergleichen der Hashwerte bei gleichem URL-Stamm könnte man das Problem umgehen.

2.7 Probleme

Während der Entwicklung des Servers sind keine grossen Probleme entstanden. Dennoch können vielleicht ein paar kleinere genannt werden.

- Versionskonflikte beim Java Environment

  Während der Entwicklung stellten wir unerwartet fest, dass einer von uns mit J2SE 1.4.2 und der andere mit J2SE 5.0 arbeitet. Daraufhin diskutierten wir kurz mit allen ob es sinnvoll wäre Kompatibilität zu 1.4.2 zu erhalten. Die Entscheidung viel schnell darauf auf 5.0 Basis weiterzumachen. Erstens hätte sonst Code umgeändert werden müssen und zweitens einigten wir uns das es Release 5.0 inzwischen schon lange genug gibt.

- Milestone 2

  Milestone 2 brachte eine völlige Umstellung auf Java und Tomcat. Hinzu kam die Erweiterung um eine Datenbank und tiefere Einarbeitung in Lucene. Es waren also erstens viele neue Techniken zu lernen und zweitens noch eine ganze Menge Probleme zu lösen. Hinzu kamen Prüfungen

und vorlesungsfreie Zeit. Dadurch kam es zu einem kleinerem Motivationstief, was erst mit näherrücken des Vortragtermins überwunden wurde.

Kapitel 3

Protokoll

Die in unserem Projekt von Anfang wohl grundlegendste Frage war die des Protokolls, also wie unsere Clients mit dem Server kommunizieren sollen. Auf jeden Fall sollte ein Protokoll spezifiziert werden, welches um spätere, jetzt noch unbekannte Features, möglichst leicht erweiterbar sein sollte. Es sollte ausserdem gut Dokumentiert und eindeutig sein, damit ähnliche Projekte, die die Idee hinter TeamFound ebenfalls implementieren wollen, **motiviert werden** dasselbe Protokoll zu benutzen. Somit könnten dann Server und Clients aus unterschiedlichen Projekten miteinander funktionieren und arbeiten.

Wir sind uns nicht sicher, wie weit wir diese Ziele mit Milestone 2 bereits erreicht haben, aber wir glauben auf dem richtigen Weg zu sein ;-)

3.1 Designentscheidungen und verwendete Techniken

Für unser Protokoll kamen prinzipiell viele verschiedene Möglichkeiten in Betracht.

Die erste, und einfachste Designentscheidung war, ein Protokoll zu benutzen, welches auf TCP/IP aufsetzt. Somit können schonmal alle Computer die über einen Internet-Zugang verfügen prinzipiell mit unseren Componenten kommunizieren.

Der zweite Schritt war die Festlegung unseres Protokolls nicht direkt auf TCP/IP aufzusetzen, sondern erst auf dem HTTP Protokoll. Diese Entscheidung viel auch sehr schnell und war für uns naheliegend, da wir von Anfang an vorhatten Browser-Plugins zu programmieren, und jeder Internet-Browser HTTP ebenfalls unterstützt.

Als weitere Grundlage für unsere Designentscheidungen war unser Wunsch, die Möglichkeit einen ganz einfachen Web-Client bereitstellen zu können, so

daß TeamFound auch ohne Installation einer Browser-Erweiterung ausprobiert und genutzt werden kann. Diese Entscheidung führte dann auch direkt zu der eigentlichen Basis unseres Protokolls:

HTML-Antworten des Servers Somit kann jeder Browser die Suchergebnisse unseres Servers direkt anzeigen.

Anfragen an Server mittels HTTP-GET Variablen Somit kann ein einfaches HTML-Formular korrekte Suchanfragen an unseren Server stellen.

XML-Antworten des Servers Diese Möglichkeit, konfigurierbar über die Suchanfrage, ermlglicht den Browser-Plugins, und somit dem End-Benutzer, die Darstellung der Suchergebnisse selber zu beeinflussen bzw. komplett anzupassen.

Die Implementierung des Protokolls, sowie auch alle anderen Komponenten von TeamFound, hatte wir in kleine Schritte unterteilt, unsere Milestones. So hat das Protokoll für Milestone 1 z.B. noch keine Unterstützung für XML-Antworten und Kategorien gehabt.

Auf Basis dieser gundlegenden Entscheidungen, haben wir also begonnen sowohl die HTTP-GET Anfragen zu spezifizieren als auch (ab Milestone 2) XML-Tags zu definieren, um die volle Funktionalität unserer Implementation nutzen zu können.

3.2 Interface Milestone 1

3.2.1 Seite hinzufügen

Request http://url/addpage.pl?url=http://yy.org/blabla.html

Antwort keine

3.2.2 Suchen

Request http://url/search.pl?keyword=zzz

Antwort HTML-Seite mit URLs gefundener Übereinstimmungen

3.3 Interface Milestone 2

Alle Anfrage-Parameter werden in Form von HTTP GET oder POST Variablen übertragen. Prinzipiell gibt es eine Unterscheidung, ob für jedes Kommando eine eigene URL verwendet wird, oder ob das Kommando in Form einer weiteren HTTP-GET Variablen mit übertragen wird. Unsere Implementation des Servers verwendet den zusätzlichen HTTP-GET Parameter `command`. In diesem Interface haben wir aber beide Möglichkeiten spezifiziert.

want=xml or html Die zu erwartende Antwort soll in xml oder html-format sein (default soll html werden, da dann einfache Link-Clients möglich sind)

version=2 Milestone-Version des Interfaces

command=search Das Kommando das ausgeführt werden soll. Dieses Argument kann wegfallen wenn für jedes Kommando eine eigene, gleichnamige Anfrage-URL existiert.

HTML-Antwort

Die Clients geben an ob Sie eine HTML oder eine XML Antwort erwarten.

Soll eine vollständige HTML-Seite zurückgeben, die direkt im Browser angezeigt werden kann. Die Return-Values wie bei einer XML-Antwort müssen in diesem Fall nicht zurückgegeben werden, sondern der Text sollte gleich eine entsprechende Meldung beinhalten.

XML-Antwort

<**response**> Umschliesst alle anderen xml-tags und gibt die XSD-Datei zum verifizieren der XML-Daten an

<**interface-version**> Gibt immer die Interface-Version an, in der die Antwort des Servers formuliert wurde

<**server**> Gibt Name und Versionsnummer des Servers an. Der Server der unter der URL https://developer.berlios.de/projects/teamfound entwickelt wird gibt den Namen TeamFoundßurück. Clones dürfen Ihren eigenen Namen natürlich frei wählen.

<**addpage**> <**search**> <**addcategory**> <**getcategories**> Diese Tags beinhalten die Antworten auf die gleichnamigen Anfragen an den Server. Pro Anfrage darf nur eines dieser Tags vorkommen.

<**return-value**> <**return-description**> Der Return-Value bzw. Fehler-Code falls etwas schiefging. Die Description wird nicht genauer spezifiziert, sollte aber semantisch mit dem aufgetretenen Fehler übereinstimmen.

<**project-counter**> Bei jeder Änderung des Kategorien-Baumes(soll der Server einen internen Zähler um eins inkrementieren. Der aktuelle Wert soll bei jeder XML-Antwort mit übertragen werden. (Dann weiss die Toolbar wann sie selber die Kategorien neu vom Server abfragen muss.)Jedes Projekt fuehrt einen Counter, somit muss der Client nur die fuer ihn intressanten Bäume beachten.

```
<response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="teamfound-interface-milestone2.xsd">

 <interface-version>2</interface-version>

 <return-value>0</return-value>
 <return-description>OK</return-description>

 <project-counter>
   <project>
     <projectID>0</projectID>
     <count>3</count>
   </project>
   <project>
     <projectID>3</projectID>
     <count>5</count>
   </project>
 </project-counter>

 <category-counter>54</category-counter>

 <server>
  <name>TeamFound</name>
  <version>0.2</version>
 </server>

 <addpage>
 </addpage>
```

```
 <search>
 </search>

 <addcategory>
 </addcategory>

 <getcategories>
 </getcategories>

</response>
```

Return Codes Die Return-Codes stehen immer in den Tags <return-value>xx</return-value>. Die Description ist optional.

0 Alles ok

1 Fehler, konnte URL nicht finden

2 Ungültige Anfrage (Pflicht-Parameter fehlen oder haben die Länge null)

3 Inkompatible Interface-Version (die Anfrage hat eine Interface-Version benutzt die der Server nicht unterstützt)

4 Kategorie existiert schon (beim hinzufuegen einer neuen Kategorie)

5 Kategorie nicht gefunden (beim suchen nach einer bestimmten Kategorie)

-1 Anderer Fehler

Die Return-Descriptions sind frei wählbar, sollten aber dem Return-Code semantisch entsprechen ;-)

Die vollständige Spezifikation des Interface Milestone 2 befindet sich im Anhang A.1 auf Seite 46.

Kapitel 4

Clients

4.1 Web-Client

Der Web-Client ist ein einfaches HTML-Formular (<form>..</form>), welches Anfragen an den Server sendet und die Antwort direkt im Browser anzeigt. Dieser Client ist von daher praktisch, da man keine extra Erweiterung für den Browser installieren muss, und von daher unpraktisch, da man die URL einer neu hinzuzufügende Seite immer zuerst manuell kopieren, dann die Seite des Web-Clients laden und schliesslich die URL manuell eintragen muss.

