Auswertung und verteilte Darstellung von Statistikdaten

Wolfgang Kandlbauer

Analysis and Distributed Representation of Statistical Data

Master's Thesis

at

Graz University of Technology

submitted by

Wolfgang Kandlbauer

Knowledge Management Institute (KMI),
Graz University of Technology
A-8010 Graz, Austria

17th March 2011

Advisor: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic

Auswertung und verteilte Darstellung von Statistikdaten

Masterarbeit

an der

Technischen Universität Graz

vorgelegt von

Wolfgang Kandlbauer

Institut für Wissensmanagement (IWM),
Technische Universität Graz
A-8010 Graz

17. März 2011

Diese Arbeit ist in deutscher Sprache verfasst.

Begutachter: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic

Abstract

The Web allows for information to be linked together and to be made accessible to everyone. The demands on this information system have increased considerably in recent years. To facilitate users work on the net, high value has been placed on the usability of web applications which allows users to add and modify content easily. As a result a huge amount of information and data has been created.
A major part of this thesis provides users with a tool to generate graphical representations of data. These can then be viewed and analyzed by users.
The goal is to create a generally working framework, which allows for adding statistics in an easy way. To reduce the amount of work for developers when adding new features, extensibility and maintenance are key factors. To meet these requirements, the framework is written in Java and interacts with gnuplot to generate the graphical representations. During the design phase of the application, extensibility was a very important aspect.
The user interface is a web application which allows the user to upload the data to generate statistics. Admin users also have access to the structure of the filesystem where they can create any folder structure(create folders and subfolders). All of the generated graphs can be viewed, compared and analyzed in a separate window.
An important aspect of this user interface is the usability. Depending on the input, users only see the usable components. For instance, the ”create subfolder”-button is only available, if users specify the folder in which they want to create the subfolder.

Kurzfassung

Das Web ermöglicht es, Informationen jedem anderen zugänglich zu machen und miteinander zu verlinken. Die Nachfrage dieses Informationssystems stieg in den letzten Jahren sehr stark an. Um dem Benutzer das Arbeiten im Netz zu erleichtern, wurde in den letzten Jahren viel Wert auf die Usability der Webapplikationen gelegt. Diese sollen es dem Anwender ermöglichen, auf einfache Art und Weise Inhalte zu gestalten und zu bearbeiten. Dadurch entstand eine riesige Menge an Informationen und Daten.
Ein Hauptbestandteil dieser Arbeit ist es, dem Benutzer ein Tool zur Verfügung zu stellen, um aus Daten grafische Darstellungen zu generieren. Diese können anschließend vom Benutzer betrachtet und analysiert werden. Ziel ist es, einen Rahmen dieses Tools zu schaffen, der möglichst allgemein gehalten ist, um ein nachträgliches Hinzufügen von Statistiken möglichst einfach zu ermöglichen.
Um diesen Anforderungen gerecht zu werden, wurde die Möglichkeit geschaffen, gnuplot Grafiken von Java aus zu generieren. Dabei wurde sehr viel Wert auf die Erweiterbarkeit gelegt.
Als Benutzerschnittstelle dient eine Webapplikation, die es dem Benutzer ermöglichen soll, die Daten hochzuladen, aus denen eine Statistik generiert werden soll. Der Admin Benutzer hat außerdem Zugriff auf die Ordnerstruktur, wo er aus Gründen der Übersichtlichkeit eine beliebige Ordnerstruktur anlegen kann. Alle generierten grafischen Darstellungen können in einem eigenen Fenster betrachtet, verglichen und analysiert werden.
Ein wichtiger Aspekt für die Oberfläche war die Usability. Dem Benutzer werden immer nur die Komponenten angezeigt, die er aufgrund der getroffenen Auswahlen auch verwenden darf.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

Die hier vorgestellte Masterarbeit ist am Institut für Wissensmanagement (IWM) entstanden. An dieser Stelle möchte mich herzlich bei Herrn Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic bedanken, der mir ermöglichte, diese Arbeit zu verfassen und mich über die gesamte Dauer umfassend betreut und mit Ratschlägen unterstützt hat. Vielen herzlichen Dank!

Mein Dank gilt auch all meinen Studienkollegen, die mich während meines Studiums begleiteten. Vor allem Christoph Portsch gilt hier mein Dank, der mir immer zur Seite stand und mich moralisch unterstützte. Vielen Dank!

Ein herzliches Dankeschön gilt meiner Familie und ganz besonders meinen Eltern, Friedrich und Gertrud, die mir die Ausbildung ermöglicht haben und auf deren Unterstützung ich immer zählen konnte. Sie überließen mir stets die Freiheit, meine Entscheidungen selbst zu treffen und meine Erfahrungen selbst zu machen. Auch bei meinen Geschwistern Jürgen und Sandra - und deren Lebenspartnern Margret und Ewald-, die mir stets mit Rat und Tat zur Seite standen und mich moralisch unterstützten, möchte ich mich bedanken. Ich danke euch allen von Herzen!

Ein besonderer Dank gilt einem ganz besonders wichtigen Menschen in meinem Leben - meiner lieben Freundin Claudia. Ohne ihre moralische Unterstützung, ihren aufmunternden Worten in schwierigen Zeiten und vor allem ihrer Liebe wäre diese Arbeit nicht so schnell entstanden. Ich danke dir von Herzen!

Kapitel 1

Einleitung

1.1 Motivation

> *„Suppose all the information stored on computers everywhere were linked, I thought. Suppose I could program my computer to create a space in which anything could be linked to anything.“*
>
> [Tim Berners-Lee]

Bei der Idee zur Entwicklung des World Wide Web hatte Tim Berners-Lee ein klares Ziel vor Augen. Er wollte einen Raum schaffen, in dem es jeden Benutzer auf der ganzen Welt möglich ist, eigene Inhalte mit anderen zu teilen und die Inhalte anderer anzusehen und zu bearbeiten. Außerdem sollte es möglich sein, Informationen beliebig miteinander zu verlinken [Berners-Lee und Fischetti, 2000, Seite 37f].

Er hatte damals keine Ahnung, welch große Ausmaße die Entwicklung seines Informationssystems annehmen würde. Heute ist nahezu alles im World Wide Web zu finden. Die Leute kaufen übers Netz ein, recherchieren Informationen, teilen ihre Aktivitäten mit anderen, u.v.m.

Das Problem dabei ist die riesige Anhäufung an Daten die generiert wird. Laut [Keim, 2002, Seite 1] wurden im Jahr 2002 ca. 1 Exabyte (=1 Million Terrabyte) an Daten generiert. Aufgrund der großen Anzahl und der immer komplexer werdenden Daten wird eine statistische Auswertung für Wissenschaftler immer schwieriger.

1.2 Zielsetzung

Ziel der Arbeit ist es, ein Tool zu entwickeln, um die zuvor genannten Probleme der Datenvielfalt für den Benutzer umgänglicher zu machen. Eine Webapplikation soll den Anwendern (Wissenschaftler, etc.) ermöglichen, individuell Statistiken aufgrund der vorhandenen Input-Daten zu generieren und diese am Server zu speichern.

Ein wichtiger Askept dabei ist die verteilte Anwendung der Applikation. Jeder Benutzer soll dadurch die Möglichkeit haben, die Statistiken zu betrachten und so bei der Auswertung bzw. bei der Generierung von Wissen mithelfen zu können.

1.3 Struktur der Arbeit

In dieser Masterarbeit werden theoretische Hintergründe des World Wide Web, der Visualisierung von Daten und das Thema Statistiken behandelt. Unter deren Berücksichtigung wird eine Java Webapplikation entwickelt, die es ermöglicht, via gnuplot Statistiken zu generieren und diese mit anderen Benutzern

zu teilen. Die Themen gnuplot und Webapplikationen werden ebenfalls auf theoretischer Ebene betrachtet.

Das Kapitel 2 dieser Masterarbeit beschäftigt sich mit dem Thema „Verteilte Darstellung von Statistikdaten“. Da dieser Begriff relativ breit gefächert ist, wird das Kapitel in die Themenbereiche „World Wide Web“, „Statistiken“ und „Visualisierung von Daten“ unterteilt. Der Inhalt reicht von der Erfindung des Web mit deren Zielen und Auswirkungen, über Arten von Statistiken bis hin zu Methoden, um große Datenmengen anschaulich zu visualisieren.

In Kapitel 3 wird ein Überblick über die Anforderungen der Applikation, sowohl von der gnuplot, als auch von der Webapplikations-Seite gegeben.

Kapitel 4 befasst sich mit dem Thema „gnuplot“. Unterteilt wird dieses in die Unterpunkte „Definition“ und „Funktionalität“ von gnuplot. Einerseits soll in diesem Kapitel definiert werden, was gnuplot überhaupt ist und wozu es eingesetzt werden kann. Im Bereich „Funktionalität“ wird betrachtet, wie gnuplot funktioniert, d.h. Einstellungsmöglichkeiten, Generierung von Plots, usw. Dieser Bereich dient als eine Art Tutorial für den Leser und sollte ihm das Grundverständnis und die Möglichkeiten von gnuplot vermitteln, um dessen Einsatz in der Applikation besser verstehen zu können.

In Kapitel 5 wird das Thema „Webanwendungen“ behandelt. Der Inhalt unterteilt sich in eine Übersicht der vorhandenen Web Application Frameworks (WAF) zur Erstellung einer Java Webapplikation und in die detaillierte Betrachtung der verwendeten Technologien. Diese hat die Definitionen, Funktionsweisen und Anforderungen, um damit arbeiten zu können sowie kurze Code-Beispiele zum Inhalt.

Kapitel 6 hat das Thema „Umsetzung der Applikation“ zum Inhalt. Darin wird die Java Architektur, die es möglich macht, gnuplot von Java aus zu starten und einen Plot mit den gesetzten Einstellungen zu generieren, ausführlich beschrieben. Ziel dabei war es, diese Architektur möglichst allgemein zu halten, um Erweiterungen der Applikation zu erleichtern.
Abschließend gibt das Kapitel detaillierte Informationen zur Umsetzung der Webapplikation und beschäftigt sich u.a. mit deren Grundeinstellungen. Ein weiterer wichtiger Punkt ist die Verwendung einzelner Komponenten, um den Anforderungen gerecht zu werden. Einige Screenshots zeigen die Benutzeroberfläche. Besonders wichtige Funktionen, wie etwa FileUpload werden im Detail betrachtet.

In Kapitel 7 werden die Ideen und Konzepte der Arbeit erneut betrachtet und ein Ausblick darauf gegeben, wo der Hebel angesetzt werden kann, um das Projekt möglicherweise noch zu erweitern bzw. auch zu verbessern.

Kapitel 2

Verteilte Darstellung von Statistikdaten

2.1 World Wide Web

> *„Suppose all the information stored on computers everywhere were linked, I thought. Suppose I could program my computer to create a space in which anything could be linked to anything.“*
>
> [Tim Berners-Lee]

2.1.1 Allgemeine Informationen und Definitionen

Laut Anforderungen (siehe Abschnitt 3) soll im Zuge des Projekts eine Webapplikation erstellt werden. Aus diesem Grund möchte ich in diesem Abschnitt beschreiben, wie das Web überhaupt entstanden ist und welche Schritte in der Entwicklung durchlaufen wurden. Gezeigt werden die ersten „Gehversuche“ bis hin zum World Wide Web wie wir es heute kennen. Zu Beginn müssen die Begriffe „Internet“ und „World Wide Web“ definiert werden um zu zeigen, dass diese beiden keinesfalls ident verwendet werden können. Ich muss zugeben, dass auch ich den Unterschied zuvor nicht kannte und sicherlich im Alltag die Begriffe falsch verwendet habe und ich bin mir sicher, dass es vielen anderen Leuten auch so gegangen ist. Um klarzustellen wovon die folgenden Bereiche handeln, sind diese Definitionen unerlässlich.

Internet: Das Internet ist eine Netzwerk-Infrastruktur, die Millionen Computer weltweit miteinander verbindet. Jeder Computer kann mit jedem kommunizieren, sofern beide mit dem Internet verbunden sind [Beal, 2011].

World Wide Web (Web): Ist eine Möglichkeit, Informationen über das Internet auszutauschen. HTTP (HyperText Transfer Protocol) wird verwendet, um Daten zu transportieren. Das Web unterstützt Browser, um Internetseiten anzusehen und den Hyperlinks zu folgen [Beal, 2011].

Um die Geschichte des Web möglichst detailgetreu beschreiben zu können, war zuerst etwas Recherche notwendig. Im Zuge dessen bin ich auf einigen Webseiten auf den Namen Tim Berners-Lee gestoßen, der als Erfinder des World Wide Web gilt. Ein wenig später habe ich das Buch *Weaving the Web* von Tim Berners-Lee gefunden, welches die schrittweise Entwicklung des Web aus seiner Sicht beschreibt.

Dieses Buch war sehr interessant zu lesen, da im Zuge der Entwicklung sehr oft eine geniale Idee entstand, aber diese kaum Unterstützung bei Anderen fand. Es traten immer wieder Probleme auf. Aus technologischer wie aus menschlicher Sicht war es sehr interessant zu lesen. Irgendwie muss der Leser im Entwicklungsprozess doch etwas mitleiden . Ich möchte in diesem Abschnitt einige wichtige Prozesse seiner Entwicklung näher betrachten.

Da Tim Berners-Lee als der Erfinder des World Wide Web gilt und er auch heute noch viel Einfluss bei der Steuerung des WWW hat, möchte ich zuerst ein paar Informationen zu seiner Person auflisten

[Berners-Lee]:

- Name: Sir Timothy Berners-Lee
- Geboren: 08. Juni 1955
- Familie: 2 Kinder, getrennt von seiner Frau
- Studium: Physik an der Oxford Universität (England)
- World Wide Web: Er hatte die Idee zur Vernetzung von Informationen. Die Schritte der Entwicklung werden in Abschnitt 2.1.2 detailliert beschrieben.
- Auszeichnungen: All diese zu erwähnen würde zu lange dauern, deshalb sei hier nur eine Auswahl aufgelistet [1]
  - 1995: Erfinder des Jahres; Kilby Stiftung („Young Innovator oft he Year“)
  - 1997: Britischer Verdienstorden („Order of the British Empire“)
  - 2004: Tim Berners-Lee wird zum Ritter geschlagen
  - usw.

2.1.2 Entwicklung des World Wide Web

1980 nahm Tim Berners-Lee einen Software Consulting Job bei CERN in Genf an. Das CERN war stets sehr beliebt bei Physikern und so fanden sich Wissenschaftler aus der ganzen Welt ein, um dort ihre Arbeiten durchzuführen [Berners-Lee und Fischetti, 2000, Seite 4ff].

Tim Berners-Lee hatte schon lange die Vision, Informationen über Computer auszutauschen und diese zu verlinken - alles soll mit allem verbunden werden können. Der Traum eines World Wide Web war aber noch weit entfernt. Der erste Schritt in diese Richtung war „Enquire“ [Berners-Lee und Fischetti, 2000, Seite 1].

2.1.2.1 Enquire

War ein Programm zur Beobachtung, wer welches Program schrieb, welches Programm auf welchem Rechner lief und welcher Mitarbeiter in welchem Projekt arbeitete. Ziel war es also, Beziehungen zwischen Menschen und technischen Einrichtungen herzustellen. Jede Information stand auf einer Seite und konnte als Knoten gesehen werden. Um neue Knoten hinzufügen zu können, musste ein Link von einem bereits existierenden Knoten gesetzt werden. Berners-Lee konnte seine Arbeit hier nicht vertiefen, weil sein Vertrag bei CERN auslief [Berners-Lee und Fischetti, 2000, Seite 9f].

2.1.2.2 Tangle

John Poole, ein Arbeitskollege von Tim vor seiner Zeit bei CERN, gründete in der Zwischenzeit die Firma Image Computer Systems. Nach dem Auslauf des Vertrags wechselte Tim in diese Firma. Die Arbeit dort war aber nicht die Herausforderung die er suchte und deshalb bewarb er sich für einen frei werdenden Job bei CERN (1984) [Berners-Lee und Fischetti, 2000, Seite 11f].

Da Enquire nicht mehr auf den Rechnern vorhanden war verfolgte Berners-Lee einen alternativen Ansatz. In diesem wurden Informationen als Verbindungen dargestellt. Ein Wort besteht beispielsweise als Verbindung mehrerer Buchstaben, ein Satz aus mehreren Worten, usw. Tritt ein Wort (Knoten) öfter auf, wird nur mehr die Referenz eingesetzt und nicht mehr das Wort selbst. Das Problem dabei war das fast unmögliche Debuggen [Berners-Lee und Fischetti, 2000, Seite 12f].

[1] Anm.: Eine Übersicht aller Auszeichnungen ist unter `http://www.w3.org/People/Berners-Lee/Longer.html` zu finden.

2.1.2.3 Enquire Version 2

Im Jahr 1984 stiegen das Wachstum von CERN und damit auch die Möglichkeiten für deren Mitarbeiter. Neue Computer mit neuen Betriebssystemen und neue Programmiersprachen kamen auf den Markt. Das gab Berners-Lee die Möglichkeit, ein erstes RPC (Remote procedure call [2])-Programm zu schreiben [Berners-Lee und Fischetti, 2000, Seite 14].

Berners-Lee begann mit der Entwicklung von Enquire Version 2 für die Systeme VAX und Compaq. Es unterstützte vorerst aber nur interne Links. Ziel war es, ein möglichst generelles System ohne Einschränkungen und Regeln zu schaffen, welches komplett dezentralisiert arbeitet und somit mehr Flexibilität für den Benutzer bringen soll. Er testete das Programm zwar ständig, aber bei CERN kam es kaum zur Verwendung [Berners-Lee und Fischetti, 2000, Seite 15f].

2.1.2.4 Internet

Mit seinen Programmen war es nur möglich, ein paar Computer via Kabel zu verbinden und mit der begrenzten Anzahl an Rechnern zu arbeiten. Das entspricht nicht Berners-Lee's Vorstellungen von seinem Projekt. Zu dieser Zeit gab es aber bereits standardisierte Protokolle [Berners-Lee und Fischetti, 2000, Seite 17ff]:

$$\text{VAX} \rightarrow \text{DecNet}$$
$$\text{UNIX} \rightarrow \text{TCP/IP}$$

Er modifizierte sein RPC-Programm und fügte TCP/IP-Unterstützung hinzu; das war sein erstes Internet. Berners-Lee entschied sich also für TCP/IP. Er benötigte für sein Ziel ein Framework, um Hypertext zu verwirklichen und so Informationen zu teilen und darüber hinaus extern zu verlinken. TCP/IP ermöglichte ihm das im Prinzip [Berners-Lee und Fischetti, 2000, Seite 20].

2.1.2.5 Hypertext

Es gab bereits damals unterschiedliche Arten von dokumentierten Informationen, wie etwa Hilfsprogramme, Telefonbücher, u.v.a.m. Berners-Lee wollte aufgrund deren Gemeinsamkeiten ein konsistentes Mastersystem. Dazu hatte er drei Möglichkeiten [Berners-Lee und Fischetti, 2000, Seite 20]:

1. Ein neues Schema entwickeln, welches besser war als alle anderen
2. Ein existierendes Schema verwenden und mit dessen Einschränkungen leben lernen
3. Eventuell herausfinden, dass alle Remote Systeme etwas gemeinsam haben

Letzterer sollte schließlich sein Ziel werden. Er schrieb einen Ansatz dazu (März 1989), eine allgemeingültige Basis für die Kommunikation zwischen Geräten zu entwickeln, wobei die Individualität der einzelnen Knoten erhalten bleiben sollte. Hypertext sollte der Weg zu seinem Ziel sein [Berners-Lee und Fischetti, 2000, Seite 20].

Unglücklicherweise erhielt Berners-Lee zu seinem Ansatz keinerlei Feedback und auch ein Jahr später, als er es erneut verbreitete, meldete sich niemand [Berners-Lee und Fischetti, 2000, Seite 20f].

[2] Anm.: Informationen zu RPC sind unter `http://www.webopedia.com/TERM/R/RPC.html` verfügbar

2.1.2.6 NeXT

Berners-Lee führte intensive Gespräche mit Mike Sendall über den Kauf eines NeXT Computers [3]. Diese hatten als erste Computer eine intuitive „point-and-click“ und Ordner-Oberfläche. Nach Kauf dieses Rechners sahen auch andere die nützlichen Features und kamen dabei auf den Geschmack. Durch die Ordner wurde auch mein Thema von Hypertext zum Teilen und Verlinken von Informationen interessant. Voller Elan und mit dem OK von Ben Segall begann Tim Berners-Lee das Projekt. Diese wurde von ihm World Wide Web genannt [Berners-Lee und Fischetti, 2000, Seite 22f].

2.1.2.7 Web

> *„The Web was not a physical thing that existed in a certain place. It was a space in which information could exist.“*
>
> [Tim Berners-Lee]

Die Grundidee hinter dem Begriff *Web* war, dass irgendwann jeder und von jedem Ort aus Dokumente, Datenbanken, Grafiken, Sounds, Videos, etc. verfügbar machen und jeder Benutzer(sofern autorisiert) auf der ganzen Welt diese Informationen abfragen können sollte. Diese Informationen sollen dargestellt, verlinkt, aber auch bearbeitet werden können [Berners-Lee und Fischetti, 2000, Seite 37f].

Entwicklung

Berners-Lee begann Code für das Web zu schreiben. Gleichzeitig versuchte sein Kollege Robert Cailliau das WWW für CERN umzusetzen. CERN gab den beiden aber keinerlei Unterstützung. Bei ihnen galt der Grundsatz „Buy don’t build“ für neue Technologien. Deshalb suchten die beiden nach bereits vorhandenen Hypertext-Editoren. Sie wurden auch fündig und versuchten diese mit dem Internet zu kombinieren. Die Idee wurde aber ebenfalls prompt von den Vorgesetzten abgelehnt. Dies sei „zu kompliziert“ [Berners-Lee und Fischetti, 2000, Seite 26].

Alle Bemühungen und guten Ideen schienen umsonst. Tim Berners-Lee war aber glücklicherweise so fest von der Idee überzeugt, dass er im Oktober 1990 auf eigene Faust damit begann, das Projekt umzusetzen.

Als erstes Ziel setzte er sich die Entwicklung eines Web-Client. Dieser sollte laut [Berners-Lee und Fischetti, 2000, Seite 28] folgende Funktionalitäten unterstützen:

- Erstellen von Hypertext-Seiten
- Durchsuchen von Hypertext-Seiten
- Bearbeiten von Hypertext-Seiten

NeXT hatte bereits einige Tools mit an Bord, unter anderem NeXTStep (Textverarbeitungsprogramm), welches er verwenden konnte. Ein wichtiger Punkt dabei war die Unterscheidung von Text, der als Link dient und gewöhnlichem Text. Um die Informationen von einem Link zu speichern, benötigte Berners-Lee etwas Speicher. Zum Glück gab es in den NeXT Systemen 32 Reserve-Bits. Diese machte er sich zu seinem Nutzen [Berners-Lee und Fischetti, 2000, Seite 28].

Mit diesem Wissensstand war Hypertext „leicht“ und die Entwicklung schritt sehr schnell voran [Berners-Lee und Fischetti, 2000, Seite 28f] [4]:

[3] Anm.: NeXT INC. wurde von Steve Jobs (Gründer von Apple) gegründet

[4] Anm.: Die Begriffe HTTP, HTML, und URI werden in Abschnitt 2.1.2.7genauer definiert

- Code für das HTTP (Hypertext Transfer Protocol) wurde geschrieben
- URI (Unified Resource Identifier) wurde definiert
- Mit November hatte Berners-Lee den ersten „point-and-klick“-Browser mit dem Namen *WorldWideWeb* fertig
- HTML (Hypertext Markup Language) wurde eingeführt
- Bereits im Dezember funktionierte der Browser auch mit HTML
- Entwicklung des ersten Web-Servers (info.cern.ch)
  - Dieser existiert nach wie vor. Jetzt ist dort eine kurze Geschichte über die Entstehung des Web zu finden

Das alles hört sich sehr gut an, aber es gab ein sehr großes Problem dabei: es gab keine Möglichkeit, Hypertext-Seiten ohne NeXT-System zu betrachten und damit einer größeren Benutzerzahl zur Verfügung zu stellen.

Aus diesem Grund musste ein neuer Browser entwickelt werden. Nach der Idee von Tim Berners-Lee sollte Web für jedermann von jedem internetfähigen Gerät zugänglich sein. Der Browser soll also auf allen Geräten funktionsfähig und der Inhalt auch darauf darstellbar sein. Nach kurzem Überlegen kam er auf einen gemeinsamen Nenner. Alle Computerarten haben eine Art von Inputgerät (Keyboard) und können damit ASCII-Zeichen erzeugen. Die primitivsten Computer in Bezug auf Visualisierung waren damals Terminals. Um auch mit diesen konform zu sein, beauftragte Berners-Lee seine Kollegin Nicola Pellow mit der Entwicklung eines „Zeilen“-Browsers [Berners-Lee und Fischetti, 2000, Seite 30], welcher in Abbildung 2.1 dargestellt wird.

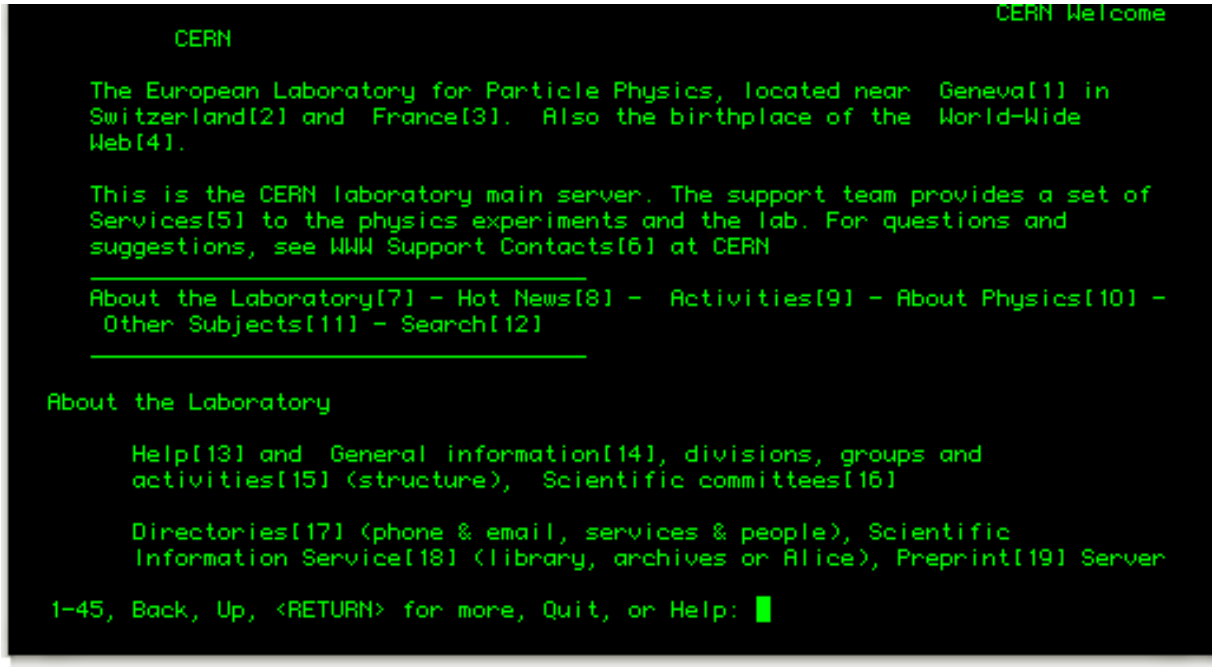

Abbildung 2.1: Der erste Browser, der auf verschiedenen Computersystemen funktionierte [CERN, 2008].

Das Produkt funktionierte soweit. Bevor es aber verkauft werden konnte, wollte Berners-Lee es noch weiter entwickeln. Um den Kontext von CERN nicht völlig aus den Augen zu verlieren und Fragen bezüglich seiner aktuellen Tätigkeiten zu umgehen (er wurde ja nicht für die Entwicklung des Web bei

CERN eingestellt), befasste er sich damit, das aktuelle Telefonbuch von CERN ins System zu integrieren. Jeder Mitarbeiter sollte von seinem Rechner aus darauf zugreifen können [Berners-Lee und Fischetti, 2000, Seite 31f].

Diese verstanden aber nicht, warum nicht die alte, lokale Applikation beibehalten werden konnte. Berners-Lee wollte die Informationen übers Netz transportieren, um sich seinem Ziel zu nähern. Er wollte basierende Hypertext Systeme, welche über CD-ROM oder Diskettenlaufwerk agierten, durch seine Idee von weltweit vernetzten Informationen ersetzen. Jeder könnte dann jederzeit Informationen eines beliebigen Autors lesen oder bearbeiten. Dazu wurden allerdings Standards in den verwendeten Protokollen und Technologien benötigt [Berners-Lee und Fischetti, 2000, Seite 31ff]. Diese wurden bereits zuvor erwähnt und werden im folgenden Abschnitt detaillierter betrachtet.

Protokolle

In der Welt vor dem Web machten unterschiedliche Netzwerke, Speicherformate, Datenformate und Zeichenkodierungen einen Austausch von Informationen zwischen unterschiedlichen Computern prinzipiell unmöglich. Um das World Wide Web also so generell wie möglich halten zu können mussten einige Standards definiert werden, die unbedingt eingehalten werden mussten [Berners-Lee und Fischetti, 2000, Seite 35]. Diese sind laut Berners-Lee [Berners-Lee und Fischetti, 2000, Seite 36ff] folgende:

- URI (Universal Resource Identifier)
- HTTP (Hypertext Transfer Protocol)
- HTML (Hypertext Markup Language)

URI (Universal Resource Identifier)

Der URI ist die wichtigste Erfindung im Web. Es ist eine Spezifikation, die jedes Web-Programm verwendet (Client oder Server), um das Ziel eines Links zu kennen. Jedes Dokument kann irgendwo am Server liegen und mit Hilfe des URI vom Browser gefunden werden [Berners-Lee und Fischetti, 2000, Seite 39].