Als interessante Erweiterung, bot sich hier allerdings ein einfacher Javascript-Link an, der, als Bookmark in den eigenen Browser eingefügt, die Nachteile des Web-Clients zum Teil ausräumt:

```
javascript:location.href='http://serverurl/addpage.pl?
url='+encodeURIComponent(location.href)
```

Mit diesem einfachen Link, kann das lästige Kopieren umgangen werden, indem über Javascript die URL der aktuell im eigenen Browser angezeigten Seite, durch einen einfachen Click auf diesen Bookmark, direkt an den TeamFound-Server gesendet wird. (Beispiel ist für Milestone 1)

Abbildung 4.1: Web-Client Screenshot

4.1.1 Implementation

Die *extrem einfache* Implementation des Web-Clients für eine Suchanfrage entsprechend Milestone 2 sieht wie folgt aus:

```
<form action="serverurl">
search: <input type="text" name="keyword">
<input type="hidden" name="want" value="html">
<input type="hidden" name="version" value="2">
<input type="hidden" name="command" value="search">
<input type="submit">
</form>
```

Bei dem Design der TeamFound-Interfaces war diese Möglichkeit der HTML-Formulare die entscheidende Grundlage. Alle Anfragen an den Server können mit HTML-Formularen ohne Zusätze (wie Javascript, Applets, Extensions, etc.) generiert werden.

4.2 Firefox-Toolbar

Der Mozilla Firefox lässt sich durch sogenannte *Extensions* sehr einfach erweitern. Als grundlegende Techniken werden dafür XUL (XML User Interface Language) und Javascript benötigt. Desweiteren gibt es eine exakt vordefinierte Verzeichnisstruktur sowie eine Beschreibungs-Datei (`install.rdf`) im RDF (Resource Description Framework) Format, umd die Erweiterung automatisiert installierbar in Firefox-Browsern zu machen.

4.2.1 Dateien

Die Verzeichnisstruktur der TeamFound Firefox Extension sowie alle enthaltenen Dateien sind:

```
chrome.manifest
install.rdf

content/
- overlay.js
- overlay.xul
- settings.js
- settings.xul

defaults/
```

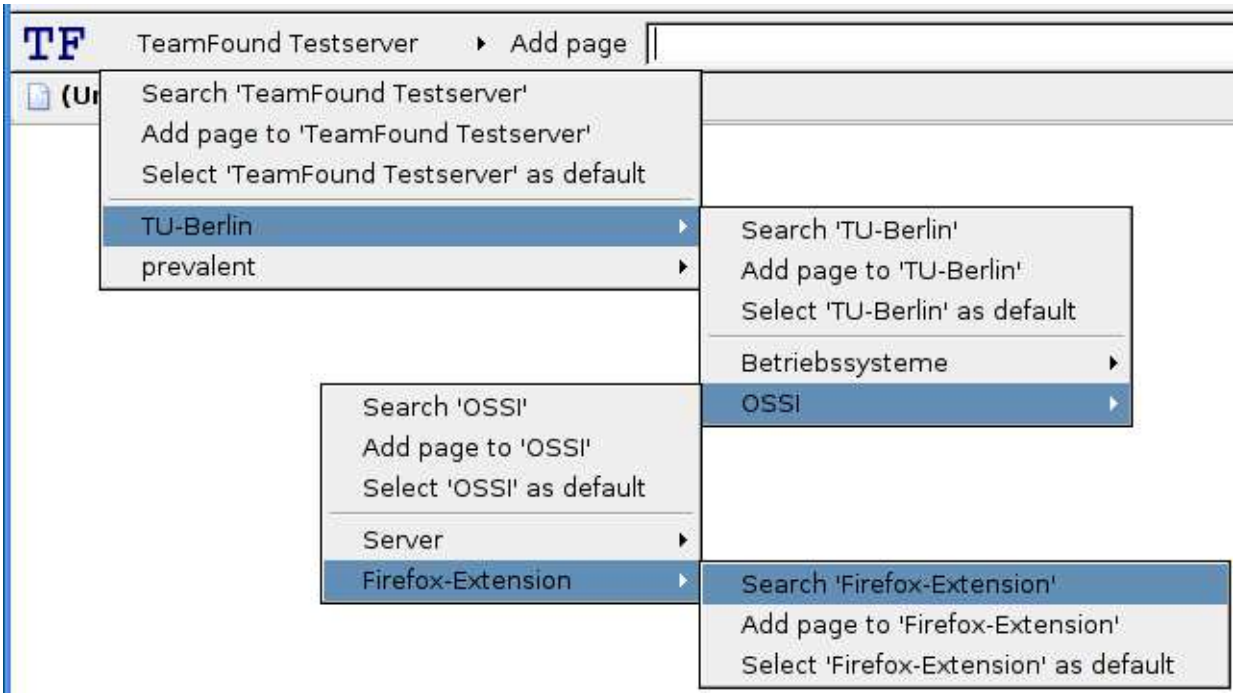

Abbildung 4.2: Firefox-Toolbar mit Kategorien-Baum (v0.8, Milestone 2)

```
- preferences/
  - settings.js

skin/
- icon.png
- logo_tf_32x32.png
- overlay.css
- search_h.html
- search_v.html
- xmlstyle.css
```

Die eigentliche Funktionalität steckt dabei in dem Unterverzeichnis content:

overlay.xul Definiert das Aussehen der Toolbar

overlay.js Definiert die Funktionalität der Toolbar

settings.xul Definiert das Aussehen des Einstellungen-Dialogs

settings.js Definiert die Funktionalität des Einstellungen-Dialogs

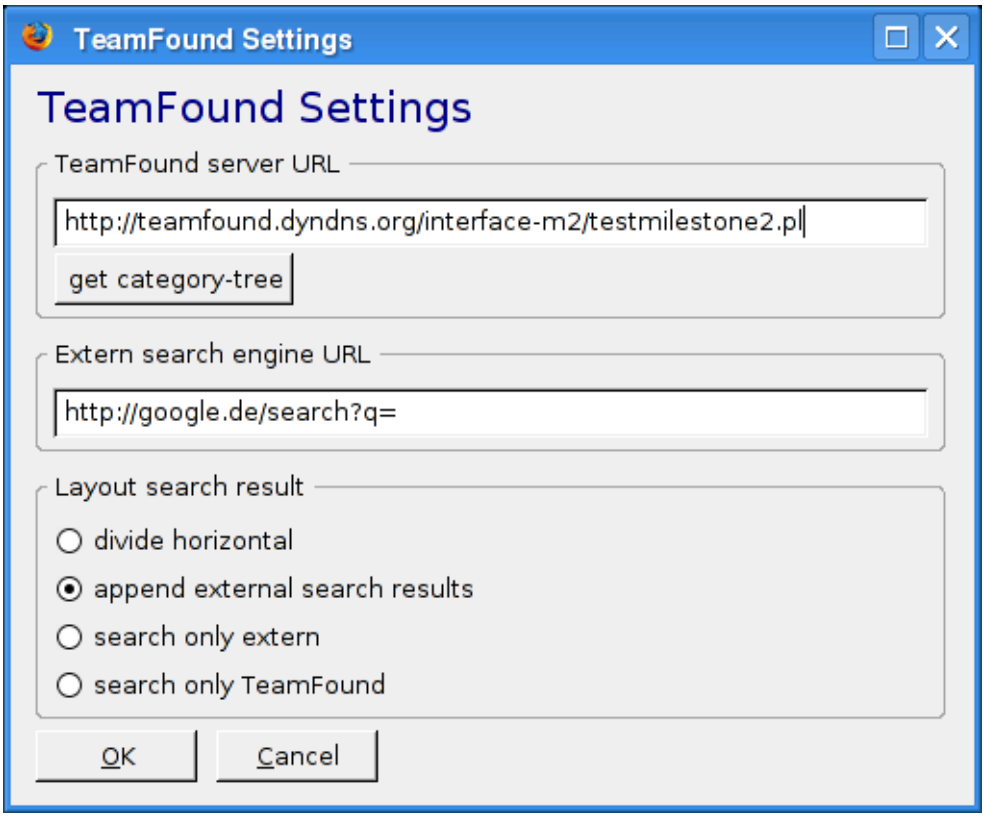

Abbildung 4.3: Firefox-Toolbar Settings Dialog (v0.8, Milestone 2)

4.2.2 Implementation

Das originale Changelog in englischer Sprache der Firefox-Toolbar befindet sich im Anhang B.2 auf Seite 64.

Die wichtigsten Funktionen der Toolbar sind:

onLoad Initialisiert die Toolbar: Lädt die gespeicherten lokalen Einstellungen und fragt den Kategorien-Baum vom Server ab. Diese Funktion wird über einen Event-Handler automatisch vom Firefox-Browser aufgerufen, sobald dieser geladen wird.

loadCategories Stellt einen XMLHttpRequest an den Server, und definiert `onLoadCategoriesFinished` als Callback-Funktion der Anfrage.

onLoadCategoriesFinished Wird aufgerufen wenn der Server eine Antwort auf das Kommando `getcategories` gibt. Auf Basis dieser Antwort wird das Kategorien-Menü gelöscht und die Basis-Informationen wie

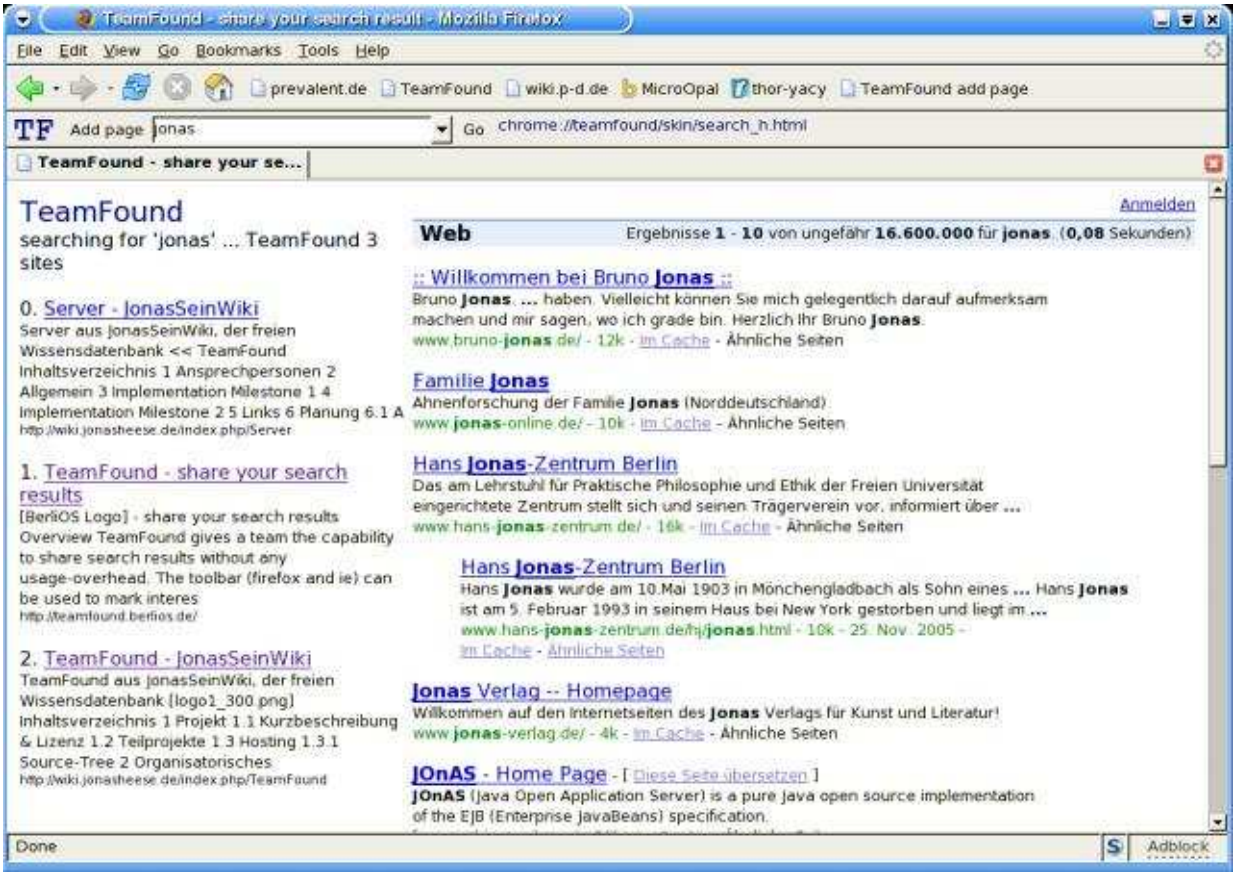

Abbildung 4.4: Firefox-Toolbar Suchergebnis (v0.4, Milestone 1)

der Server-Name aus der XML-Antwort ausgelesen. Danach wird die Funktion `addCategory` aufgerufen, die rekursiv alle Kategorien durchläuft und einen entsprechenden Menü-Baum in der Toolbar aufbaut.

onSettings Event-Handler wenn auf den 'TF' Button oben links in der Toolbar geklickt wird. Diese Funktion öffnet den Einstellungen-Dialog.

onSearch Diese Funktion startet eine Suchanfrage, und wird aufgerufen wenn in dem Kategorien-Baum 'Search' oder der 'Go' Button angeklickt werden. Die <return>-Taste wenn der Fokus auf dem Eingabefeld der Kategorien ist ruft ebenfalls diese Funktion auf. OnSearch überprüft daraufhin die Einstellungen, und leitet die Suchanfrage entsprechend an die Funktionen `myGotoUrl` oder an `myTeamFoundSearch` und `myExternSearch` weiter. Je nach Einstellung wird das HTML-Template zur Anzeige der Suchergebnisse `search_h.html` für eine horizontale Verteilung bzw.

`search_v.html` für eine vertikale Verteilung der Suchergebnisse geladen.

myGotoUrl Lädt ganz einfach die angegebene URL im aktiven Tab des Browser.

myTeamFoundSearch Erstellt einen XMLHttpRequest and den TeamFound Server entsprechend den übergebenen Suchwörtern und registriert `onTeamFoundSearchFinished` als Callback-Funktion.

myExternSearch Erstellt einen XMLHttpRequest an eine externe Suchmaschiene entsprechend den übergebenen Suchwörtern und registriert `onExternSearchFinished` als Callback-Funktion.

onAddPage Erstellt einen XMLHttpRequest an den TeamFound Server um die aktuell im Browser angezeigte Seite hinzuzufügen und registriert `onAddPageFinished` als Callback-Funktion.

onAddPageFinished Wertet die Antwort des Server aus und zeigt dem Benutzer eine entsprechende Meldung an.

onExternSearchFinished Such in dem (schon geladenen) Template nach der HTML-Tag-ID "`teamfound-result-two`" und fügt an dieser Stelle die Antwort als `innerHTML` ein.

addCategory Fügt an das übergebene Menu-Element die Einträge "`Search`", "`Add`" und "`Select as default`" für die ebenfalls übergebene Kategorie an. Für jede existierende Unterkategorie wird ein `menupopup`-Eintrag erstellt und diese Funktion jeweils rekursiv wieder aufgerufen.

onTeamFoundSearchFinished Sucht in dem Ergebnis-Template nach der HTML-Tag-ID `teamfound-result-one` und fügt das TeamFound Ergebnis an dieser Stelle als `innerHTML` ein.

4.3 Internet Explorer-Toolbar

Auch beim Internet Explorer stand zunächst die Frage wie dieser überhaupt zu erweitern ist. Google und Yahoo haben mit ihren Toolbar's vorgemacht das es geht, nur das wie war zu lösen.

Erste Anlaufstelle um der Problemlösung näher zu kommen war CodeProject [1]. Mit 2,8 Mio Mitgliedern ist sie die größte Community Seite für die

[1] www.codeproject.com

Windows Entwicklung. Eine Suche nach "IE Toolbar" führte zu einer .NET basierten Bibliothek für die Entwicklung von BandObjects, also Band Objekten [2]. Das mitgelieferte Beispiel funktionierte auf Anhieb und es wurde begonnen die Extension zu implementieren.

4.3.1 Custom Explorer Bars, Tool Bands, and Desk Bands

Es gibt verschiedene Möglichkeiten eigene Komponenten in den Internet Explorer zu integrieren. Im folgenden werden einige Konzepte vorgestellt.

Explorer Bars Ein zentrales Konzept zur Erweiterung des Internet Explorers, sind Explorer Bars. Explorer Bars sind Teilbereiche der Oberfläche des Explorers, welche nicht zur Renderfläche der HTML Seite gehören. Explorer Bars wurden erstmalig mit dem Internet Explorer 4.0 eingeführt und bieten eine sehr flexible Möglichkeit den Explorer zu erweitern. Derzeit werden Funktionen wie Favorieten und die Suche innerhalb einer Explorer Bar dargestellt. Die Darstellung von Explorer Bars kann entweder horizontal oder vertikal erfolgen und sie werden über den Menu Ansicht - Explorerleisten ein- und ausgeblendet.

Tool Bands Die Werkzeugleiste des Internet Explorers ist eine sogenannte "Rebar", also ein Container der wiederum mehrere Werkzeugleisten enthalten kann. Diese Werkzeugleisten werden Toolbands genannt. Erstmalig wurde das Konzept mit dem Internet Explorer 5.0 eingeführt um die Radio Leiste zu realisieren. Mittlerweile gibt es verschiedene Toolbands wie die Adressleiste und auch Fremdanbieter wie Google und Yahoo nutzen das Konzept.

Deskbands Diese Komponenten verfolgen den selben Ansatz, haben aber primär nichts mit dem Internet Explorer zu tun. Sie dienen dazu den Desktop um weitere Leisten zu erweitern. Berühmtester Vertreter dieser Kategorie ist die Adressleiste. Sie kann ausser im Explorer und Internet Explorer auch in der Taskleiste erscheinen.

4.3.2 Entwicklung der Band Objekte

Die Entwicklung dieser Komponenten basiert, wie ein sehr großer Teil der Windowsentwicklung auf COM. Das Component Object Model von Microsoft ist eines der ältesten Komponentensysteme und bildet zusammen mit dem

[2] http://www.codeproject.com/csharp/dotnetbandobjects.asp

Windows API die Grundlage für die Entwicklung von Windowssoftware. Da die Toolbar aus solchen Bandobjekten besteht, müssen also COM Objekte geschrieben werden.

4.3.3 COM, Schnittstellen, Klassen, Registry

Im folgende soll ein kurzer Überblick über COM gegeben werden, der die Schritte auf dem Weg zu Entwicklung der Band Objects erklärt.

COM Schnittstellen Zentrales Konzept von COM ist das Interface. In der COM Terminologie ist eine Schnittstelle, eine nach der COM Spezifikation aufgebauten, Struktur mit Platzhaltern für Funktionszeiger. Diese Erklärung macht bereits deutlich auf welchem Level COM ursprünglich definiert worden ist. Schnittstellen können von Komponenten erfragt werden, und anschließend über die Methodenzeiger entsprechende Funktionen aufgerufen werden. In einer einfachen - praktisch puren - COM Application muss man das Aussehen der Schnittstellen genau kennen. Später kam ein Feature mit dem Namen "OLE Automatisierung" hinzu, welches ähnliche Konzepte wie Java Reflection bietet und somit den Scriptsprachen den Zugang zu COM ebenete. Jede Schnittstelle wird eindeutig über einen GUID - Global Unique Identifier - indentifiziert.