Dieses Schema ist ähnelt der im Alltag gebräuchlichen Postleitzahlen, bei denen in Österreich die erste Stelle die entsprechende Region angibt (Bsp.: 8xxx bedeutet Steiermark). Ein URI kann laut Berners-Lee [Berners-Lee und Fischetti, 2000, Seite 39] ähnlich unterteilt werden, wobei Slashes zur Trennung der Teile verwendet werden:

- Das vom Server verwendende Protokoll (HTTP, FTP)
- Der Server, auf dem sich das Dokument befindet (www.myserver.com)
- Ein spezifischer Pfad zum jeweiligen Dokument am Server (/mydocument)

Ein kompletter Pfad zu einem Dokument am Server könnte dann beispielsweise so aussehen:

http://www.myserver.com/mydocument

HTTP (Hypertext Transfer Protocol)

Wie bereits erwähnt dient HTTP zur Kommunikation zwischen Client und Server. Der große Vorteil von HTTP ist, dass der Client dem Server sagen kann, welche Datenformate er unterstützt und dieser dann ein Dokument im entsprechenden Format zurücksendet [Berners-Lee und Fischetti, 2000, Seite 40].

HTML (Hypertext Markup Language)

HTML ist ein Weg, um Hypertext repräsentieren zu können. Handelt es sich bei der nachgefragten URI um eine HTTP-Anfrage, müssen sich Client und Server über das Datenformat einigen, das in Datenpakete unterteilt werden kann und von beiden verstanden wird. Werden keine „besseren“ Formate verstanden, versuchen beide, in HTML zu konvertieren, um den Dateiaustausch zu vollziehen [Berners-Lee und Fischetti, 2000, Seite 41].

HTML wurde laut Berners-Lee [Berners-Lee und Fischetti, 2000, Seite 41f] unter folgenden Aspekten entwickelt:

- Befördern der Struktur eines Hypertext-Dokuments, keine Details über die Darstellung
- Basierend auf SGML
  - Inhalt und Layout voneinander getrennt
  - Dokumenttypdefinitionen zum strukturierten Aufbau
  - Verwendung von Tags
- Der Benutzer soll nie mit HTML-Code in Berührung kommen
  - Der Browser dient als Übersetzer

Vermarktung des Web

Alle beteiligten Mitarbeiter entschieden sich dafür, dass Produkt noch nicht zu verkaufen, sondern zuerst die potentiellen Benutzer von der Idee zu überzeugen. Deshalb erzählten die Mitarbeiter vielen Personen bei CERN von dem Projekt und über den Server (info.cern.ch) wurden Basisinformationen, wie Verwendung und verfügbare Software, verbreitet. Tim Berners-Lee verfasste ein Paper „Hypertext at CERN“, in dem er versuchte, auf die Bedeutung der bereits erfolgten Arbeit aufmerksam zu machen. Auf Wunsch bereits vieler interessierter Leser legte er eine Newsgroup an, um Informationen zum Thema Web auszutauschen. Diese zog schnell viele Interessenten an und verbreitete sich sehr schnell. Das Problem war allerdings, dass das Produkt bis dato nur auf NeXT-Systemen funktionierte. Der bereits vorhandene „Zeilen“-orientierte Browser konnte noch nicht eingesetzt werden, weil dieser in „C“ programmiert war, der restliche Code jedoch in „objective c“. Aufgrund der großen Anteilnahme in der Newsgroup sah sich Berners-Lee in seiner ursprünglichen Idee, das Web für alle zur Verfügung zu stellen, bestätigt. Deshalb entschloss er sich auch dazu, die Basis auf C umzustellen [Berners-Lee und Fischetti, 2000, Seite 43ff].

Im Juli 1991 begann Berners-Lee die Besucherzahlen des Web-Servers aufzuzeichnen. Er merkte, dass sich diese von Juli bis August verzehnfachten. Er ging davon aus, dass sein Web-Projekt früher oder später eine Eigendynamik entwickeln würde und er dieses Projekt irgendwann nur mehr steuern müsste. Aufgrund der großen Beteiligung in den Newsgroups richtete Berners-Lee eine Community ein. Sein Ziel war es, sein Produkt in dieser zu fördern [Berners-Lee und Fischetti, 2000, Seite 49f].

Ein wichtiger Termin für die Vermarktung des World Wide Web war im Dezember 1991. Hier fand eine Konferenz (Hypertext 91) in San Antonio statt, wo eine große Anzahl an interessierten Teilnehmern anwesend war. Deshalb bereitete Berners-Lee mit seinem Kollegen Robert Cailliau eine Demonstration des Web mit Hilfe seines NeXT Computer und eines Modems vor. Die Präsentation bereitete zuerst einige unvorhergesehene Probleme [Berners-Lee und Fischetti, 2000, Seite 50f]:

- Es war keine Telefonleitung vorhanden
  - Sie mussten den Hotelmanager überreden, ein Telefonkabel zum entsprechenden Raum zu legen

- Das mitgebrachte Modem passte nicht in die amerikanische Stromversorgung

Aber auch dieses Problem konnten sie mittels Löten beheben und waren so bereit für die erste Präsentation über das World Wide Web. Bereits zwei Jahre später hatte jede Präsentation auf dieser Konferenz das Thema Web zum Inhalt [Berners-Lee und Fischetti, 2000, Seite 51].

Das einzige was im Bereich des World Wide Web noch fehlte, waren Clients für PC, Unix und Macintosh. ViolaWWW, entwickelt von Pei Wie war bereits ein „point-and-click"-Browser, der u.a. HTML mit Grafiken und Animationen darstellen konnte. Das Problem dabei war allerdings die schwierige Installation am Client [Berners-Lee und Fischetti, 2000, Seite 56f].

Die Entwicklung von Browsern wurde aber immer mehr zu einem Hype. Für X-Window-Systeme gab es schon bald mehrere, wie etwa Erwisa, Viola und Midas. Samba funktionierte bereits teilweise am Mac. Der Konkurrenzkampf stieg an und damit der Versuch, besser zu sein als Andere [Berners-Lee und Fischetti, 2000, Seite 67].

Nach der Lizenzierung von Gopher (1993), einem laut [NETPLANET] menügestützten Informationssystem, sprachen viele Anwender von Verrat und viele große Firmen ließen Gopher einfach fallen. Auch Tim Berners-Lee wollte sein Projekt ursprünglich unter die GPL (General Public License) nehmen, aber das Scheitern von Gopher mit dessen Lizenzen brachte ihn zum Umdenken [Berners-Lee und Fischetti, 2000, Seite 73].

Das World Wide Web wurde immer populärer und immer mehr Internet Service Provider kamen auf [Berners-Lee und Fischetti, 2000, Seite 80]. Auch große Firmen wollten den Zug nicht verpassen und so beschlossen auch Netscape und Microsoft, Browser auf den Markt zu bringen. Diese konnten kostenlos vom Internet heruntergeladen und anschließend installiert werden [Berners-Lee und Fischetti, 2000, Seite 82ff].

Das Web Konsortium

Tim Berners-Lee hatte seine Ziele zur Schaffung eines weltweiten Informationssystems erreicht. Das Internet wuchs immer schneller. Das einzige Ziel, das er sich noch setzte war, dass das Web auch weiterhin seiner ursprünglichen Idee entsprechen sollte [Berners-Lee und Fischetti, 2000, Seite 84]. Deshalb gründete Berners-Lee 1994 das „World Wide Web Consortium"(W3C) am Massachusetts Institute of Technology (MIT). Kurz darauf wurde er Direktor und war verantwortlich für die Entwicklung kompatibler Technologien (Spezifikationen, Richtlinien, Software, Werkzeuge) um das Potenzial des World Wide Web voll auszuschöpfen [Berners-Lee].

Weitere Informationen zu den Tätigkeiten von Tim Berners-Lee können unter [Berners-Lee] nachgelesen werden.

2.1.2.8 Web 2.0

Viele Menschen assoziieren den Begriff Web 2.0 mit den Begriffen Blogs, Wikis, RSS feeds, Social Networks, usw., welche eine soziale Komponente im Web einbringen. Dies wird dadurch gewährleistet, dass jeder die Möglichkeit hat, nicht nur Informationen anzusehen, sondern diese auch zu verändern oder neue Informationen einzufügen [Anderson, 2007, Seite 5].

Das ist aber nichts Neues. Wie bereits in Abschnitt 2.1.2 erwähnt, wollte bereits Tim Berners-Lee dem Benutzer die Möglichkeit bieten, Informationen zu schreiben und zu bearbeiten. Er selbst sagte in einem Interview auf die Frage, ob es einen Unterschied zwischen Web 2.0 und seinem „Web 1.0" in Bezug auf die zwischenmenschliche Interaktion gebe, dass all das bereits in der ursprünglichen Version des World Wide Web möglich war [Anderson, 2007, Seite 5].

Das Problem war lediglich, dass keine Browser entwickelt wurden, die es erlaubten, Inhalte zu bearbeiten. Deshalb kam es auch zu dem verbreiteten Glauben, dass nur eine Gruppe von Leuten Inhalte ins Web stellen und andere diese betrachten können [Anderson, 2007, Seite 5].

So einfach ist der Begriff Web 2.0 aber dann doch nicht definiert. Es stecken noch einige Dinge dahinter, die der Standardbenutzer nicht sieht. Definiert wurde der Begriff Web 2.0 im Jahr 2004 von Dale Dougherty - Vizepräsident von O'Reilly Media Inc., mit dem Ziel, die Wirtschaft zu unterstützen, sowie neue Ideen und Technologien zu entwickeln. Laut [Tim O'Reilly, 2005] und [Anderson, 2007, Seite 14ff] wurde Web 2.0 aus folgenden Gründen so beliebt:

- Das Web als Plattform
  - Es gibt Kooperationen zwischen Webseiten
- Das Nutzen kollektiver Intelligenz
  - Jeder kann seine Meinung einbringen
    - Hohe Nutzung
  - Die Firmen können daraus Nutzen ´ziehen
    - Bsp.: Ebay stellt nur den Rahmen bereit und wächst auf natürliche Weise durch die intensive Nutzung
    - Bsp.: Amazon zeigt dem Benutzer anhand bereits gekaufter Artikel, mögliche für ihn interessante Produkte
- Datenvielfalt
  - Nahezu alles ist heutzutage online
  - Immer mehr Daten werden generiert
  - Problem: Wem gehören diese Daten? Führt das eventuell zu einer Zentralisierung des Web?
- Das Web 2.0 ist für jeden offen
- Soll für den Benutzer überall verfügbar sein
  - Architekturen werden angepasst, um ein Service für jedermann verfügbar zu machen
  - z.B.: Programme für Smartphones

Vor allem im Bereich der Entwicklung von Web 2.0-Applikationen wurden einige Standards und Technologien (wie etwa AJAX oder APIs) definiert, welche die Erstellung von „Desktop-ähnlichen" Applikationen erleichtern soll [Anderson, 2007, Seite 26ff].

2.1.2.9 Semantic Web

Semantic Web steckt zwar noch in den Fußstapfen [Palmer, 2001], aber es könnte durchaus die Zukunft des Web sein und deshalb möchte ich hier überblicksmäßig darauf eingehen.

Laut [Berners-Lee et al., 2001] ist das Web heute so konstruiert, um Informationen für den Menschen darzustellen. Das *Semantic Web* soll den Inhalt des Web so aufbereiten, dass die Inhalte vom Computer gelesen und die Verbindungen untereinander verstanden werden. Dies soll dem Benutzer viel Arbeit beim Durchsuchen von Informationen abnehmen.

Laut [Palmer, 2001] und [Berners-Lee und Fischetti, 2000, Seite 177] ist Semantic Web ein Netz von Informationen. Dieses ist so verlinkt, dass es direkt oder indirekt von Maschinen abgearbeitet werden kann.

Der Unterschied zum jetzigen Web soll anhand eines Beispiels verdeutlicht werden. Am klarsten wird der Unterschied durch eine Suche in einer Suchmaschine. Angenommen der Benutzer sucht nach dem Buch „Hamlet“ vom Autor „Hamburger“. Wird nach den Begriffen „Hamlet“ und „Hamburger“ gesucht, werden zuerst Ergebnisse von einer amerikanischen Restaurantkette namens „Hamburger Hamlet“ angezeigt. Viele Ergebnisse werden Aufführungen vom Stück Hamlet in Hamburg sein. Eigentlich interessierte den Benutzer aber das Buch [e-teaching.org, 2008].

Ein weiteres Beispiel wäre, eine Frage an das Web zu stellen. Wie zum Beispiel: „Gibt es im Skigebiet Schladming noch ein Hotel für nächstes Wochenende für weniger als 50 Euro pro Nacht?“ Nach den Grundideen des Semantic Web wäre es dem Computer möglich, solche Fragen durch die Verknüpfungen der Informationen zu beantworten.

Die zuvor erwähnten Ergebnisse könnten eigentlich außer Acht gelassen werden. Im jetzigen Web kennt der Computer die notwendigen Verbindungen der Informationen allerdings nicht, um dem Benutzer das richtige Ergebnis zu zeigen und die anderen zu ignorieren. Er weiß nicht, dass „Hamlet“ ein Buch ist und vom Autor „Hamburger“ geschrieben wurde. Genau diese Lücke soll das Semantic Web schließen.

Semantic Web ist laut [Berners-Lee et al., 2001] kein eigenes Web, sondern eine Erweiterung des bereits bestehenden.

Verbindungen zwischen Dokumenten im Netz und deren Beziehungen werden im Web als **Ontologien** bezeichnet [Berners-Lee et al., 2001].

Um eine Ontologie darzustellen, gibt es bereits eine entwickelte Sprache: das RDF (Resource Description Framework). Abbildung 2.2 zeigt ein Beispiel, wie ein RDF-Graph aussehen könnte. Je mehr Ontologien sich anhäufen, desto unübersichtlicher wird die entsprechende Datei.

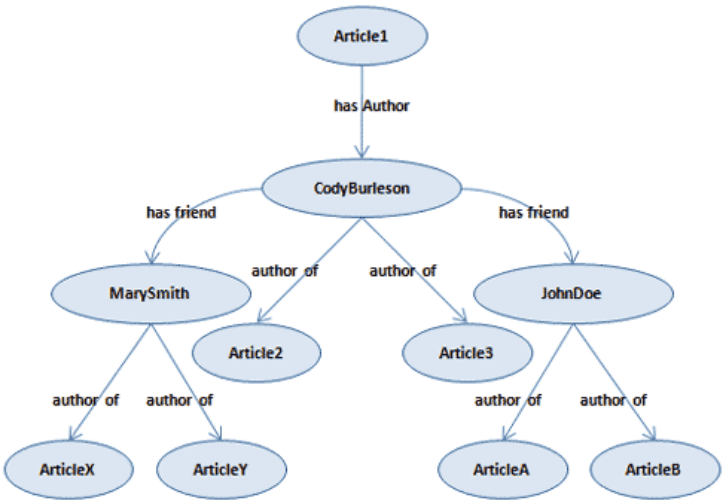

Abbildung 2.2: Beispiel für einen RDF-Graph [Semantic Focus, 2007].