COM Komponenten Eine Komponente in der COM Terminologie ist eine abgeschlosse binäre Einheit, welche Funktionalität über eine Schnittstelle bereitstellt. Komponenten sind also in Programmen (Executables) oder Bibliotheken (Dynamic Link Libraries) enthalten.

Registry Normalerweise, auch wenn das problemlos möglich wäre, wird eine COM Bibliothek nicht direkt referenziert und geladen. Stattdessen nutz man Funktionen der COM API um eine Instanz einer Komponente mit einer bestimmten Schnittstelle zu erhalten. Dazu übergibt man der Funktion die GUID der Schnittstelle welche man nutzen möchte. Diese Funktion bemüht die Windows Registry um die Bibliothek zu finden in der die Komponente mit der Schnittstelle implementiert ist, lädt diese, erzeugt die Komponente und gibt die Schnittstelle zurück. Damit das funktioniert, ist jede COM Komponente vor ihrer Benutzung in der Registry einzutragen.

4.3.4 Band Objekte und COM

Alle Band Objekte sind wiederum COM Komponenten, welche bestimmte Schnittstellen implementieren müssen. Diese Schnittstellen werden vom

Internet Explorer erwartet und abgefragt. Für die Entwicklung von Band Objekten müssen in jedem Fall die folgenden Schnittstellen implementiert werden:

IUnknown Das ist die Basisschnittstelle welche jedes COM Objekt implementieren muss. Sie bietet Funktionen zum Abrufen weiterer Schnittstellen sowie zur Referenzzählung.

IClassFactory Bietet Funktionen zum Erzeugen neuer Instanzen von COM Klassen.

IDeskBand Das ist die Basisschnittstelle für alle Band Objekte. IDeskBand erbt von IDockingWindow welches wiederum von IOleWindow erbt. Anhand der Namen der Schnittstellen lassen sich ungefähr die Funktionen ableiten. IOleWindow liefert das Fenster (jedes Element in Windows ist ein Fenster) an sich, IDockingWindow sorgt für das Anzeigen und Deaktivieren und IDeskBand liefert Informationen über das Band Objekt an sich.

Mit Hilfe von GetBandInfo kann der Internet Explorer ermitteln wie die Leiste heist, wie sie dargestellt werden soll und welche Größen sie haben darf.

IObjectWithSite Über diese Schnisttelle hat man die Möglichkeit mit dem Container - also mit dem Internet Explorer zum Beispiel - des Band Objektes zu kommunizieren. Der Internet Explorer ist selbst ebenfalls ein COM Objekt welches diverse Schnittstellen implementiert. Über die SetSite Funktion weist er dem Band Objekt die eigene IUnknown Schnittstelle zu.

IPersistStream Diese Schnittstelle wird vom Container verwendet um den Zustand des Band Objekts zu speichern oder zu laden. Das Band Objekt implementiert diese Schnittstelle um sich selbst in den übergebenen Stream zu serialiasieren oder zu deserialisieren

Daneben gibt es noch einige weitere Schnittstellen z.B. für Benutzereingaben, auf welche hier jedoch nicht näher eingegangen werden soll.

4.3.5 Registrierung der Band Objekte

Jedes Band Objekt muss wie bereits erwähnt, als COM Objekt registriert werden. Darüber hinaus muss dem Internet Explorer nocht mitgeteilt werden, das er eine Toolleiste zur Verfügung hat. Dies geschieht ebenfalls über einen Eintrag in der Windows Registry.

4.3.6 Die BandObject Library

Alles was im Abschnitt COM und Band Object Entwicklung beschrieben worden ist, nimmt einem die Band Objekt Library ab. Der ganze "schmuzige" COM Teil wird durch eine CSharp Klasse gekapselt welche von UserControl erbt und all die Schnittstellen implementiert. Was dem Entwickler der Toolbar bleibt ist eine Klasse zu erstellen welche von der BandObject erbt. Da BandObject selber von UserControl erbt, steht einem auch der visuelle Designer zum erstellen der Toolbar zur Verfügung. Die Funktionalität der Toolbar kann also bequem in CSharp erfolgen.

4.3.7 Installation

Installation

Im die Internet Explorer Toolbar zu verwenden, wird der Internet Explorer benötigt (mindestens v5.0) sowie das .NET Framework in der Version 2.0.

Für die Installation auf einem Client PC kopiert man alle im Paket enthaltenen Quellen in einen Ordner. Die Toolbar lässt dann einfach mit dem Install.cmd installieren. Im wesentlichen passiert dabei folgendes:

Alle anderen Bibliotheken werden ebenfalls im globalen Assemblycache registriert.

Die Bibliotheken BandObject.dll und Teamfound.IE.dll werden als COM Objekt registriert und im globalen Assembly Cache registriert. Alle anderen Bibliotheken werden ebenfalls im globalen Assemblycache registriert.

4.3.8 Entwicklung

Dateien Folgende Dateien sind im Projekt enthalten:

Interna für die Verwaltung der Daten

```
AssemblyInfo.cs
Category.cs
Win32.cs
ComInterop.cs
Configuration.cs
Controller.cs
Event/CategorySelectEventArgs.cs
Event/SearchEventArgs.cs
```

Darstellung - Formulare und Controls

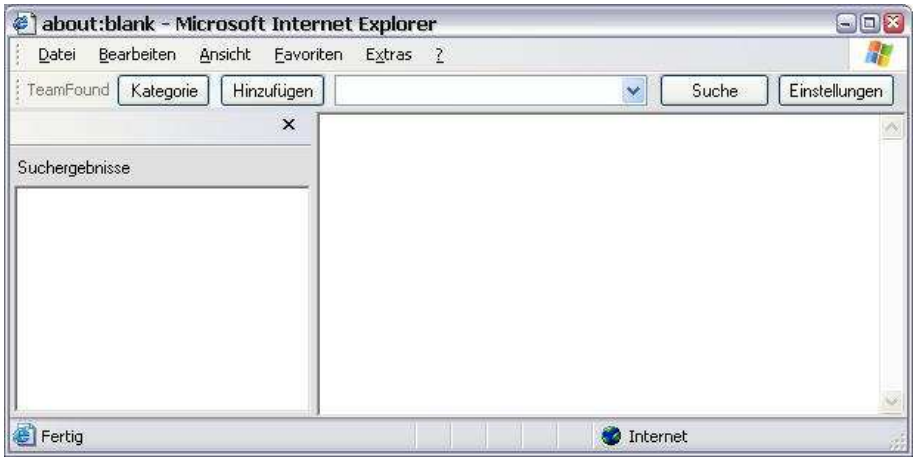

Abbildung 4.5: Internet Explorer Toolbar (v0.2)

```
CategoryTreeView.cs
CategoryTreeView.Designer.cs
FrmAddCategory.cs
FrmAddCategory.Designer.cs
FrmAddPage.cs
FrmAddPage.Designer.cs
FrmSettings.cs
TeamFoundBar.cs
TeamFoundResultBar.cs
TeamFoundResultBar.Designer.cs
```

Die Toolbar ist nach dem Modell View Controller Pattern aufgebaut. Es gibt einen View - oder besser zwei Views, ein Modell - also Datenklassen welche die Daten für den Client halten und einen Controller der die Logik enthält.

Views Es gibt zwei Views. Einen für die Eingabe der Suchdaten, dies ist die eigentliche Toolbar, und einen für die Darstellung der Ergebnisse.

Die Buttons auf der Toolbar rufen vier verschiedene Funktionen im Controller auf.

Controller Der Controller enthält die Logik des Clients bzw. delegiert die Aufrufe an den Server, welcher wiederum Logik enthält. Es gibt vier wesentliche Funktion im Controller. Man könnte auch von Anwendungsfällen reden.

Search Ruft den Server mit den übergebenen Suchworten und den Kategorien auf. Sobald das Suchergebniss verfügbar ist, wird ein Ereignis ausgelöst. Der View für die Suchergebnisse, reagiert darauf und stellt diese dar.

AddPage Diese Funktion zeigt ein Fenster zur Auswahl einer Kategorie und zur eingabe einer URL an. Wird dieses Fenster erfolgreich geschlossen, also mit OK statt mit Abbrechen oder dem X, wird das addpage Kommando auf dem Server aufgerufen und so die Seite dem Index hinzugefügt.

AddCategorie Ruft das addcategorie Kommando auf dem Server auf, und legt damit eine neue Kategorie an.

EditSettings Zeit ein Fenster zur Bearbeitung der Server URL an. Bei Erfolg, wird das Ergebniss in der Registry gespeichert.

Modell Das Modell besteht aus der Klasse Category. Ein Category Objekt kann selber wieder Kategorie Objekte enthalten. Somit können die Kategorien vom Server abgeholt und im Speicher gehalten werden. Die Views können über die Eigenschaft Categories, des Controllers auf die Kategorien zugreifen und Anzeigen. Der Controller wiederum aktualisiert die Kategorien wenn es erforderlich ist.

4.3.9 Fazit Internetexplorer Erweiterung

Es war sehr spannend zu sehen wie unterschiedlich die Entwicklung der Erweiterungen für den Firefox und den Internet Explorer war. Während die eine Extension mit einem einfachen Editor entwickelt werden konnte, wurde für die andere Visual Studio 2005 verwendet.

Beide Varianten haben ihre eindeutigen Vorteile. Beim Firefox liegen die in der einfachen Entwicklung und der einfachen Installation. Man brauch keine besonderen Werkzeuge um den Browser zu erweitern.