Der Code von RDF sieht ähnlich aus wie XML und die Beziehungen der einzelnen Elemente werden über deren URIs angegeben. Unter `http://www.altova.com/semanticworks.html` gibt es bereits einen graphischen Editor, um RDF-Dateien zu erstellen.

Die Zukunft von Semantic Web steht noch in den Sternen. Dieses Kapitel sollte nur einen Überblick geben, wie das Web in Zukunft aussehen könnte und was gegenüber dem heutigen Web noch verbesserungswürdig ist. Ich möchte an dieser Stelle nicht detaillierter auf das Thema Semantic Web eingehen. Interessierte Leser möchte ich hier auf das Buch „Semantic Web: Grundlagen“ von Pascal Hitzler verweisen [Hitzler et al., 2007].

2.2 Statistiken

Laut [Pruscha, 2005, Seite VII] dringt die Statistik in die Bereiche Technik, Naturwissenschaft, Medizin, u.a. weiter vor, wie etwa folgende Beispiele zeigen:

- Wirksamkeitsprüfung von Medikamenten
- Qualitätsprüfung von technischen Apparaturen
- Ermittlung der Faktoren, die Schäden im menschlichen Körper bewirken
- Voraussage von Naturereignissen
- Erkennung klimatischer Entwicklungen

Statistik gewinnt im Alltag immer mehr an Bedeutung und kann sehr vielseitig eingesetzt werden. Was genau bedeutet aber eigentlich der Begriff Statistik? Der folgende Abschnitt soll mehrere Definitionen des Begriffs zeigen.

2.2.1 Überblick und Definition des Begriffes „Statistik“

> „Eine Wissenschaftliche Disziplin, deren Gegenstand die Entwicklung und Anwendung formaler Methoden zur Gewinnung, Beschreibung und Analyse sowie zur Beurteilung quantitativer Beobachtungen (Daten) ist [Vogel, 2005, Seite 3].“

Lothar Sachs und Jürgen Hedderich [Jürgen Hedderich, 2006, Seite 1] definieren den Begriff Statistik als

> „die Lehre von Variabilität / Streuung in Beobachtungen“

und als

> „die Kunst, Daten zu gewinnen, darzustellen, zu analysieren und zu interpretieren um zu neuem Wissen zu gelangen.“

Mit dem Begriff „Statistik“ werden meist umfangreiche Tabellen und grafische Darstellungen, die beliebige Sachverhalte verdeutlichen sollen, assoziiert. Dies ist allerdings nur ein Teilbereich der Statistik, welcher als „deskriptive“ Statistik bezeichnet wird. Neben dieser gibt es allerdings noch weitere wichtige Teilbereiche der Statistik [Dutter, 2010].

Der folgende Abschnitt soll einen Überblick über die einzelnen Bereiche der Statistik vermitteln und diese überblicksmäßig beschreiben. Auf die mathematischen Aspekte wird dabei nicht näher eingegangen.

2.2.2 Teilbereiche der Statistik

Abbildung 2.3 zeigt eine Übersicht der Teilbereiche der Statistik.

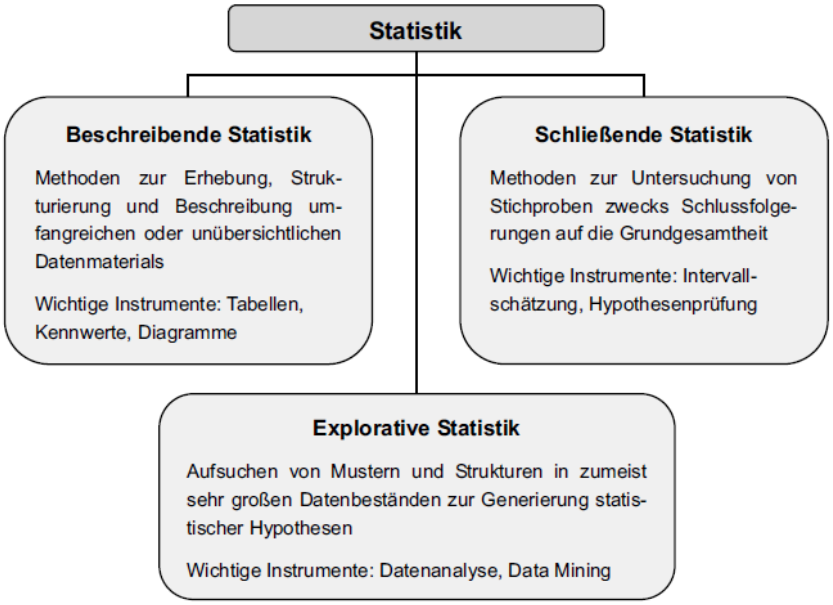

Abbildung 2.3: Teilbereiche der Statistik [Udo Bankhofer, 2007, Seite 4].

2.2.2.1 Explorative Statistik

John W. Tukey, ein amerikanischer Statistiker, beginnt sein Buch „Exploratory Data Analysis“ [Tukey, 1977] mit folgendem Satz:

> „Exploratory data analysis is detective work.“

Die explorative Datenanalyse steht also zu Beginn der statistischen Arbeiten. Laut [Jürgen Hedderich, 2006, Seite 9] und [NIST, 2003] gehören folgende Punkte zu den Aufgabenbereichen von EDA:

- Auffinden von Mustern
- Darstellung von Daten
- Erkennen von Besonderheiten und wichtigen Variablen
- Annahmen testen
- Suche nach neuen Möglichkeiten
- Optimale Einstellungen finden

Alle diese Methoden sollen den Umgang mit großen Datenmengen erleichtern, indem Muster und Strukturen erkennbar gemacht werden. Eingesetzt wird die EDA laut [Jürgen Hedderich, 2006, Seite 10] bei folgenden Problemstellungen:

- Unklare Fragestellung
- Unbekannte Grundgesamtheit [5]
- Kaum geplante Datenerhebung

[5] http://www.sdi-research.at/lexikon/grundgesamtheit.html

- Unklarheit über die Auswahl des Modells
- Quantitativ nicht fassbare Aussagegenauigkeit

Wichtige Intrumente in der explorativen Statistik sind laut [Udo Bankhofer, 2007, Seite 4]:

- Datenanalyse
- Data Mining (siehe Abschnitt 2.3.2)

2.2.2.2 Beschreibende (deskriptive) Statistik

Befasst sich laut [Jürgen Hedderich, 2006, Seite 8] mit der Untersuchung und Beschreibung (möglichst) der ganzen Grundgesamtheit. Diese soll einfach und verständlich unter Zuhilfenahme grafischer Methoden zur Darstellung der Resultate dienen.
Unterstützende Instrumente sind laut [Udo Bankhofer, 2007, Seite 4]:

- Tabellen
- Diagramme
- Kennwerte

Im folgenden Bereich wird eine Übersicht mit den wichtigsten Begriffen in der beschreibenden Statistik dargestellt. Vorweg sei erwähnt, dass diese nicht detailliert betrachtet werden [6]. Laut [Jürgen Hedderich, 2006, Seite 55ff] gehören folgende Begriffe dazu:

- Häufigkeiten
  - Absolute und relative Häufigkeiten
  - Sinnvolle Verhältniszahlen
  - Prozentwerte
  - Tortendiagramme
  - Balkendiagramme
  - Tabellen
  - Bedingte Häufigkeiten
- Ordinaldaten
  - Quartile
  - Quantile
  - Streuung skalierter Daten
  - Punktdiagramm und Box-Plot
- Metrische Daten
  - Mittelwert
  - Standardabweichung
  - Varianz
  - etc.

[6] Mehr Informationen dazu gibt es im Buch [Jürgen Hedderich, 2006, Seite 55ff]

- Häufigkeitsverteilung
  - Histogramm
  - Stamm-Blatt Darstellung
- Konzentration
  - Gini Index
- Maßzahlen für den Zusammenhang
  - Punktwolken
  - Die lineare Regression
  - etc.
- Nichtlineare Regression

Klassifiziert werden deskriptive Statistiken nach folgenden Gesichtspunkten [Jürgen Hedderich, 2006, Seite 55]:

- Maßzahlen, die eine zentrale Tendenz (Lage) der vorhandenen Daten ausdrücken
- Erfassung von Streuung oder Variabilität in Beobachtungen
- Ausdrücken der Form der Verteilung
- Zusammenhang zwischen mehreren Beobachtungsreihen

2.2.2.3 Beurteilende (Schließende) Statistik

Die schließende Statistik untersucht im Gegensatz zur beschreibenden Statistik nur den Teil der Grundgesamtheit, dessen Eigenschaften repräsentativ für das Ergebnis sein sollen [Jürgen Hedderich, 2006, Seite 8].

Der Name „schließende Statistik“ kommt daher, weil Schlussfolgerungen von Stichproben auf die Grundgesamtheit gezogen werden sollen. Ziel ist es, vom Besonderen auf das Allgemeine zu schließen [Udo Bankhofer, 2007, Seite 91].

Laut [Jürgen Hedderich, 2006, Seite 2] basiert die beurteilende Statistik auf Wahrscheinlichkeitsrechnung und dient zur Beschreibung zufallsbedingter oder stochastischer Experimente.

Wichtige Instrumente im Bereich der beurteilenden Statistik sind nach [Udo Bankhofer, 2007, Seite 4] und [Lohninger, 2010]:

- Intervallschätzung
- Angabe von Konfidenzintervallen
- Hypothensenprüfung (Tests)

2.2.3 Zusammenfassung

Zusammenfassend lässt sich sagen, dass die Statistik aus drei Teilbereichen besteht, deren Aufgaben es sind, Muster und Strukturen im Datendschungel zu erkennen (explorativ), die Daten zu beschreiben und zu strukturieren (beschreibend) und Schlussfolgerungen einzelner Stichproben bzw. Datensätze auf die Grundgesamtheit zu ziehen (schließend), um Hypothesen zu überprüfen und daraus allgemeingültige Tatsachen bzw. Wissen zu generieren.

2.3 Visualisierung von Daten

In diesem Abschnitt wird das Thema „Visualisierung von Daten“ detailliert betrachtet. Bei kleinen Datenmengen stellt die Visualisierung keine besonders große Herausforderung dar. Ganz anders sieht das aber bei sehr großer Anzahl von Datenpunkten aus.

Wissenschaftler fanden heraus, dass bereits im Jahr 2002 ca. 1 Exabyte (=1 Million Terabyte) an Daten produziert wurde. Das bedeutet umgerechnet, dass in den Jahren 2002 - 2005 mehr Daten generiert wurden, als in der ganzen menschlichen Geschichte davor. Ein Grund für die riesigen Datenmengen ist, dass diverse Prozesse im Alltag (z.B.: Zahlung per Kreditkarte, etc.) Computer-überwacht ablaufen und dieser eine große Anzahl von Parametern mit aufzeichnet. Alle diese Daten werden dann auch gespeichert und nicht wieder gelöscht, weil der Mensch immer daran glaubt, dass hinter jedem einzelnen dieser Parameter potenziell verwertbare Informationen stecken und diese große Bedeutung erreichen könnten [Keim, 2002, Seite 1].

Die Anzahl an Daten kann also sehr groß werden. Das Problem ist, diese Menge so darzustellen, dass auch wirklich verwertbare Informationen generiert werden können - die Anzahl an Daten ist riesig, die Aufnahmefähigkeiten des Menschen aber begrenzt. Um mit dieser Datenmenge umgehen zu lernen bzw. um etwas interpretieren zu können, gibt es laut [Keim, 2002, Seite 1f]zwei Ansätze:

1. Visual Data Exploration (Siehe Abschnitt 2.3.1)
2. Visual Data Mining(Siehe Abschnitt 2.3.2)

2.3.1 Visuelles Durchsuchen von Daten (Visual Data Exploration)

Laut [Keim, 2002, Seite 2] ist Visual Data Exploration ein Prozess, der im Wesentlichen immer die folgenden drei Schritte durchläuft:

1. Überblick verschaffen
2. Zoomen und Filtern
3. Details erarbeiten

Hat sich der Anwender einen Überblick in Schritt 1 verschafft, geht es darum, die für ihn interessanten Bereiche herauszufiltern. Um genauere Informationen zu erlangen, muss er diese ausgewählten Bereiche detailliert betrachten. Wichtig ist dabei aber immer, den Gesamtüberblick im Auge zu behalten. Eine entscheidende Rolle dabei spielt also der Anwender selbst [Keim, 2002, Seite 1f].

Ziel der VDE ist es die positiven Eigenschaften des Menschen wie Flexibilität, Kreativität und Wissen mit denen des Computers(z.B. Speicherkapazität und Rechenleistung) miteinander zu verbinden um die Daten für den Anwender interpretierbar zu machen [Keim, 2002, Seite 1].

Ein weiterer wichtiger Punkt ist die Interaktion mit den Daten, um mögliche Schlussfolgerungen ziehen zu können. Gelingt es dem Anwender, eine Hypothese dadurch aufzustellen, kann diese wiederum mit Visual Data Exploration oder über die Anwendung statistischer Methoden verifiziert werden. Vorteile der VDE sind laut [Keim, 2002, Seite 1]:

- VDA ist intuitiv
- Erfordert kein Verständnis komplexer, mathematischer Algorithmen
- Liefert oft bessere Ergebnisse
- Der Benutzer ist DIREKT einbezogen

2.3.2 Data Mining

2.3.2.1 Definition

Laut [Ferreira de Oliveira und Levkowitz, 2003, Seite 378] ist Data Mining definiert als die Erkennung von Mustern oder Modellen aus beobachteten Daten. Dabei ist DM oft Teil eines Prozesses welcher dazu dient, nützliches Wissen aus den Daten zu extrahieren. Der in Abschnitt 2.3.1 definierte Begriff VDA kann dabei eine entscheidende, unterstützende Rolle spielen.

2.3.2.2 Klassifikation von Data Mining-Techniken

Visualisierungstechniken wie „x-y plots“, Liniendiagramme oder Histogramme werden beim visuellen Durchsuchen von Daten eingesetzt. Deren Nachteil ist die Beschränkung auf relativ niedrig-dimensionale Datensätzen. Im letzten Jahrzehnt wurden viele Visualisierungstechniken entwickelt, die es ermöglichen, Daten höherer Dimensionen zu betrachten und analysieren [Keim, 2002, Seite 2].