Bei der Internet Explorer Variante liegen die Vorteile in der Flexibilität. Da ich an keine Sprache gebunden bin um die Erweiterung zu entwickeln, kann ich auch beliebe Features realisieren. Auch habe ich Zugriff auf die gesamte Umgebung, innerhalb und ausserhalb der Browsers. Somit ergeben sich natürlich noch viel mehr Möglichkeiten.

Ein weiterer Vorteil liegt in der besseren Fehlersuche. Mit Visual Studio steht mir ein umfangreicher Debugger zur Verfügung, der auch vor dem IE nicht halt macht. Fehler können so natürlich schneller gefunden werden.

Kapitel 5

Projektorganisation

Gleich zu Anfang des Projekts haben wir uns auf feste Kommunikationswege geeinigt. Diese bildeten, neben regelmäßigen Treffen an der Uni, die Basis unserers Entwicklungsprozesses.

5.1 Kommunikation

1. Entwicklungs-Arbeit zu Standards, Interfaces, Realisierungen etc. finden auf Wiki-Seite `http://wiki.jonasheese.de/index.php/TeamFound` statt.

   Dadurch kann jeder immer schnell und einfach Nachschlagen und sich informieren. Ausserdem sind Änderungen auch gleich für alle sichtbar.

2. Mailingliste `teamfound-development@lists.berlios.de` - zum subscriben/unsubscriben folgenden Link benutzen:
   `https://lists.berlios.de/mailman/listinfo/teamfound-development`

   Über die Mailingliste unterrichten wir uns untereinander immer über Fortschritte oder neue wichtige Aufgaben. Ausserdem werden Terminabsprachen so erledigt.

3. Diskussionen bezüglich der Vorlesung finden auf der ossi Mailingliste `ossi@insel.cs.tu-berlin.de` mit Subject: TeamFound: xxx statt.

4. Chat - QuakeNet #teamfound

   Vor allem bei der Arbeit an der Implementierung hat es sich extrem bezahlt gemacht auch direkt mit den anderen Mitgliedern neu auftretende Fragen klären zu können. Ausserdem konnten durch den direkten Chat kleinere Aufgaben schnell untereinander aufgeteilt werden.

5.2 Source-Code Verwaltung

Um mit mehreren Personen gleichzeitig an gleichen Teilen Source-Codes zu Arbeiten, wird eine Sourcecode-Verwaltung benötigt. Es gibt viele verschiedene Internet-Seiten, die ‚für freie Software, kostenlos einen solchen Service anbieten. Wir haben uns für Berlios (`http://developer.berlios.de/`) entschieden, da hier im Gegensatz zu Sourceforge (`http://sourceforge.net`) SVN (Subversion) anstelle von CVS (Concurrent Version Control) angeboten wird. Berlios beschreibt sich selber auf seiner Homepage als:

BerliOS Developer ist ein freier Dienst für Open Source Entwickler und bietet einfachen Zugang zum Besten aus CVS/SVN, Mailinglisten, Bug-Tracking, Diskussionsforen, Aufgabenverwaltung, Webhosting, dauerhafte Dateiarchivierung, Backups und vollständige Verwaltung per Web-Interface.

Die Details des TeamFound-Projekts auf Berlios sind:

Project page https://developer.berlios.de/projects/teamfound

Project full name TeamFound

Project unix name teamfound

SVN server svn.berlios.de[1]

Shell server shell.berlios.de

Web server http://teamfound.berlios.de

5.3 Teilprojekte

Als zweite, grundlegende organisatorische Entwicklungsplattform, haben wir uns für ein MediaWiki entschieden, welches Jonas freundlicherweise auf einem seiner Server aufgesetzt hatte. Um die Entwicklung in klare Bahnen zu lenken, haben wir dort verschiedene Teilprojekte erstellt, die mehr oder wenig unabhängig voneinander bearbeitet werden konnten. Eher weniger von einander unabhängig war und ist besonders das Teilprojekt Interface-Spezifikation. Dieses wurde bei beginn eines neuen Milestones jeweils als erstes Ausgearbeitet, damit die Server- und Client-Implementationen zeitgleich durchgeführt werden konnten.

Milestones Aktueller Stand: Milestone 1 fertig, Milestone 2 fast fertig

[1]Details zum SVN: `https://developer.berlios.de/svn/?group_id=5199`

Toolbar als Extension für Firefox Aktueller Stand: Toolbar 0.7 funktionsfaehig (Milestone 1), Toolbar Milestone 2 fast fertig

Toolbar als Extension für Internet Explorer Aktueller Stand: Toolbar 0.1 fertig (Milestone 1)

Server Aktueller Stand: Version 0.1 einsatzbereit (Milestone 1), Betaversion für Milestone2 einsatzbereit

Interface-Spezifikation Aktueller Stand: Interfaces fuer Milestone 1 fertig, Milestone 2 fertig

Präsentation Aktueller Stand: Enduser-Homepage angefangen, Logo vorgeschlagen

Link als Client Aktueller Stand: Milestone 1 fertig: http://hqpm.dyndns.org/tf/

Kategorien Aktueller Stand: Ein Kategorienbaum, der über den Server bei allen Clients auf dem aktuellen Stand gehalten wird.

5.4 Meilensteine

Einer der Grundsätze unserer Entwicklung war, mlglichst schnell einen funktionierenden Prototypen zu erstellen. Außerdem wollten wir stets mit lauffähigen Versionen arbeiten und keine großen "Sprünge" planen. So sollte die allgemeine Motivation sowie eine zeitgerechte Fertigstellung gewährleistet werden. Auf Basis dieser Wünsche haben wir den ersten Meilenstein auch entsprechen klein definiert:

5.4.1 Meilenstein 1

1. Lauffähige Versionen der Toolbars und des Servers
2. Über Toolbar einzelne HTML-Seiten hinzufügen
3. Server soll diese HTML-Seiten indizieren und durchsuchbar machen
4. Über Toolbar soll der Server zum durchsuchen der indizierten Seiten nach Schlüsselwörtern gebracht werden und ein Liste der Links als Web-Seite zurückliefern

```
[ Hinzufuegen-Button ] [<--   textfeld   -->] [ Suchen-Button ]
```

5.4.2 Meilenstein 2

- Kategorien-System
- Konfigurations-Dialog in Toolbars für Server-Adresse
- Zusätzlich das Suchergebnis von google mit den gleichen Key-Wörtern anzeigen
- Interface-Version Milestone 2 implementieren
- XML oder HTML als Antwort von Server anfordern

5.4.3 Meilenstein 3

- User-Management
- Rating Mechanismen

5.4.4 Ideen für zukünftige Versionen

- Mehrere Server in Toolbar einstellbar
- Mehrere Server über Toolbar in einem rutsch durchsuchen und die Ergebnisse kombinieren
- Web-Interface für Server (und Server-Administration)
- mehrere Sprachen unterstützen
- Die Toolbar koennte mit etwas eingeschraenkter Funktionalitaet komplett durch links mit javascript ersetzt werden (del.icio.us macht das). Also hinzufuegen kann einfach ein link in der Link-Leiste sein, und zum suchen muss man dann halt auch auf die Suchseite des servers gehen

Kapitel 6

Fazit & Ausblick

Mit dem Abschluss des zweiten Milestones während des Projektes hat TeamFound die wichtigsten Basisfeatures erreicht, der Einsatz als Teamsuchmaschine ist bereits möglich. Platz für weitere Entwicklungen ist natürlich reichlich vorhanden, seien dies eine Benutzerverwaltung, Indexierung anderer Dokumenttypen wie PDF oder PostScript oder auch die Generierung eines Kataloges der eingetragenen Links. TeamFound hat aber schon im bestehenden Umfang gezeigt das sich eine derartige Suchmaschine mit freier Software ohne weiteres Umsetzen lässt. Die nötige Planung und Recherchearbeit vorrausgesetzt, haben sich die richtigen Komponenten gefunden um alle Teile der Implementierung abzudecken, seien das die - wirklich aufwändige - Anforderung an einen Volltext-Index, viele kleinere Bibliotheken für XML, Datenbankzugriff oder eine Java-Servlet-Container-Implementierung um TeamFound schliesslich ins Internet zu bringen. Es wurde sehr genau darauf geachtet möglichst nur vorhandene Komponenten zu verbinden, nur dort eigenen Code zu schreiben wo er wirklich Applikationsspezifisch ist. Dies war auf der Serverseite nur notwenbdig um die einzelnen Komponenten wie Lucene und HSQLDB zu verbinden. Ein weiterer Teil bestand darin, ein entsprechendes Servlet und einen Controller zu implementieren, welcher die Anfragen an die darunterliegende Technik weiterleitet.

Auf der Clientseite gab es zumindest mit dem Firefox-Browser ein klares Heimspiel, TeamFound ist hier nur eine von warscheinlich tausenden Erweiterungen, welche die offene Architektur des Browsers möglich macht. Aber auch der erklärte Feind der Firefox-Entwicklung, der Internet Explorer, lässt sich Erweitern, so war es unserem Team möglich auch für diesen Browser eine Client-Integration zu erstellen. Dies dürfte unter gleichartigen Projekten wohl nicht besonders häufig vorkommen und stellt somit einen Beleg dar, das auch Entwickler proprietärer Systeme das Phänomen der freien Software immer mehr beachten und eine Lehrveranstaltung an einer Universität eine

gute Möglichkeit bietet dieses Arbeitsprinzip genauer kennenzulernen.

Das TeamFound-Projekt an sich, hat in den ersten Monaten wenig mit einem 'verteilten'm Entwicklungsmuster zu kämpfen, annähernd die komplette Planungsphase könnte mit wöchentlichen Treffen effektiv gestaltet werden, erst gegen Ende des Semesters, als der 'Implementierungsdruck' zu steigen begann, wurde stärker auf klassische Techniken wie die Projekt-Mailingliste oder den Internet-Relay-Chat (IRC) zurückgegriffen. Dies hat der Produktivität allerdings nur wenig geschadet, was auch bedeutet das die bestehende Infrastruktur die von Anbietern wie Berlios oder Sourceforge bereitgestellt wird, viele technische Probleme von OpenSource-Projekten lösen kann. So kann zumindest ein kleines Projekt wie dieses sofort auf Infrastruktur wie Quellcodeverwaltung, Mailinglisten und viele andere Dinge zurückgreifen. Das TeamFound-Projekt versteht sich selber als einen Beitrag um auch die Planungs- und Recherchephasen von OS-Projekten technisch zu unterstützen oder wenigstens einen Beitrag zur Diskussion um technische Verfahren für derartige Probleme zu leisten.

Von Seiten der Teammitglieder wurde im Rahmen des Abschlusskolloquiums bereits das Interesse an einer Fortsetzung des Projektes, auch im universitären Rahmen, erklärt. Somit scheint es durchaus möglich, das es einen dritten TeamFound-Milestone geben wird und das Projekt auch längere Zeit ohne universitäre Unterstützung am Leben bleibt.

Das TeamFound-Projekt möchte sich zum Abschluss dieser Ausarbeitung vorallem bei Steffen Evers für die Organisation dieser Lehrveranstaltung, sowie dem Berlios-Projekt für die technische Unterstützung, allen Entwicklern der von uns verwendeten freien Bibliotheken und natürlich auch allen anderen freiwilligen Mitarbeiten an freier Software bedanken.

Anhang A

Interface

A.1 Interface Milestone 2

A.1.1 Allgemein

Tests

Zum Testen der Serverseitigen Interface-Implementation existiert folgende Webseite, die entsprechende Requests erstellen kann:
`http://teamfound.berlios.de/test_the_server_interface_m2.html`

Zum Testen der Clientseitigen Interface-Implementation existiert folgendes Perl-Skript, welches den Parameter command auswertet und eine enstsprechende statische XML-Antwort zurueckgibt:
`http://teamfound.dyndns.org/interface-m2/testmilestone2.pl`
Demo:
`http://teamfound.dyndns.org/interface-m2/testmilestone2.pl?command=getcategories`
Das Skript liegt auch im SVN unter `teamfound/interface/milestone2/testmilestone2.pl`

Alle Anfragen an den Server

Alle Anfrage-Parameter werden in Form von HTTP GET oder POST Variablen übertragen. Prinzipiell gibt es eine Unterscheidung, ob für jedes Kommando eine eigene URL verwendet wird, oder ob das Kommando in Form einer weiteren HTTP-GET Variablen mit übertragen wird. Unsere Implementation des Servers verwendet den zusätzlichen HTTP-GET Parameter `command`. In diesem Interface haben wir aber beide Möglichkeiten spezifiziert.

want=xml or html Die zu erwartende Antwort soll in xml oder html-format sein (default soll html werden, da dann einfache Link-Clients

möglich sind)

version=2 Milestone-Version des Interfaces

command=search Das Kommando das ausgeführt werden soll. Dieses Argument kann wegfallen wenn für jedes Kommando eine eigene, gleichnamige Anfrage-URL existiert.

HTML-Antwort

Die Clients geben an ob Sie eine HTML oder eine XML Antwort erwarten.

Soll eine vollständige HTML-Seite zurückgeben, die direkt im Browser angezeigt werden kann. Die Return-Values wie bei einer XML-Antwort müssen in diesem Fall nicht zurückgegeben werden, sondern der Text sollte gleich eine entsprechende Meldung beinhalten.

XML-Antwort

<**response**> Umschliesst alle anderen xml-tags und gibt die XSD-Datei zum verifizieren der XML-Daten an

<**interface-version**> Gibt immer die Interface-Version an, in der die Antwort des Servers formuliert wurde

<**server**> Gibt Name und Versionsnummer des Servers an. Der Server der unter der URL https://developer.berlios.de/projects/teamfound entwickelt wird gibt den Namen TeamFoundßurück. Clones dürfen Ihren eigenen Namen natürlich frei wählen.

<**addpage**> <**search**> <**addcategory**> <**getcategories**> Diese Tags beinhalten die Antworten auf die gleichnamigen Anfragen an den Server. Pro Anfrage darf nur eines dieser Tags vorkommen.

<**return-value**> <**return-description**> Der Return-Value bzw. Fehler-Code falls etwas schiefging. Die Description wird nicht genauer spezifiziert, sollte aber semantisch mit dem aufgetretenen Fehler übereinstimmen.

<**project-counter**> Bei jeder Änderung des Kategorien-Baumes(soll der Server einen internen Zähler um eins inkrementieren. Der aktuelle Wert soll bei jeder XML-Antwort mit übertragen werden. (Dann weiss die Toolbar wann sie selber die Kategorien neu vom Server abfragen muss.)Jedes Projekt fuehrt einen Counter, somit muss der Client nur die fuer ihn intressanten Bäume beachten.

```
<response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="teamfound-interface-milestone2.xsd">

 <interface-version>2</interface-version>

 <return-value>0</return-value>
 <return-description>OK</return-description>

 <project-counter>
   <project>
     <projectID>0</projectID>
     <count>3</count>
   </project>
   <project>
     <projectID>3</projectID>
     <count>5</count>
   </project>
 </project-counter>

 <category-counter>54</category-counter>

 <server>
  <name>TeamFound</name>
  <version>0.2</version>
 </server>

 <addpage>
 </addpage>

 <search>
 </search>

 <addcategory>
 </addcategory>

 <getcategories>
 </getcategories>

</response>
```

Return Codes Die Return-Codes stehen immer in den Tags <return-value>xx</return-value>. Die Description ist optional.

0 Alles ok

1 Fehler, konnte URL nicht finden

2 Ungültige Anfrage (Pflicht-Parameter fehlen oder haben die Länge null)

3 Inkompatible Interface-Version (die Anfrage hat eine Interface-Version benutzt die der Server nicht unterstützt)

4 Kategorie existiert schon (beim hinzufuegen einer neuen Kategorie)

5 Kategorie nicht gefunden (beim suchen nach einer bestimmten Kategorie)

-1 Anderer Fehler

Die Return-Descriptions sind frei wählbar, sollten aber dem Return-Code semantisch entsprechen ;-)

A.1.2 Seite hinzufügen

Anfrage an Server

command=addpage Der Kommando-Name

addpage.pl Der (default) Skriptname der die Anfrage entgegennimmt (dann fällt das Argument command weg).

category=453 Die Kategorien-Nummer zu der der Link hinzugefuegt werden soll, in dieser Version nur genau eine oder keine. Wird keine Kategorie angegeben, so soll die Top-Kategorie genommen werden (also nicht gefunden werden wenn bei der Suche eine genauere Kategorie angegeben wird).

url=http://yy.org/blabla.html Die zu indizierende url, das Protokoll ist mit anzugeben

XML Antwort

xml oder html um über Erfolg oder Misserfolg zu berichten
Verwendet in der Antwort das XML-Tag: <addpage>

```
<response>

 <interface-version>2</interface-version>

 <return-value>0</return-value>
 <return-description>OK</return-description>

 <server>
  <name>TeamFound</name>
  <version>0.2</version>
 </server>

 <addpage>
  <url>http://blabla.html</url>
 </addpage>

</response>
```

HTML Antwort

siehe Allgemeines zu HTML-Antworten

A.1.3 Suchen

Anfrage an Server

command=search Der Kommando-Name

search.pl Der (default) Skriptname der die Anfrage entgegennimmt (dann fällt das Argument command weg).

keyword=zzz Eine durch ' ' (space) getrennte Liste der Suchwörter die alle vorkommen sollen (URL-Codiert mit "%20" getrennt, also eine Suche nach auto und tv wird als keyword=auto%20tv codiert). Dies beinhaltet eine serverseitig impliziete UND-Verknüpfung aller Suchworte. Fehlt der Parameter so soll ein Fehler zurückgegeben werden. Die genaue Definition einer umfassenderen Suchanfragen-Sprache (wie OR, NOT

etc.) ist noch festzulegen, wird aber auf spaetere Interface-Milestones verschoben.

category=5&category=9&category=15 Eine Liste aller Kategorien-IDs, jeweils immer wieder neu mit category= durch die die Suche gemacht werden soll. Fehlt der Paramater, so soll alles durchsucht werden. (Grund: HTML-Formulare machen dies bei Listen und ComboBoxen mit Mehrfachauswahl genau so).