Diese können laut [Keim, 2002, Seite 2ff] in drei Klassen aufgeteilt werden:

1. Art der Daten
   (a) Eindimensional
   (b) Zwei - oder dreidimensional
   (c) Multidimensional
      i. Viele Datenpunkte besitzen mehr als drei Attribute
   (d) Text und Hypertext
      i. Kann nicht als Zahl dargestellt werden
      ii. Die Standard-Visualisierungstechniken funktionieren hier nicht
   (e) Hierarchien und Graphen
      i. Ein Graph besteht aus einer Menge an Objekten (Knoten) und den Verbindungen zwischen den Knoten (Kanten)
      ii. z.B.: Hyperlinks im Internet
   (f) Algorithmen und Software
      i. Soll den Informationsfluss eines Programmes anzeigen um den Code besser zu verstehen
2. Visualisierungstechniken
   (a) Zusätzlich zu den Standardvisualisierungstechniken wie etwa „x-y-Plots“, Säulendiagramm, Liniendiagramm, usw. gibt es eine Vielzahl an Techniken für den multidimensionalen Bereich (>3)

   Aus Gründen der Komplexität möchte ich diese hier nicht detaillierter betrachten. Interessierte Leser seien an dieser Stelle auf [Keim, 2002, Seite 4f] verwiesen.
3. Techniken zur Interaktion und Verformung
   (a) Interaktionstechniken
      i. Ermöglichen es, mit dem Benutzer direkt in der Visualisierung zu interagieren
      ii. Zum Beispiel wenn der Anwender einen bestimmten Bereich vergrößern möchte, um diesen genauer zu betrachten
   (b) Verformungstechniken

i. Dienen dazu, trotz Fokus auf einzelne Bereiche den Überblick auf das Gesamte zu behalten
ii. Unterstützen die Visual Data Exploration

(c) Beispiele für Interaktionstechniken sind
i. Interaktives Filtern
A. Funktioniert ähnlich, als würde ein bestimmter Ausschnitt durch ein Vergrößerungsglas betrachtet werden
ii. Interaktives Zoomen
A. Vergrößert die Objekte und kann auch die Repräsentation der Daten ändern
B. Beispiel: Datenpunkte werden als Pixel angezeigt. Wird das Bild via Zoom vergrößert, werden Beschriftungen von Objekten sichtbar

In diesem Abschnitt wurde das Thema Visualisierung von Daten behandelt. Wie bereits erwähnt kann durch die große Anzahl von Daten und Dimensionen ein großes Problem für die Interpretation der Daten oder der Verifizierung von Hypothesen entstehen. Entsprechende Visualisierungstechniken werden also immer wichtiger. Im Zuge des Projekts ist noch keine dieser Techniken relevant, da wir uns zum jetzigen Zeitpunkt nur im zweidimensionalen Bereich bewegen (siehe Abschnitt 3) und dort Punkt-, Linien-, Balkendiagramme, etc. momentan für die Betrachtung ausreichend sind.

Kapitel 3

Anforderungen an die Applikation

In diesem Abschnitt werden die Anforderungen an die Applikation, die im Zuge des Projektes erstellt wird, aufgelistet. Diese sind nach Absprache mit meinem Betreuer Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic folgende und werden in den nächsten Abschnitten detailliert betrachtet:

1. Allgemeine Anforderungen
2. Anforderungen an die Statistikgenerierung
3. Anforderungen an die Webapplikation (User-Interface)

3.1 Allgemeine Anforderungen

Ziel des Projekts ist, die Implementierung eines Frameworks zur Generierung von Statistiken auf Basis von Input-Dateien. Diese werden vom Benutzer an die Applikation übergeben. Als Benutzerschnittstelle (User-Interface) soll dabei eine Java Webapplikation dienen, die dem Benutzer den Upload der Input-Datei und die Verwaltung der Ziel-Ordnerstruktur am Server ermöglicht. Die Anforderungen an die Statistikgenerierung und die Webapplikation werden in den Abschnitten 3.2 und 3.3 detailliert betrachtet. Die wichtigste Anforderung im Bereich der „allgemeinen Anforderungen“ ist die „Abstraktion der Applikation“, vor allem im Bereich der Statistikgenerierung.

3.1.1 Abstraktion der Applikation

Wie bereits erwähnt, dient die Applikation zur Statistikgenerierung. Da dieses Thema sehr komplex und umfangreich ist, soll das Ergebnis dieses Projekts, ein Framework zur Generierung von Statistiken sein. Deshalb werden vorerst nur einige Arten von Statistiken unterstützt. Ziel ist also, den Teil der Statistikgenerierung so allgemein wie möglich zu halten, um möglichst einfach nachträgliche Erweiterungen und Veränderungen hinzuzufügen. Um dieser Anforderung gerecht zu werden, gibt es im Java-Sourcecode eine abstrakte „Statistikklasse“, die allgemeine Einstellungen und Funktionen, wie etwa das Setzen des Ausgabetyps, oder das Zeichnen, beinhaltet. Davon abgeleitet gibt es die konkreten Statistiken (im Moment die „frequency“-, „percent“- und „difference“-Statistik), welche die statistikspezifischen Einstellungen beinhalten, wie etwa ein Objekt des zu zeichnenden „Plots“. Diese Struktur soll das Hinzufügen neuer Statistiken erleichtern und den Programmieraufwand minimieren.

3.2 Anforderungen zur Statistikgenerierung

Das Tool zur Generierung der Statistiken in der Applikation soll folgende Anforderungen erfüllen:

1. Keine Kosten
2. Gute Anbindung zu Java
3. Möglichkeit, das Tool über einen Java-Prozess zu starten
4. Generierung von zwei- und dreidimensionalen Darstellungen
5. Möglichkeit, Plots aus Input-Dateien zu erzeugen

Es gibt eine Menge an Programmen, die es ermöglichen graphische Darstellungen zu generieren, wie etwa:

1. gnuplot
2. MATLAB
3. Octave
4. Google Charts
5. usw.

Die zuvor erwähnten Tools wurden im Zuge der Arbeit genau unter die Lupe genommen und entsprechend den Anforderungen an die Applikation untersucht. Dabei kam ich auf folgende Ergebnisse:

Mit „Google Charts“ [1] können relativ einfach graphische Darstellungen generiert werden. Bei genauerem Betrachten wird auch die Funktionsweise dahinter ersichtlich. Der Benutzer muss eine „URL“ basteln, welche die darzustellenden Daten repräsentiert und die Parameter über einen „Request“ an „google“ senden. Als Antwort bekommt man das passende „png“-File retourniert. Google Charts ist ebenfalls gratis, aber aus Gründen der Datensicherheit nicht für das Projekt geeignet. Des Weiteren konnte ich keine Möglichkeit finden, um Input-Dateien als Parameter zu übergeben. Ein Problem könnten auch die - bei großen Datenmengen und vielen Einstellungen - sehr komplex und unübersichtlich werdenden „URLs“ darstellen. Ein weiterer wichtiger Punkt, der nicht erfüllt werden kann, ist das Speichern der Bilddatei. Diese kann nur über einen „Tag“ im HTML-Code eingebunden und angezeigt werden. Google Charts eignet sich gut, um schnell und einfach, simple Datensätze darzustellen, aber für den Einsatz im Projekt ist es aus den zuvor genannten Gründen unbrauchbar. Für interessierte Leser ist unter http://psychopyko.com/tutorial/how-to-use-google-charts/ ein hilfreiches Tutorial zu finden. Folgendes Beispiel zeigt eine, von mir generierte „URL“:

```
http://chart.apis.google.com/chart
   ?chbh=a,5
   &chs=180x140
   &cht=bvs
   &chd=s:MYGJIH
```

[1] http://code.google.com/intl/de-DE/apis/chart/docs/chart_wizard.html

„Cht“ gibt dabei den Typ des Diagramms an („bar vertical“), „chs“ die Größe, „chbh“ die Breite der einzelnen Balken und den Abstand zwischen zwei Balken. „Chd“ gibt die darzustellenden Daten in kodierter Form an.

Die nun folgenden Tools unterscheiden sich grundsätzlich in der Funktionsweise von „Google Charts“. Alle folgenden Tools können am Rechner installiert und ausgeführt werden.

MATLAB ist sehr umfangreich und komplex. Es gibt viele Möglichkeiten um graphische Darstellungen zu generieren, z.B. durch selbst definierte Funktionen, vordefinierte Funktionen oder Kombinationen daraus. Input-Datein können ebenfalls visualisiert werden. Das Problem bei MATLAB ist, dass schon die Grundvoraussetzung, um im Projekt Anwendung finden zu können, nicht gegeben ist - MATLAB ist kostenpflichtig.

Octave bietet ebenfalls einen derartigen Funktionsumfang und ist auch gratis. Die Funktionen zum zeichnen einer graphischen Darstellung sind exakt gleich, mit denen von gnuplot. Octave verwendet gnuplot als Werkzeug, um Plots zu generieren. [2]. Nicht sehr ausgereift ist die Anbindung von Octave an Java.

Gnuplot erfüllt alle zuvor erwähnten Anforderungen. Die Möglichkeiten mit gnuplot Daten zu visualisieren sind groß und es fallen keine Kosten bei der Verwendung von gnuplot an. Eine möglichst allgemein gehaltene Anbindung an Java ist mit „jgnuplot“ gegeben, welches in praktischen Teil auch zum Einsatz kam und in Abschnitt 6.1.1 detailliert beschrieben wird. „Jgnuplot“ ist dabei die Schnittstelle zwischen Java und „gnuplot“.

Der praktische Teil beinhaltet im Moment nur die Darstellung zweidimensionaler Graphen, soll aber offen für Erweiterungen (z.B. dreidimensionalen Darstellungen) sein. Da gnuplot alle Anforderungen erfüllt, habe ich mich dazu entschlossen, die graphischen Darstellungen damit umzusetzen. Aus diesem Grund wird das Thema „gnuplot“ in Abschnitt 4 detailliert betrachtet. Dieses soll einen Überblick über die Möglichkeiten von „gnuplot“ geben und den Einstieg in „gnuplot“ erleichtern.

Tabelle 3.1 zeigt eine Übersicht, der zuvor erwähnten Tools, in Bezug auf die gestellten Anforderungen [3].

[2] http://sunsite.univie.ac.at/textbooks/octave/octave_15.html

[3] Anm.: ***** = sehr gut, *** = durchschnittlich, * = nicht genügend

	Matlab	Gnuplot	Octave	Google Charts
Gratis	Nein	Ja	Ja	Ja
General Public License (GPL)	Nein	Nein, gnuplot entschied sich für eine eigene „open-source“-Lizenz.	Ja	Nein, eigene Lizenz, die es erlaubt, deren Service zu nutzen
Java Anbindung	***	*****	***	****
Graphische Darstellungen aus Input-Dateien erzeugen	Ja	Ja	Ja	Nein
Über Java-Prozess ausführbar	Ja	Ja	Ja	Nein
Erstellung zweidimensionaler Graphen	Ja	Ja	Ja	Ja
Erstellung dreidimensionaler Graphen	Ja	Ja	Ja	Nein
Auf Rechner installierbar	Ja	Ja	Ja	Nein
Ranking	4	1	2	3

Tabelle 3.1: Vergleich der Werkzeuge zur Statistikgenerierung.

3.3 Anforderungen an die Webapplikation

Laut den Anforderungen (siehe Abschnitt 3.1) soll eine Java Web-Applikation erstellt werden. Dieser Abschnitt soll einen Überblick darüber geben, welche Frameworks es in Java gibt, welches letztlich verwendet wurde und warum gerade dieses zur Umsetzung des Projekts ausgewählt wurde.

3.3.1 Web Application Frameworks

Framework: Ein Framework ist eine definierte, unterstützende Struktur, in der andere Software-Applikationen organisiert und entwickelt werden können. Frameworks können Code-Bibliotheken, Skriptsprachen, Schnittstellen oder andere Softwarepakete beinhalten, die dem Benutzer dabei helfen, die Entwicklung zu optimieren und verschiedene Teile eines Softwaresystems zusammenzufügen [Shan und Hua, 2006, Seite 378].

Web Application Framework (WAF): Laut [Shan und Hua, 2006, Seite 379] ist ein WAF ein wiederverwendbares Grundgerüst, das speziell zum Erzeugen eigener Webapplikationen verwendet werden kann. Die Web Browser werden dabei mit dem HTTP bedient. Ein weiterer wichtiger Aspekt ist das von Grund auf unterstützte MVC (Model View Controller) Design Pattern, welches in Abschnitt 5.1.2.5 detaillierter betrachtet wird. Ein WAF kann wiederum andere nützliche Teile wie etwa ein User Inferface Framework oder eine UI-Komponentenbibliothek beinhalten, welche eine schnelle Entwicklung sicherstellen sollen.

Die Gründe ein Web Application Framework zu verwenden sind laut [Shan und Hua, 2006, Seite 379] folgende:

- Reduzierung von
  - Zeitaufwand
  - Anstrengung

  - Benötigter Mittel
- Offene Architektur basierend auf anerkannten Standards (z.B.: Java, XML, Servlet, JSP)
- Erhöht den Fokus auf die Problemstellungen der Applikation
- Sehr viele Grundanforderungen vieler Applikationen werden unterstützt
  - Sicheres Login
  - User Management und Gruppen
  - Zugriffsberechtigungen
- Zukunftssicherheit

Abbildung 3.1 zeigt einen Überblick einiger Web Application Frameworks mit deren Grundeigenschaften und Attributen.