XML Antwort

Verwendet in der Antwort das XML-Tag: <search>

```
<response>

 <interface-version>2</interface-version>

 <return-value>0</return-value>
 <return-description>OK</return-description>

 <server>
  <name>TeamFound</name>
  <version>0.2</version>
 </server>

 <search>

  <keywords>
   <word>xxx</word>
   <word>yyy</word>
  </keywords>

  <result>

   <count>30</count>
   <offset>0</offset>

   <found>
    <url>http://xxx.html</url>
    <title>Der Titel der Seite</title>
    <incategory>5</incategory>
   </found>
```

```
   <found>
    <url>http://xxx.html</url>
    <title>Der Titel der Seite</title>
    <incategory>5</incategory>
   </found>

   <found>
    <url>http://xxx.html</url>
    <title>Der Titel der Seite</title>
    <incategory>5</incategory>
    <incategory>3</incategory>
    <incategory>7</incategory>
   </found>

  </result>
 </search>

</response>
```

- Die Keywords sind zur Fehlerbehandlung alle nochmals mit anzugeben.
- Jeder gefundene Link wird in einem eingenen <found> Block augegeben.
- Wurde kein Suchergebnis gefunden, so ist in <count> die Anzahl 0 einzutragen, der Return-Code aber immernoch 0 (OK) wenn sonst alles ok war

HTML Antwort

HTML-Seite mit URLs gefundener Übereinstimmungen

A.1.4 Kategorien von Server abfragen

Anfrage an Server

command=getcategories Der Kommando-Name

getcategories.pl Der (default) Skriptname der die Anfrage entgegennimmt (dann fällt das Argument command weg).

projectID=5 Nummer des projekts für diese Anfrage

XML Antwort

Verwendet in der Antwort das XML-Tag: <getcategories>

```
<response>

 <interface-version>2</interface-version>

 <return-value>0</return-value>
 <return-description>OK</return-description>

 <server>
  <name>TeamFound</name>
  <version>0.2</version>
 </server>

 <getcategories>
  <category>
   <name>name der kategorie</name>
   <description>laengere beschreibung</description>
   <id>0</id>
   <subcategories>
    <category>
     ...
    </category>
    <category>
     ...
    </category>
   </subcategories>
  </category>
 </getcategories>

</response>
```

Hinweis: die Top-Kategorie hat immer die ID 0 (null)

HTML Antwort

siehe Allgemeines zu HTML-Antworten

A.1.5 Kategorie hinzufügen

Anfrage an Server

command=addcategory Der Kommando-Name

addcategory.pl Der (default) Skriptname der die Anfrage entgegennimmt (dann fällt das Argument command weg).

name=kategoriename Den Namen den die neue Kategorie bekommen soll

subcategoryof=45 Die Kategorie der die neue Kategorie untergeordnet werden soll. Soll eine neue 1st-Level Kategorie erstellt werden ist hier 0 (null) anzugeben.

description Eine etwas längere Beschreibung der Kategorie (max. 255 Zeichen)

XML Antwort

Verwendet in der Antwort das XML-Tag: <addcategory>

```
<response>

 <interface-version>2</interface-version>

 <return-value>0</return-value>
 <return-description>OK</return-description>

 <server>
  <name>TeamFound</name>
  <version>0.2</version>
 </server>

 <addcategory>
  <name>kategoriename</name>
  <gotid>53</gotid>
 </addcategory>

</response>
```

- der neue Name
- die ID die die neue Kategorie bekommen hat

- name und gotid fallen weg wenn die kategorie schon existiert (return-value 4 - Kategorie existiert schon) oder die subcategoryof-id nicht existiert (return-value 5 - Kategorie nicht gefunden).

HTML Antwort

siehe Allgemeine HTML-Antwort

A.1.6 Alle Projekte auslesen

Anfrage an Server

command=getprojects Der Kommando-Name

addcategory.pl Der (default) Skriptname der die Anfrage entgegennimmt (dann fällt das Argument command weg).

XML Antwort

Verwendet in der Antwort das XML-Tag: <projects>

```
<response>

 <interface-version>2</interface-version>

 <return-value>0</return-value>
 <return-description>OK</return-description>

 <server>
  <name>TeamFound</name>
  <version>0.2</version>
 </server>

 <projects>
   <project>
     <name>prjectname</name>
     <description>beschreibung</description>
     <id>7</id>
   </project>
   ...
 </project>

</response>
```

HTML Anwort

siehe Allgemeine HTML-Antwort

A.1.7 Löschen einer Kategorie

Wird erst in Interface Milestone 3 implementiert werden. Es stellt sich die Frage was mit den in dieser Kategorie bereits befindlichen Links gemacht wird .. verschieben?

Anhang B

Logbuch

B.1 Projekt Logbuch

Dies ist ein Snapshot des Projekt-Logbuchs vom 17. April 2006. Die jeweils aktuelle Version kann unter `http://wiki.jonasheese.de/index.php/Logbuch` eingesehen werden.

B.1.1 2006

April 2006

17. April 2006

- Firefox-Extension v0.8 released Kechel 20:07, 17. Apr 2006 (CEST)

11. April 2006

- Vortrag gehalten Kechel 21:46, 11. Apr 2006 (CEST)

10. April 2006

- Vortrag-Slides fertig Kechel 20:14, 10. Apr 2006 (CEST)
- FF-Extension 0.8 beta fertig incl. default-Kategorie (reicht fuer Praesentation ;-) Kechel 20:14, 10. Apr 2006 (CEST)

9. April 2006

- Vortrag-Slides angefangen Kechel 20:14, 10. Apr 2006 (CEST)
- FF-Extension 0.8 funktioniert schon ganz gut Kechel 20:14, 10. Apr 2006 (CEST)

8. April 2006

- Ausarbeitung praktisch fertig incl. Anhang, Lizenz und Index Kechel 20:14, 10. Apr 2006 (CEST)

7. April 2006

- Ausarbeitung Anhang Kechel 20:14, 10. Apr 2006 (CEST)
- ff-toolbar kategorien von echtem server geht Kechel 20:14, 10. Apr 2006 (CEST)

4. April 2006

- Ausarbeitung weitergeschrieben, Interface und Protokoll Kechel 20:14, 10. Apr 2006 (CEST)
- FF-Extension 0.8 weiterprogrammiert Kechel 20:14, 10. Apr 2006 (CEST)

2. April 2006

- Ausarbeitung angefangen Kechel 20:14, 10. Apr 2006 (CEST)

März 2006

31. Maerz 2006

- FF Client kann kategorien, kategorien durchsuchen und auch seiten zu kategorien hinzufuegen, die Anzeige von Suchergebnissen laesst noch sehr zu wuenschen uebrig Kechel 20:14, 10. Apr 2006 (CEST)

12. Maerz 2006

- XML-Kategorien-Baum wird in FF-Extension korrekt angezeigt. Kechel 16:07, 12. Mär 2006 (CET)

Januar 2006

22. Januar 2006

- Diese Woche den DBLayer vorlaeufig beendet. Moddin 14:27, 22. Jan 2006 (CET)
- DBerlaeuterung im Wiki. Moddin 14:27, 22. Jan 2006 (CET)
- Vortrag im Wiki vorbereitet, Themenauflistung, konkrete Vorschlaege. Kechel 21:51, 22. Jan 2006 (CET)

12. Januar 2006

- FF-Toolbar erstellt ersten M2 request (getcategories), werted die XML-Antwort aus und zeigt die erste Kategorie als Menuepunkt an. Kechel 17:05, 12. Jan 2006 (CET)
- Testscript fuer Clients Milestone 2 geschrieben (http://wiki.jonasheese.de/index.php/Interfa Kechel 13:55, 12. Jan 2006 (CET)

B.1.2 2005

Dezember 2005

7. Dezember 2005

- FF-Toolbar 0.7 released (bug-fix). Kechel 16:08, 7. Dez 2005 (CET)
- Test-Seite fuer Server Interface Milestone 2 Implementation erstellt (http://teamfound.berlios.de/test_the_server_interface_m2.html). Kechel 20:09, 7. Dez 2005 (CET)

6. Dezember 2005

- Label mit aktueller URL aus FF-Toolbar entfernt. Kechel 18:48, 6. Dez 2005 (CET)

5. Dezember 2005

- Code Review von Martins Zeug, morgen müssen wir auf jedenfall die Architektur genauer festelegen, das wir kein Chaos produzieren. –Jonas 21:18, 5. Dez 2005 (CET)

2. Dezember 2005

- Toolbar jetzt auch ueber http://addons.mozilla.org verfuegbar. Kechel 19:22, 2. Dez 2005 (CET)

```
TeamFound 0.6 - Approval Granted
Your item, TeamFound 0.6, has been reviewed by a
Mozilla Update editor who took the following action:
Approval Granted

Please Note: It may take up to 30 minutes for your
```

```
extension to be available for download.

Your item was tested by Chris Blore using Firefox 1.5 on Windows XP.
Editor's Comments:
Thanks for submitting
```

1. Dezember 2005

- Firefox-Toolbar ist jetzt auch eine Flock-Toolbar (also compatibel zum flock-browser http://www.flock.com). Kechel 19:13, 1. Dez 2005 (CET)
- Firefox-Toolbar ist jetzt auch eine Flock-Toolbar (also compatibel zum flock-browser http://www.flock.com). Kechel 19:13, 1. Dez 2005 (CET)

November 2005

30. November 2005

- Milestone 1 Version als Servlet gebastelt.Moddin 19:57, 30. Nov 2005 (CET)
- Firefox toolbar 0.6 released. Kechel 23:54, 30. Nov 2005 (CET)

29. November 2005

- Gemeinsam Interface Milestone 2 verabschiedet. Kechel 15:23, 30. Nov 2005 (CET)

28. November 2005

- Firefox-Toolbar 0.5 released, google links funktionieren endlich (auch die weitere-ergebnisse-links). In CustomizeGoogle Malingliste gefragt wie wir unsere extensions zur Kooperation bringen koennten. Kechel 16:15, 28. Nov 2005 (CET)

27. November 2005

- Toolbar bei addons.mozilla.org hochgeladen, waiting for review and aproval. Kechel 23:16, 27. Nov 2005 (CET)
- Firefox-Toolbar v0.4 ist fertig, asyncrones laden der TeamFound und google ergebnisse, sowie eingegebene urls werden nicht gesucht sondern direkt angesprungen. Kechel 15:08, 27. Nov 2005 (CET)

26. November 2005

- Firefox-Toolbar hat nun auch einen Einstellungen-Dialog fuer die Server-Adresse. Kechel 22:49, 26. Nov 2005 (CET)

25. November 2005

- XSD-Datei zum verifizieren der XLM-Daten nach Interface Milestone 2 erzeugt. Kechel 18:12, 25. Nov 2005 (CET)