WAF Type	Primary Attributes	Product	Key Features	Guidelines/Trends
Request-based	• Controllers and actions to handle incoming requests • Stateless • Logic mapped to URLs	Struts	▪ Extend Java Servlet API ▪ Adopt the "Model 2" approach ▪ Action and ActionForm	- Decomposed to two frameworks: Struts Action Framework (merger with WebWork) and Struts Shale Framework (incorporation of JSF) - Predominant front controller implementation
		Beehive	▪ Metadata-driven programming model ▪ NetUI Page Flow ▪ Controls	- Required JDK 1.5 - NetUI Page Flow is based on Struts - More mature than WebFlow in Spring - Pollinate is an Eclipse-based IDE
		Stripes	▪ Lightweight presentation framework ▪ Built-in support for multiple events per form ▪ Transparent file upload	- Good for building wizard forms in an application - Indexed property support - Reuse ActionBeans as view helpers
Component-based	• Logic encapsulated in components • State handled in component instance • Similar to GUI development	JSF	▪ User interface components ▪ API for handling states, events, server-side evaluation ▪ 2 JSP custom tag libraries	- MyFaces is an open source implementation - Reference implementation offered by Sun Microsystems - MyEclipse Enterprise Workbench support
		Tapestry	▪ Component-based structure to construct pages ▪ Templates ▪ Special HTML attribute to denote components	- Applicable in niche areas - Easy editing of templates with ordinary HTML editors - MyEclipse Enterprise Workbench support
		Wicket	▪ Mock-up page via WYSIWYG HTML tools ▪ POJO data beans ▪ Transparent state management with no XML configurations	- Combination of the component model in Echo with the Special HTML attribute to denote components in Tapestry - Use POJO to automate the server-side state management
Hybrid	• Data and logic flow in a request-based model • Component object model to handle actions and controllers • Combination of raw control and reusability in two types of frameworks	RIFE	▪ Logic-less HTML templates ▪ Uniform component model ▪ Metaprogramming ▪ Multi-dimensional conversational state management with scoping ▪ Language-independent template engine ▪ Persistence layer with content management integration and versioning	- Work with domain-specific API and language - Automatic generation of CRUD - Rich dynamic metadata APIs for JavaBeans instances and their properties - Integration with DWR
Meta	• Core interfaces for common services • Extensible backbone for integrating components and services • Inversion of Control pattern	Keel	▪ Multi-layer structure: security, database abstraction, messaging, business logic, and user interface layer ▪ Strict separation of roles and interfaces ▪ Auto-configuration	- Service-based architecture - Extensible structure - Open platform - Anti-patterns collected - Leverage Avalon framework
		Spring	▪ Light-weight container for automated configuration and wiring of application objects ▪ Abstraction layer for transaction management ▪ AOP functionality	- A preferred baseline - Incorporate an appropriate front controller implementation in design - Spring IDE is a graphical user interface for the configuration files used by the Spring Framework
RIA-based	• Rich Internet Application • Client-side container model • Stateful user interaction	DWR	▪ Invoking Java objects at the server side from JavaScript in the browser ▪ Dynamically generating JavaScript code-based Java classes	- Support of DWR in request- and component-based frameworks - Reverse Ajax - Pageable and sortable lists
		Echo2	▪ Ajax-based rendering engine ▪ Object-oriented and event-driven paradigm for user interface development ▪ Server push technology	- Event-oriented design - Support modal dialogs - EchoStudio2 is an Eclipse plug-in for visual development: Form Editor and StyleSheet Editor
		JSON-RPC-Java	▪ JSON-RPC implementation in Java ▪ Light-weight RPC JavaScript client ▪ Basic ORB (Object Request Broker) functionality	- Lightweight JSON format instead of XML for speed - Remote procedure call protocol - Transparent marshalling/unmarshalling of primitive types

Abbildung 3.1: Web Application Frameworks im Überblick [Shan und Hua, 2006, 384].

Die Gründe für die Verwendung von JSF und RichFaces als WAF in der Applikation werden in Abschnitt 3.3.3 detailliert beschrieben.

3.3.2 Funktionelle Anforderungen an die Webapplikation

Wie bereits in Abschnitt 3.1 erwähnt wurde, soll eine Java-Webapplikation als Benutzerschnittstelle (User Interface) dienen. In diesem Abschnitt werden die funktionellen Anforderungen der Webapplikation detailliert betrachtet:

3.3.2.1 Ordnerstruktur anzeigen

Anforderung 1: Ordnerstruktur anzeigen

Dem Benutzer soll die Ordnerstruktur vom Server, inklusive der darin vorhandenen Dateien angezeigt werden. Als „root"-Ordner dient, der entsprechend spezifizierte Ordner der „Property-Datei" (siehe Abschnitt 6.2.1. Dieser muss auf einen Ordner in der Projekthirarchie des „Application Servers" (z.B. bei Verwendung eines Tomcat Servers: „TOMCAT_HOME/webapps/projektname") zeigen. Um nicht immer den kompletten Pfad anzeigen zu müssen, muss ein Unterordner dieses Ordners im „Property-File" angegeben werden (siehe Abschnitt 6.2.1).

Anforderung 2: Unterstützung beliebig vieler Ebenen

Ausgehend von dem spezifizierten „root"-Ordner sollen dem Benutzer alle vorhandenen Ebenen der Ordnerstruktur angezeigt werden.

3.3.2.2 Ordnerstruktur bearbeiten

Der Benutzer soll die Möglichkeit haben Ordner anzulegen. Dabei gibt es drei unterschiedliche Varianten.

Anforderung 3: Erstellen eines Hauptordners

Anlegen eines Hauptordners bedeutet, dass ein Ordner im spezifizierten „root-Unterordner" angelegt wird. Der „root-Unterordner" ist der zuvor beschriebene Unterordner im Projekt, der zum Zwecke der Übersichtlichkeit eingefügt wurde.

Anforderung 4: Anlegen eines Unterordners in jedem vorhandenen Hauptordner (zwei Ebenen unter dem „root folder")

Beim Anlegen eines Unterordners in jedem vorhandenen Hauptordner, wird durch alle zuvor angelegten Hauptordner iteriert. Ist der Unterordner mit dem Namen bereits vorhanden, passiert nichts. Ansonsten wird dieser Ordner neu angelegt.

Anforderung 5: Einen Unterordner im aktuell ausgewählten Ordner erzeugen

Die beiden zuvor erwähnten Methoden beschränken sich auf zwei unterschiedliche Ebenen. Um dem Benutzer die Möglichkeit zu geben, Ordner beliebig zu Strukturieren, soll er die Möglichkeit haben, einen Unterordner im aktuell ausgewählten Ordner anzulegen.

3.3.2.3 Input-Dateien hochladen

Anforderung 6: Dateien hochladen

Die Aktion „Datei hochladen“ soll eine Input-Datei in einen spezifizierten Ordner hochladen. Aus diesem Grund ist die „Upload“-Komponente nur verfügbar, wenn der Benutzer einen Ordner ausgewählt hat, in dem er diese speichern möchte. Es werden von der Applikation nur Dateien mit der Endung „.dat“ akzeptiert. Jede Art von Statistik kann immer nur einmal in jedem Ordner gespeichert werden. Aus diesem Grund ist der Dateiname verantwortlich dafür, welche Art von Statistik generiert werden soll. Wird ein gültiger Dateinamen hochgeladen, werden die notwendigen Schritte eingeleitet, um die entsprechende Statistik zu generieren.

Der Benutzer hat die Möglichkeit den Stil, der zu generierenden Statistik, im voraus zu definieren. Da im Moment nur zweidimensionale Plots erzeugt werden können, stehen auch nur die zweidimensionalen Stile zur Verfügung. Je nach Menge und Art der Input-Daten kann der eine-, oder andere Stil das beste Ergebnis erzielen.

Ab einer gewissen Datenmenge kann es passieren, dass in der Grafik nichts mehr zu erkennen ist. Dies ist noch ein offener Punkt, der nach jetzigem Stand, von Studenten, die an dem Projekt weiter arbeiten, bearbeitet werden soll.

3.3.2.4 Statistiken betrachten und vergleichen

Anforderung 7: Statistikenn betrachten und vergleichen

Dem Anwender soll es möglich sein, alle am Server vorhandenen Statistiken (innerhalb des spezifizierten „root“-Ordners) zu betrachten. Um Ergebnisse vergleichen und analysieren zu können, soll der Benutzer die Möglichkeit haben, mehrere Plots parallel zu betrachten.

Alle Ordner werden nach „.png“ Dateien durchsucht und deren Pfade werden in einer fortlaufenden Tabelle, mit zehn Einträgen pro Seite, angezeigt. Diese Pfade können via Drag&Drop in die Detailansicht gezogen werden. Dort wird das Bild zur Analyse angezeigt. In der Detailansicht hat der Benutzer die Möglichkeit die Spaltenanzahl einzustellen.

3.3.2.5 Unterschiedliche Rechte für Benutzergruppen

Grundsätzlich ist hier zwischen zwei unterschiedlichen Benutzern zu unterscheiden, deren Rechte in den folgenden Abschnitten beschrieben werden.

Anforderung 8: Der Admin-Benutzer

Der Admin-Benutzer hat alle Rechte in der Webapplikation. Dazu gehören alle zuvor erwähnten Punkte, die hier überblicksmäßig noch einmal aufgelistet werden sollen:

- Ordnerstruktur des Servers ansehen
- Neue Ordner anlegen
- Dateien zur Generierung der Statistiken hochladen
- Statistiken generieren
- Statistiken betrachten bzw. vergleichen

Anforderung 9: Der Standard-Benutzer

Der Standard-Benutzer hat nur die Möglichkeit Statistiken zu betrachten bzw. diese miteinander zu vergleichen. Ihm wird die Ordnerstruktur des Servers nicht angezeigt, da er weder Ordner anlegen, noch Statistiken generieren darf. Er bekommt nur die Tabelle mit allen Pfaden der „.png“-Dateien präsentiert, welche er, sowie der Admin via Drag&Drop in die Detailtabelle zur weiteren Analyse ziehen kann.

3.3.3 Gründe für die Verwendung von JSF und RichFaces

Aus Gründen der Usability, habe ich bei der Entwicklung sehr viel Wert auf die Komponenten und deren intuitive Anwendung gelegt. Die einfachste Lösung, um diese Anforderung zu erfüllen, ist die Verwendung eines „komponenten-basierten“ Web Application Frameworks (WAF). Die Verwendung eines solchen bietet laut [Shan und Hua, 2006, Seite 380] folgende Vorteile:

- Abstrahiert die Vorgänge des Request-Handling
- Wiederverwendbare Komponenten
- Abgrenzung von Logik und Präsentationsschicht
- In Verbindung mit den unterstützten „Event-Handling“-Mechanismen, ist ein komponenten-basiertes WAF, der Entwicklung einer Desktop GUI, sehr ähnlich

Zu den komponenten-basierten WAFs, gehören laut [Shan und Hua, 2006, Seite 384], folgende:

- JSF
- Tapestry
- Wicket
- etc.

Diese funktionieren im Prinzip sehr ähnlich. Der Hauptunterschied ist der Umfang, der zur Verfügung gestellten Komponenten. Um die Ordnerstruktur möglichst übersichtlich anzeigen zu können (siehe Anforderungen 1 und 2 auf Seite 27), habe ich mich dazu entschlossen, eine „Tree“ Komponente zu verwenden. Anforderung 6, der File-Upload, soll über eine entsprechende „File-Upload“-Komponente realisiert werden.

In **Tapestry** ist in der Komponentenbibliothek keine „Tree“-Komponente vorhanden. Es gibt zwar die Möglichkeit, eine Library mit „Tree“ einzubinden [4]. Diese ist aber kostenpflichtig und somit nicht für das Projekt geeignet. Des Weiteren gibt es die Möglichkeit eigene Komponenten in Tapestry zu erstellen [5]. Dieser Prozess ist allerdings mit viel Aufwand verbunden. Tapestry scheidet somit für das Projekt aus.

In **Wicket** gibt es sowohl eine „Tree“, als auch eine „FileUpload“ Komponente [6]. Die Anforderungen sind damit umsetzbar. Ich habe aber, auch nach intensiven Suchen im Internet, kaum Beispiel-Code dafür gefunden. Die Dokumentation der einzelnen Komponenten und wie diese zu verwenden sind, ist meiner Meinung nach etwas unübersichtlich. Wicket scheidet aus diesem Grund und aus Gründen der einfachen Erweiterbarkeit des Projekts aus.

Ganz anders sieht es dagegen bei **JSF** und **RichFaces** aus. Sowohl bei JSF, als auch bei RichFaces wird man schnell in bezug auf umfangreiche Dokumentation einzelner Komponenten und Beispiele, wie diese eingesetzt werden, fündig [7]. Ein weiterer Vorteil dieser beiden, kombiniert eingesetzten Technologien, ist die riesige Auswahl an bereits vordefinierten Komponenten. RichFaces bringt zusätzlich den Vorteil bereits vordefinierte „AJAX“-fähige Komponenten zu verwenden und jeder beliebigen Komponente „AJAX-Fähigkeit“ zu verleihen. In Bezug auf Unterstützung beim Entwicklungsprozess gibt es bei beiden Technologien ein umfangreiches Entwicklerforum. Dort findet man sowohl Hinweise über häufig gemachte Fehler und erhält rasch Unterstützung bei eventuell auftretenden Problemen oder Fragen.

Tabelle 3.2 zeigt eine Übersicht, der zuvor erwähnten WAFs, in Bezug auf die gestellten Anforderungen [8].

	JSF & RichFaces	**Tapestry**	**Wicket**
Komponentenbibliothek	*****	***	*****
Tree Komponente	Ja	Nein	Ja
FileUpload Komponente	Ja	Nein	Ja
Dokumentation und Beispiele	*****	**	***
Developer Support	*****	****	****
AJAX-Unterstützung	Ja	Ja	Ja
Ranking	1	3	2

Tabelle 3.2: Vergleich der komponentenbasierten Web Application Frameworks.

[4] http://tapestry.1045711.n5.nabble.com/Tree-Component-in-Tapestry-5-td2429744.html

[5] http://tapestry.apache.org/tapestry4/UsersGuide/components.html

[6] http://wicketstuff.org/wicket14/ajax/

[7] Hilfreiche Seiten für Entwickler sind zum Beispiel: http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/html_single/ oder http://livedemo.exadel.com/richfaces-demo/index.jsp

[8] Anm.: ***** = sehr gut, *** = durchschnittlich, * = nicht genügend

Kapitel 4

GnuPlot

4.1 Definition

Gnuplot ist ein kommando-orientiertes Progamm zur zwei-, oder dreidimensionalen Darstellung von Funktionen und Datenpunkten. Ursprünglich wurde es entwickelt, um Wissenschaftler und Studenten ein Werkzeug zur Visualisierung mathematischer Funktionen bereit zu stellen. Mittlerweile ist Gnuplot weiterentwickelt worden und durch die große Bandbreite an Einstellungsmöglichkeiten ist es für viele Aufgabenbereiche und Anforderungen verwendbar [Elsner, 2000, Seite 15].

Nach [Elsner, 2000, Seite 1] lassen sich folgende Eigenschaften von Gnuplot ableiten:

- Gnuplot ist portabel, d.h. einsetzbar in Linux, MSWindows, OS/2, OSX, VMS und anderen Systemen
- Der Source-Code ist geschützt, aber frei verwendbar
- Zweidimensionale Funktionen f(x) lassen sich in unterschiedlichen Stilarten (Punkte, Linien, Fehlerbalken) darstellen
- Dreidimensionale Funktionen g(x,y) lassen sich in den Stilarten (Kontur, Oberfläche, Hinzufügen von Gittern) darstellen
- Eine Vielzahl mitgelieferter mathematischer Funktionen wie sqrt(x), sin(x), log(x), etc.
- Verwendung von Benutzer-definierten Funktionen
- Einfaches Portieren von gnuplot-Kommandodateien unterschiedlicher Versionen auf Plattformen
- Exportieren von Abbildungen in eine Vielzahl von Grafikformaten (png, eps, jpg)
- Gestaltungselemente wie Titel, Achsenbeschriftungen, Achseneinteilung, Legende, etc. können hinzugefügt werden
- Bearbeiten der Kommandozeile und Zugreifen auf die Kommandohistorie

4.2 Funktionalität

Durch die Vielfalt an Einstellungsmöglichkeiten lassen sich auf unterschiedliche Art und Weise Graphen generieren und ausgeben. Um mit gnuplot arbeiten zu können, muss es zuerst auf dem Rechner installiert werden. Unter Linux funktioniert die Installation direkt über die Paketverwaltung, für Windows wird empfohlen, das Programm von der gnuplot Homepage `http://www.gnuplot.info` herunter zu laden.

Nach dem Aufruf der gnuplot.exe Datei ist es möglich, über ein Konsolenfenster mit gnuplot zu arbeiten. Die folgenden Unterkategorien geben eine Übersicht zur Arbeitsweise von gnuplot und informieren darüber was mit gnuplot möglich ist. Es wird gezeigt, wie Variablen und Funktionen definiert werden, wie bereits vordefinierte Funktionen eingebunden werden können und welche Einstellungsmöglichkeiten es für die graphische Darstellung gibt. Um den Einstieg mit gnuplot zu erleichtern, werden einige Beispiele von den zuvor genannten Themen gezeigt.

4.2.1 Operatoren

Dieser Abschnitt soll anhand der Tabellen 4.1, 4.3 und 4.2 eine Übersicht der Operatoren in gnuplot geben. Wie diese in der Praxis eingesetzt werden können, wird im Abschnitt 4.2.2 anhand einiger Beispiele beschrieben.
Die Operatoren wurden gemäß der Spezifikation der Programmiersprache C festgelegt [Gavin, 2008], d.h. Funktionalität, Umfang und Anwendung sind nahezu ident.

Nach [Crawford., 2010, Seite 27f] gibt es drei verschiedene Arten von Operatoren:

1. Einseitige Operatoren (siehe Tabelle 4.1)

Symbol	Beispiel	Beschreibung
-	-a	Einseitiges Minus
+	+a	Einseitiges Plus
~	~a	Einerkomplement
!	!a	logische Negation
!	a!	Fakultät

Tabelle 4.1: Einseitige Operatoren in gnuplot [SemiByte, 2010].

2. Dreiseitige Operatoren (siehe Tabelle 4.2)

Symbol	Beispiel	Beschreibung
?:	a?b:c	If then else

Tabelle 4.2: Dreiseitige Operatoren in gnuplot [SemiByte, 2010].

   Der ternäre Operator funktioniert wie eine „if then else“-Anweisung in diversen Programmiersprachen, wie etwa in Java. Ein Beispiel, wie ein dreiseitiger Operator verwendet werden kann, ist in Abschnitt 4.2.2 zu sehen.

3. Zweiseitige Operatoren (siehe Tabelle 4.3)

Symbol	Beispiel	Beschreibung
**	a**b	Potenzierung
*	a*b	Multiplikation
/	a/b	Division
%	a%b	*Modulo
+	a+b	Addition
-	a-b	Subtraktion
==	a==b	Gleichheits-Operator
!=	a!=b	Ungleichheits-Operator
<	a <b	Kleiner-Operator
<=	a <= b	Kleinergleich-Operator
>	a >b	Größer-Operator
>=	a >= b	Größergleich-Operator
&	a&b	*Bitweiser UND-Operator
^	a^b	*Bitweises Exklusiv-ODER
|	a |b	*Bitweises Inklusiv-ODER
&&	a && b	*Logisches UND
||	a ||b	*Logisches ODER
.	A.B	Konkatenation (Verkettung) von String A mit String B
eq	A eq B	String Gleichheit
neq	A neq B	String Ungleichheit

Tabelle 4.3: Zweiseitige Operatoren in gnuplot [SemiByte, 2010].

Die mit „*“ markierten Berschreibungen erwarten einen Integer-Wert, die mit Großbuchstaben markierten einen String [Crawford., 2010, Seite 27f]

4.2.2 Variablen und Funktionen definieren

In gnuplot gibt es die Möglichkeit, Variablen und Funktionen zu definieren und u.a. auch die in Abschnitt 4.2.1 genannten Operatoren dabei zu verwenden. Definiert werden Funktionen und Variablen mit einem Gleichheitszeichen und die Namen der definierten Variablen dürfen sowohl aus Buchstaben, als auch aus Zahlen bestehen. Eine Funktion unterscheidet sich nur durch die in Klammer angegebene Index-Variable. Zu beachten ist, dass gnuplot im Umgang mit Funktionen und Variablen case-sensitiv arbeitet, d.h. es ist auf Groß- bzw. Kleinschreibung zu achten [TUBraunschweig, 2008, Seite 2].

Das folgende Beispiel soll den Umgang mit Funktionen, Variablen und Operatoren veranschaulichen.

```
#Gnuplot example -mit "#" markierte Zeilen sind Kommentare
a=5
b=2
f(x)= x ** b + a*b                # x^2 + a*b
g(x) = x<b? x**2 : x<a? x : 1/x  # wenn x<1-->x^2,sonst wenn 1<=x<5-->x,sonst
print a==5                        # output 1
print !(a==5)                     # output 0
```

4.2.3 Vordefinierte Funktionen

In gnuplot gibt es einige vordefinierte Funktionen. Im Großen und Ganzen sind es die gleichen, wie in der Unix math library und umfassen einige grundlegende, zum größten Teil mathematische Funktionen [Crawford., 2010, Seite 24]. Laut [Crawford., 2010, Seite 25f] gehören dazu u.a. folgende:

- sin(x), cos(x), tan(x) → Sinus, Kosinus und Tangens
- asin(x), acos(x), atan(x) → Inverser - Sinus, Kosinus und Tangens
- ceil(x) → Aufrunden
- floor(x) → Abrunden
- sqrt(x) → Quadratwurzel
- exp(x) → Exponentialfunktion
- usw.

Diese Funktionen können, wenn notwendig, in eigene Funktionen eingebunden werden, wie folgendes Beispiel deutlich machen soll.

```
#Gnuplot example
f(x) = floor(pi * x) * floor(sqrt(x))
```

4.2.4 Wichtige Befehle für Layout-Einstellungen

Einige wichtige Befehle, die im Zuge dieses Projekts verwendet wurden, sollen in diesem Abschnitt beschrieben und anhand kurzer Beispiele veranschaulicht werden [1]. Am Ende dieses Abschnitts werden die Layouteinstellungen anhand eines gnuplot-Beispiels inklusive der generierten Output-Datei noch einmal veranschaulicht.

Im Bereich Layout ist der set-Befehl ein sehr wichtiger, weil damit eine Vielzahl von Einstellungen vorgenommen werden können. Detailliertere Informationen zu den Einstellungsmöglichkeiten sind in der Spezifikation unter [Crawford., 2010, Seite 95ff] zu finden.

4.2.4.1 Titel

Der Titel des Diagramms wird als Art Überschrift des Diagramms angezeigt (oben, Mitte) und kann über den Befehl „set title“ eingestellt werden. Dieser bietet auch Einstellungsmöglichkeiten von Schriftart, Farbe, etc [Crawford., 2010, Seite 146].

```
#Gnuplot example
set title "Titel des Diagramms" tc lt 3
plot sin(x)
```

[1] Anmerkung: Die jeweiligen Eigenschaften können auch mithilfe von Variablen gesetzt werden.

```
#-------------------------------------------------------------------------
set title "Titel" tc rgb "green"
plot sin(x)
#-------------------------------------------------------------------------
set title "Titel des Diagramms" font "Times-New-Roman,12" tc rgb "blue"
plot sin(x)
```

Die Abkürzung „tc“ steht für textcolour, und „lt“ für linetype. D.h. in der ersten Zeile wird die Farbe des dritten Linientyps als Textfarbe gesetzt. Im zweiten Fall wird die Farbe Grün als Schriftfarbe verwendet. Das dritte Beispiel zeigt, wie die Schriftart und Größe definiert wird.

4.2.4.2 Achsen

Achsenbeschriftungen: Für die Beschriftung der Achsen gibt es laut [Crawford., 2010, Seite 149] folgende Möglichkeiten:

- Die Achsen können über den jeweiligen set label-Befehl beschriftet werden.
  - „set xlabel“ → Beschriftung der x-Achse
  - Für die y- und z-Achsen gibt es die entsprechenden Befehle „set ylabel“ und „set zlabel“
  - Die Beschriftung der x2 (x-Achse oben) - und y2 (y-Achse rechts)-Achsen wird über den Befehl „set x2label“ bzw. „set y2label“ definiert
- Auch für die Achsenbeschriftungen können Schriftart und Farbe (siehe 4.2.4.1) eingestellt werden

Achsen-Tics: Mit dem Befehl „set xtics“ werden die Tics (Intervalle) auf der x-Achse festgelegt. Für die Achsen y, z, x2 und y2 gibt es auch hier die entsprechend abgeänderten Befehle [Crawford., 2010, Seite 152].

Es gibt per Definition zwei unterschiedliche Methoden, um die Tics festzulegen [Crawford., 2010, Seite 152f]:

- Implizit
  - set xtics <start><incr><end>
  - Wird z.B. der Befehl „set xtics 0,2,10“ eingegeben, bewirkt dieser das Setzen eines Tics bei jeder geraden Zahl zwischen 0 und 10
  - Wird kein Start und kein Ende angegeben, setzt gnuplot den Wert „unendlich“ ein. Daraus ergibt sich die Möglichkeit, nur die Schrittweite zwischen zwei Tics festzulegen, ohne Start und Ende zu spezifizieren; Z.b: „set xtics 5“ bewirkt einen Abstand von 5 zwischen 2 Tics
- Explizit
  - set xtics (<label><pos><level>) erlaubt Beschriftungen einzelner Tics und die Einstellung des levels (0 bedeutet Haupttic, 1 bedeutet Zwischentic)
  - set xtics („low“ 0, „medium“ 50, „high“ 100) führt also dazu, dass in diesem Diagramm anstatt der Zahl 0 „low“ steht

  - Wird kein level angegeben, wie in dem eben gezeigten Beispiel, wird das Standardlevel 0 (Haupttick) verwendet

Neben der Schrittweite gibt es im Bereich der Tics auch noch andere Parameter, die angegeben werden können. Laut [SemiByte, 2010] können u.a. die folgenden Eigenschaften gesetzt werden:

- Axis → Tics werden direkt auf der Achse gesetzt
- Border → Tics werden am Rand gesetzt
- Mirror → Setzt die Tics auch auf der gegenüberliegenden Achse
- mxtics, mytics, usw.→ Geben die „Untertics“ für die jeweilige Achse an

Achsenbereiche Mit dem Befehl „set range“ wird festgelegt, welcher Bereich der jeweiligen Achse im Diagramm angezeigt wird. Auch hier gibt es für jede Achse einen eigenen range Befehl („set xrange“, „set yrange“, „set zrange“, „set x2range“ und „set y2range“) [Crawford., 2010, Seite 150f].

```
#Gnuplot example
set xrange [5:45]
set yrange[*:]
```

Das zuvor gezeigte Beispiel würde bewirken, dass die zu zeichnende Funktion auf der x-Achse zwischen 5 und 45 angezeigt wird. Auf der y-Achse wird das Minimum durch Angabe von „*“ auf „autoscale“ gesetzt, während das Maximum unverändert bleibt [Crawford., 2010, Seite 151].

4.2.4.3 Beschriftung einzelner Datenpunkte

Die Beschriftung einzelner Datenunkte erfolgt über den Befehl „set label“. Voraussetzung dafür ist sowohl die Angabe des bezeichnenden Textes, als auch die der Koordinaten [Crawford., 2010, Seite 117].

```
#Gnuplot example
set label "Datenpunkt 1" at 5,20
```

Der Punkt mit den Koordinaten 5,20 wird mit „Datenpunkt 1“ beschriftet.

4.2.4.4 Legende

Der Befehl „set key“ bewirkt die Darstellung einer Legende zu den geplotteten Daten. Laut [Crawford., 2010, Seite 113 ff] können u.a. folgende Eigenschaften festgelegt werden:

- Position
- Schriftart

- Breite
- Höhe
- Textfarbe
- Rahmen
  - Linientyp
  - Linienfarbe
  - Linienbreite

```
#Gnuplot example
set key bottom left box lt 1
set key outside center right
```

Im ersten Beispiel wird die Legende links-unten mit einer rot umrandeten box angezeigt, während diese im zweiten Beispiel rechts-zentriert und außerhalb des Plots dargestellt wird.

4.2.4.5 Größe festlegen

Der Befehl „set size“ verkleinert die Grafik relativ zum Fenster. Die Größe ist in erster Linie abhängig vom verwendeten Terminal-Typ und wird dann prozentuell zu dieser verkleinert [Crawford., 2010, Seite 137f]

```
#Gnuplot example
set size 0.5,0,5
```

In diesem Beispiel wird also die grafische Darstellung um die Hälfte, abhängig vom Terminal, verkleinert.

4.2.4.6 Terminal

Gnuplot unterstützt eine Vielzahl von Grafik Formaten. Der Befehl „set terminal“ wird verwendet, um gnuplot mitzuteilen, welche Art von Output generiert werden soll [Crawford., 2010, Seite 143ff].

Im Zuge dieses Projekts werden die Terminals Png und Postscript verwendet, um einerseits png - und andererseits eps-Dateien zu erzeugen.

Wird sowohl „set terminal“, als auch „set output“ (siehe 4.2.4.7) verwendet, wie das bei diesem Projekt der Fall ist, wird von [Crawford., 2010, Seite 126] empfohlen, zuerst das Terminal zu spezifizieren, da bereits in dieser Phase Flags vom Betriebssystem für den späteren Output gesetzt werden.

Eine komplette Liste über die verfügbaren Terminals ist unter [Crawford., 2010, Seite 164ff] zu finden.

4.2.4.7 Output

Standardmäßig werden die Grafiken am Bildschirm angezeigt. Mit dem Befehl „set output“ kann der Output als Datei abgespeichert werden. Hier ist aber Vorsicht geboten, da nicht alle Terminals diese Funktion unterstützen, wie z.B. X11 oder wxt [Crawford., 2010, Seite 126]. Das folgende Beispiel zeigt, wie ein Plot als png-Datei abgespeichert werden kann.

```
#Gnuplot example
set terminal png
set output "C:/test.png"
# Plot Einstellungen setzen
plot sin(x)
```

4.2.4.8 Abschließendes Beispiel

Zum Abschluss dieses Bereiches werden alle in Abschnitt 4.2.4 erwähnten Befehle anhand eines Beispiels demonstriert. Der bis dato noch nicht erwähnte Befehl „plot“ dient zum Zeichnen und wird in Abschnitt 4.2.5.9 detailliert beschrieben. Das Ergebnis des folgenden gnuplot-Kommandos ist in Abbildung 4.1 ersichtlich.

```
#gnuplot example
set title "Der Titel der grafischen Darstellung"
set xlabel "xlabel"
set ylabel "ylabel"
set x2label "x2label"
set y2label "y2label"
set xtics nomirror -10,2,10
set ytics mirror 0-11,2,11
set xrange [-11:11]
set yrange[-12:12]
set label "Ursprung" at 0,0
set grid
set key outside bottom right box 1
plot sin(x), cos(x),tan(x)
```

4.2.5 Allgemeine, wichtige gnuplot Befehle

4.2.5.1 Set

Die „set“-Anweisung dient zum Setzen von Eigenschaften im Plot. Einige dieser Anweisungen wurden bereits im Abschnitt 4.2.4 detailliert beschrieben, wie z.b.: „set title“. Es können so lange Eigenschaften gesetzt werden, bis ein plot, splot oder replot-Aufruf ausgeführt wird [Crawford., 2010, Seite 90].