24. November 2005

- Firefox-Toolbar sucht jetzt auch selber noch bei google und zeigt die Ergebnisse von TeamFound und google nebeneinander an. Version 0.3 der Toolbar released. Kechel 22:54, 24. Nov 2005 (CET)

23. November 2005

- Am ServerProgramm ein paar Verbesserungen vorgenommen hinsichtlich der Performance
- Script und java so erweitert das adding bei gesetztem lock wartet(sehr simple variante) Moddin 21:17, 23. Nov 2005 (CET)
- Server-Release 0.1 auf Berlios geladen, kleine install.txt hinzugefuegt, <category-counter> zu Interface Milestone 2 hinzugefuegt. Kechel 16:23, 23. Nov 2005 (CET)
- Web-Client geschrieben (einfaches HTML-Formular zum suchen, und link zum adden neuer seiten). Kechel 20:21, 23. Nov 2005 (CET)

22. November 2005

- Firefox-Extension 0.2 xpi-file erstellt und als erstes file auf berlios released, installations-anleitung angepasst und link auf end-user homepage gesetzt. Kechel 19:46, 22. Nov 2005 (CET)
- Interface Milestone 2 spezifiziert (als Vorschlag, aber schon sehr weit ausgearbeitet!) Kechel 21:59, 22. Nov 2005 (CET)

21. November 2005

- An den Server Interna gearbeitet um Milestone 1. naeherzukommen (Files sind im svn, fortlaufenden Index von Seiten kann gebaut und durchsucht werden) Moddin 21:05, 21. Nov 2005 (CET)

19. November 2005

- Firefox-Toolbar ist in version 0.1 fertig und einsatzbereit. Es koennen Seiten hinzugefuegt werden und diese wieder durchsucht werden. Der Server funktioniert ebenfalls in einer ersten Version bei uns lokal auf einem Rechner. Der Index wird jedoch bei jeder neu hinzugefuegten seite komplett ueberschrieben, so dass immer nur die zuletzt hinzugefuege Seite durchsucht wird ;-) Kechel 20:13, 19. Nov 2005 (CET)

- Installations-Anleitung fuer Server-Milestone 1 geschrieben. Kechel 20:39, 19. Nov 2005 (CET)

- Installations-Anleitung und kleine Doku fuer Firefox-Toolbar Milestone 1 erstellt. Kechel 22:48, 19. Nov 2005 (CET)

- Logo auf http://teamfound.berlios.de upgeloaded und in Präsentation vorgestellt. Kechel 23:07, 19. Nov 2005 (CET)

- Icon in Firefox-Toolbar eingebaut und screenshot ins wiki getan damit endlich mal etwas optisch anspruchsvolles im wiki ist ;-) Kechel 00:41, 20. Nov 2005 (CET)

17. November 2005

- Erste Server-Perl Skripte in SVN getan und in Server entsprechende Beschreibung eingefuegt. Kechel 15:59, 17. Nov 2005 (CET)

15. November 2005

- Interface-Spezifikation für Milestone 1 fertig. Kechel 11:03, 15. Nov 2005 (CET)

- Erste Server-Perl Scripte erstellt. Jan & Moddin

12. November 2005

- kleiner Testlauf mit Lucene-Demo bei mir ... 270 MB hat ungefaehr 3 min gebraucht zum index erstellen und nahm 14 MB ein. Ausserdem haben sie auch nen HTMLParser in der Demo aber nur fuer reine HTML-Files ... alles andere wird dann nicht mit index versehen, aber immerhin fuer uns schon sehr nuetzlich. Moddin 15:04, 12. Nov 2005 (CET)

09. November 2005

- Mailingliste teamfound-development@lists.berlios.de eingerichtet und entsprechende links auf diesem wiki plaziert. Kechel 15:25, 9. Nov 2005 (CET)

- Erster SVN-Checkout sowie erste Firefox-Toolbar upgeloaded. Kechel 16:06, 9. Nov 2005 (CET)

08. November 2005

- Erste eigene Firefox-Extension erstellt die auch funktioniert (Menu-Item 'TeamFound' der beim anklicken eine MsgBox ausgibt ;-) Kechel 23:25, 8. Nov 2005 (CET)

07. November 2005

- Nach nutzbaren tools fuer den Server gesucht und alles was ich so gefunden habe ins Wiki unter Server geschrieben.Moddin 16:43, 7. Nov 2005 (CET)

- Nach Informationen zum IE Plugin gesucht und alles unter Toolbar als Extension für Internet Explorer eingetragen.Andreas 22:34, 7. Nov 2005 (CET)

- Erste Firefox-Extension versucht zu programmieren .. naja, und zum ersten mal von XUL gehoert ;-) Aber dafuer hat die HelloWorld-Extension gleich funktioniert! Kechel 23:45, 7. Nov 2005 (CET)

03. November 2005

- berlios-Services sind inzwischen aktiv, also shell login geht, die domain teamfound.berlios.de ist erreichbar und ich habe auch gleich mal einen minimalen content aufgesetzt. Kechel 14:01, 3. Nov 2005 (CET)

- Projekt bei berlios.de wurde approved. Projekt ein bisschen eingerichtet, wichtige infos auf wiki uebertragen. Kechel 11:51, 3. Nov 2005 (CET)

- Projekt teamfound auf berlios.de beantragt. Kechel 00:27, 3. Nov 2005 (CET)

02. November 2005

- TeamFound Wiki-Seite grundsätzlich strukturiert, Teilprojekte angelegt und überall mal einen Anfang formuliert. Kechel 23:00, 2. Nov 2005 (CET)

01. November 2005

- Nachtrag: Gemeinsames Treffen an Uni, Festlegung der Projekte, und verteilung der Projektmitglieder (so kamen wir 4 erstmals zu diesem Projekt zusammen). Kechel 23:00, 2. Nov 2005 (CET)

B.2 Firefox Toolbar Changelog

Das originale Changelog[1] (in englischer Sprache):

B.2.1 2006

Bisher noch kein neues Release in 2006. Die version 0.8 ist zwar bereits eine vollständige Imlementation nach Mileston 2, funktioniert aber noch nicht stabil genug um diese als neues Release freizugeben.

B.2.2 2005

Firefox & Flock toolbar 0.7 (07-DEC-2005)

- Bug-Fix: Add-Button actually works now
- Label with current URL removed (though we are not going to replace the location-bar)
- Compatibility with Flock browser tested and added to install.rdf

Firefox toolbar 0.6 (30-NOV-2005)

- Input-field now behaves the same like the normal Firefox locationbar (including history popup) when urls are typed. Soon this can replace the Firefox locationbar and the Firefox searchbar with only one input-field ;-)
- Settings-Dialog allows to give custom external search engines

[1] `http://teamfound.berlios.de/#firefoxchangelog0.7`

- Settings-Dialog allows to only search external, only search TeamFound or search both at once

Firefox toolbar 0.5 (28-NOV-2005)

- Google search results are actually working now, including further results
- Compatible with Firefox 1.5 release candidates

Firefox toolbar 0.4 (27-NOV-2005)

- Clicking on TF-Icon opens preferences dialog, here you can setup your TeamFound server url and choose between two search result layouts
- Add page - Adds the current page to the TeamFound-Index.
- text field - Here you can enter search words or urls. The toolbar tries to automatically figure out what you meant and though will either search or just visit the given url.
- current url - The current url is always displayed at the end of the toolbar, though not overwriting your current search words.

Firefox toolbar 0.3 (24-NOV-2005)

- adds search at extern search-engine
- shows google and teamfound search-results in same window

Firefox toolbar 0.2 (22-NOV-2005)

- first release at berlios.de
- available as .xpi

Firefox toolbar 0.1 (19-NOV-2005)

- initial release

Anhang C

TeamFound Programm Lizenz

TeamFound sowie alle enthaltenen Komponenten wie Source-Code, Binaries und Bilder stehen under der GNU General Public License (GPL) Version 2 oder höher. Eine Kopie der Lizenz liegt allen Source- und Binär Versionen bei.

Anhang D

GNU Free Documentation License

Version 1.2, November 2002

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image

format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or

any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Literaturverzeichnis

[Apa02] *Apache Ant 1.5.1 Manual*. Apache Software Foundation, http://ant.apache.org/manual/, 2002.

[Dat00] C. J. Date. *An Introduction to Database Systems*. Addison Wesley, 2000.

[ExFF] *Extend Firefox*. Mozilla, http://developer.mozilla.org/mozilla-org/contests/extendfirefox/documentation.php

[FFTT] *Firefox Toolbar Tutorial*. Born Geek, http://www.borngeek.com/firefox/toolbar-tutorial/

[Fri97] Jeffrey E. F. Friedl. *Mastering Regular Expressions*. O'Reilly, 1997.

[Gswed] *Getting started with extension development*. mozillaZine, http://kb.mozillazine.org/Getting_started_with_extension_development.

[Goo00] M. Goossens, F. Mittelbach und A. Samarin. *Der LaTeX Begleiter*. Addison Wesley, 2000.

[LucQS] *Apache Lucene Query Syntax*. Apache Software Foundation http://lucene.apache.org/java/docs/queryparsersyntax.html .

[Oua02] S. Oualline. *Vi IMproved – Vim*. New Riders, 2002 .

[PolPos] *PolePosition*. the open source database benchmark http://www.polepos.org.

[Roac04] Eric. *How to create Firefox extensions*. roachfiend.com, http://roachfiend.com/archives/2004/12/08/how-to-create-firefox-extensions

[Spo01] J. Spolsky. *User Interface Desing for Programmers*. Apress, 2001 .

[XULPL] *XUL Planet* XUL Planet, http://xulplanet.com .

Schlagwortverzeichnis