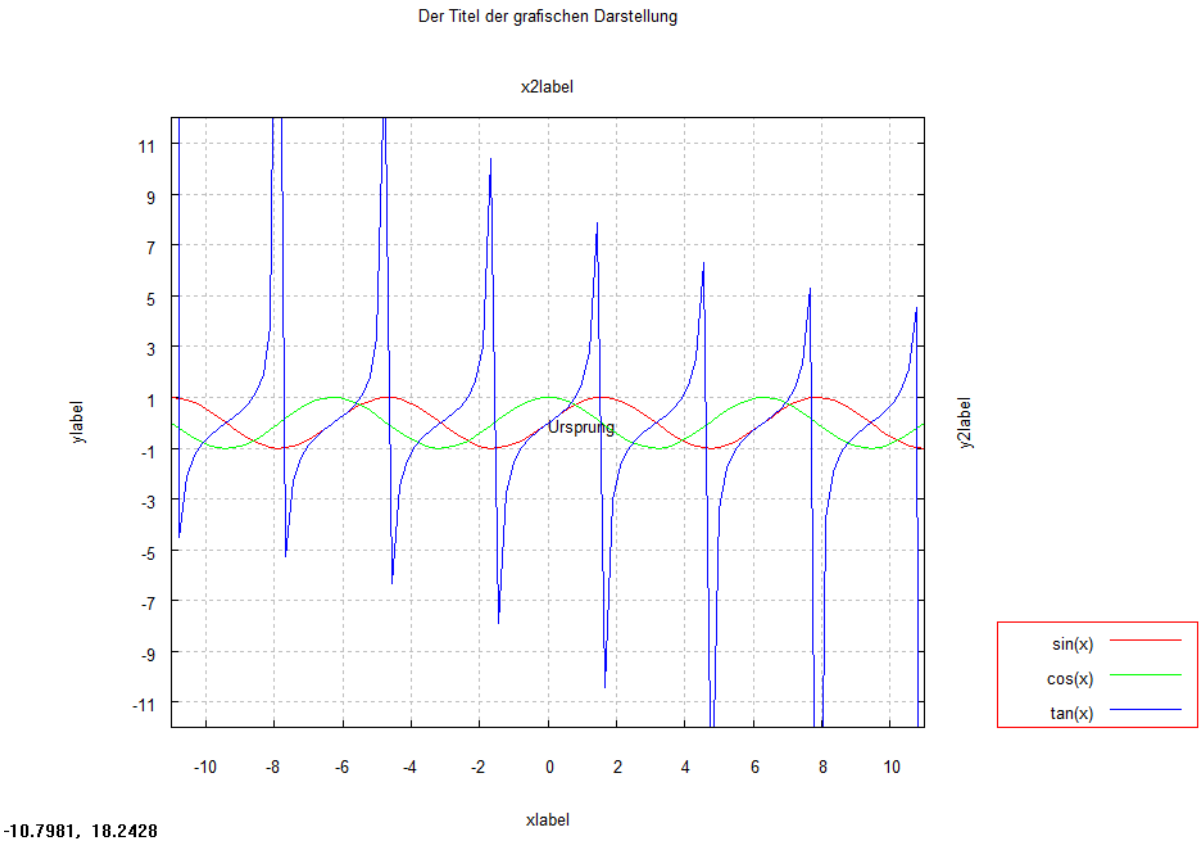

Abbildung 4.1: Output des zuvor gezeigten gnuplot-Beispiels.

4.2.5.2 Unset

Mit dem „unset“-Kommando werden sämtliche mit „set“ gesetzte Einstellungen wieder auf den Standardwert zurückgesetzt. Z.b.: unset xtics, welcher die ursprünglichen tics wiederherstellt [Crawford., 2010, Seite 163].

4.2.5.3 Reset

Im Gegensatz zu „unset“(nur für einen Befehl) setzt der „reset“-Befehl ALLE zuvor gesetzten Optionen auf ihre Standardwerte zurück [Crawford., 2010, Seite 89].

4.2.5.4 Cd - Change directory

Mit dem Befehl „cd“ kann das aktuelle Arbeitsverzeichnis gewechselt werden z.b.: cd ‚c:\folder1‘. Zu beachten ist laut [Crawford., 2010, Seite 56], dass Windows Benutzer ein einfaches Anführungszeichen verwenden sollten, weil backslash eine andere Bedeutung unter doppelten Anführungszeichen hat. Alternativ können unter doppelten Anführungszeichen zwei backslashes verwendet werden (cd „c:\\folder1“).

4.2.5.5 Exit, quit

Der „exit“- bzw. „quit“-Befehl beendet gnuplot [Crawford., 2010, Seite 58].

4.2.5.6 Print

Der „print“-Befehl dient zur Ausgabe von Zahlen oder Strings am Bildschirm und kann zum Debuggen verwendet werden [Crawford., 2010, Seite 87].

4.2.5.7 Pwd

„Pwd“ steht für print working directory und gibt das aktuelle Arbeitsverzeichnis am Bildschirm aus [Crawford., 2010, Seite 87].

4.2.5.8 Load

Mit dem „load“-Befehl können beliebige Textdateien geladen werden. Es muss allerdings gewährleistet sein, dass es sich beim Inhalt um gültige gnuplot-Anweisungen handelt. Gnuplot iteriert zeilenweise durch die Input-Datei und führt diese aus, d.h. das Einlesen einer Zeile ist äquivalent mit der Eingabe dieses Befehls in gnuplot durch den Benutzer [Crawford., 2010, Seite 66].

Der „load“-Befehl ist sehr wichtig für den praktischen Teil, weil bereits vor dem Zeichnen der Daten eine Input-Datei (je nach Statistik mit unterschiedlichen Einstellungen) generiert und diese dann mittels gnuplot ausgeführt wird. Somit muss nicht jeder Befehl einzeln von Java an gnuplot gesendet werden. Detaillierte Informationen zur Umsetzung können im Abschnitt 6.1 nachgelesen werden.

4.2.5.9 Plot, Splot, Replot

Um zu zeigen, was diese Befehle bewirken, werden diese zuerst kurz erklärt und mit einem Beispiel veranschaulicht. Anschließend wird der „plot“-Befehl detailliert betrachtet. Dadurch soll verständlich gemacht werden, welche Möglichkeiten der grafischen Darstellung möglich sind und auf welche Art und Weise der „plot“-Befehl eingesetzt werden kann. Dem Leser soll ein Grundverständnis vermittelt werden, um die Umsetzung der Statistikgenerierung im Projekt besser zu verstehen.

Plot-Allgemein: Die Befehle „plot“ bzw. „splot“ dienen zur grafischen Darstellung der zu zeichnenden Funktionen oder Daten. Der Unterschied der beiden Funktionen besteht darin, dass „plot“ die Daten zweidimensional darstellt, während „splot“ auf dreidimensionaler Ebene arbeitet. Der Befehl „replot“ wird verwendet um die spezifizierten Funktionen oder Daten erneut zu zeichnen [Crawford., 2010, Seite 36].

Dies kann verwendet werden, wenn sich einzelne Einstellungen im Vergleich zum davor gezeichneten Plot ändern [Crawford., 2010, Seite 88], wie folgendes Beispiel demonstrieren soll:

```
#Gnuplot example
```

```
set title "title"
set xlabel "x"
set ylabel "sin x"
plot sin(x)
set xtics 0-5,1,5
set xrange [-5:5]
replot
```

Bei replot kann eine Funktion angegeben werden, die erneut gezeichnet werden soll. Wird keine Funktion an replot übergeben, wird die zuletzt gezeichnete Funktion erneut dargestellt [Crawford., 2010, Seite 88].

Plot im Detail: Auf die verschiedenen Aufrufe des „plot"-Befehls möchte ich etwas genauer eingehen, weil es ein wichtiger Bestandteil im Projekt ist und ich kurz zeigen möchte, welche unterschiedlichen Möglichkeiten zur Darstellung denkbar wären.

Width: Funktionen oder Daten können auf unterschiedliche Weise dargestellt werden. Bevor also Daten angezeigt werden können, muss sich der gnuplot-Benutzer bewusst sein, auf welche Art und Weise diese veranschaulicht werden sollen. Der „with"-Befehl bietet die Möglichkeit, die Art des Diagramms für die darzustellenden Daten oder Funktionen festzulegen. Wird kein „with" mit entsprechendem Parametern angegeben, verwendet gnuplot die Standardeinstellung [Crawford., 2010, Seite 85].

An dieser Stelle seien nur einige der zahlreichen Diagrammarten erwähnt:

- Lines
- Dots
- Points
- Linespoints
- Impulses
- Errorbars
- Boxes
- Filledcurves
- Histograms
- etc.

Unterschiedliche Arten der Darstellung können verschiedene Einstellungsmöglichkeiten beinhalten, wie etwa bei „Lines" die Linienstärke und Farbe. Beim Stil „Boxes" kann z.B. die Füllfarbe angegeben werden. Detaillierte Informationen über die Einstellungsmöglichkeiten einzelner Diagrammarten, die benötigten Parameter (wie viele und welche Spalten müssen in der Inputdatei sein) für einen erfolgreichen plot und wie die einzelnen Arten aussehen, kann in der gnuplot Dokumentation nachgelesen werden [Crawford., 2010, Seite 40ff].

Plot: Der „plot“-Befehl unterstützt laut [Crawford., 2010, Seite 68] zwei grundlegend verschiedene Arten, um grafische Darstellungen zu erzeugen:

1. Plot einer Funktion

   (a) Dies kann eine bereits vordefinierte Funktion (siehe 4.2.3) sein. Möglich ist auch eine selbst definierte Funktion oder, wie das folgende Beispiel zeigt, eine Kombination aus beiden.

```
#Gnuplot example
f(x) = x**2 *sin(x)
plot f(x)
```

2. Plot von Daten

   (a) Datenpunkte werden auch über den „plot“-Befehl dargestellt, allerdings mit anderer Parameterreihenfolge. Als erster Parameter wird der Dateiname der Inputdatei unter Hochkomma angegeben, gefolgt von den Spaltennummern, die als x-, bzw. y-Werte interpretiert werden sollen [Crawford., 2010, Seite 3]. Der letzte Parameter legt noch den Typ der Darstellung, wie zuvor beschrieben, fest (Points, Lines, Impulses, etc.).

   (b) Eine Inputdatei (Bsp. „input.dat“) könnte wie folgt aussehen:

```
#x Häufigkeiten
 1 1
 2 4
 3 2
 4 2
 5 6
```

   (c) Dargestellt werden die Daten beispielsweise mit folgenden gnuplot-Befehlen:

```
#gnuplot example
Set xrange [0:6]
Set yrange [0:10]
plot "input.dat" with impulses
```

   (d) Das Ergebnis dieses Beispiels wird in Abbildung 4.2 dargestellt.

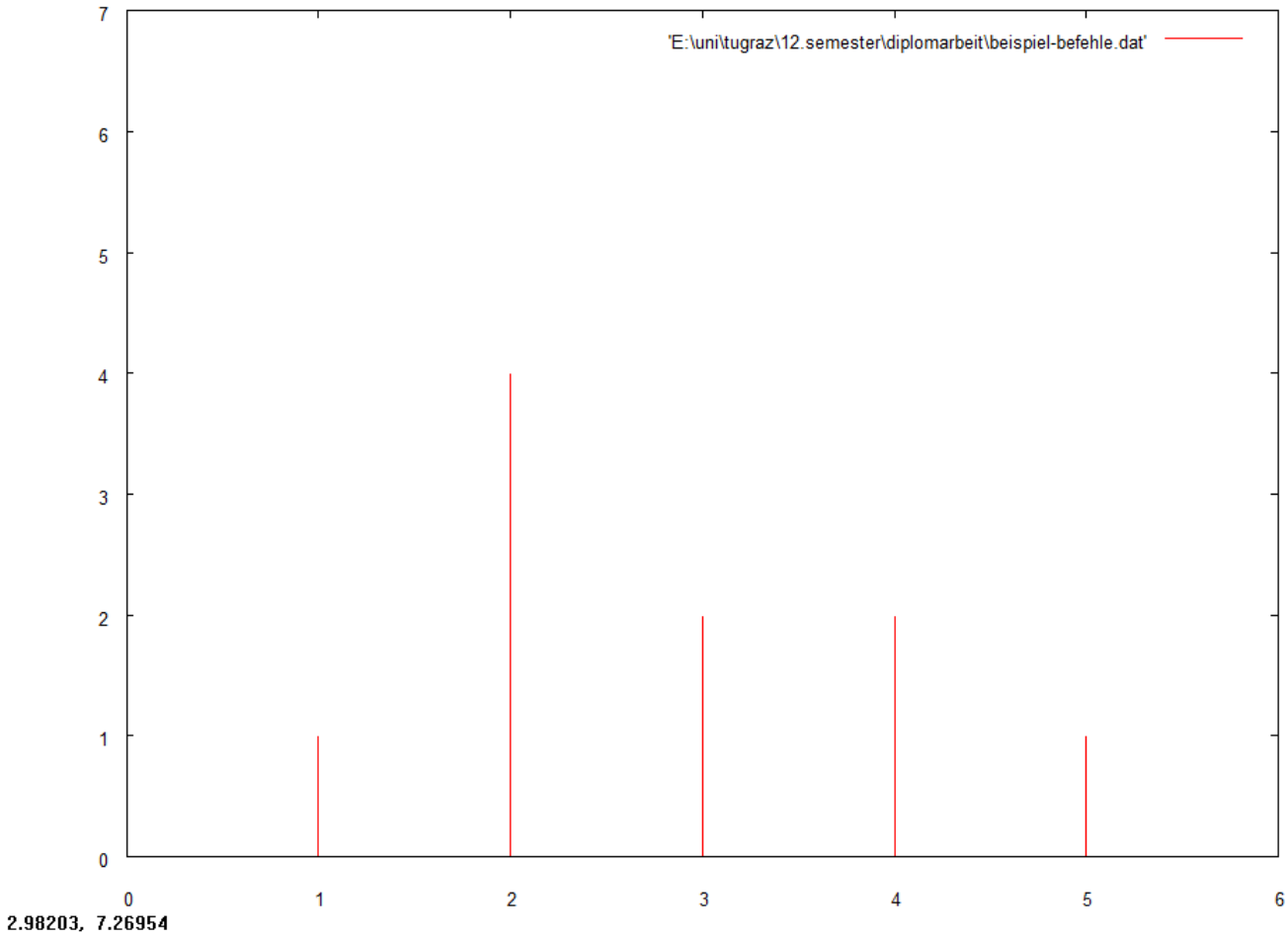

Abbildung 4.2: Output der Datei („input.dat“) mit den zuvor spezifizierten Einstellungen.

In vielen Fällen unterscheidet sich aber die Inputdatei von den erwarteten Parametern (Spaltenanzahl), da unter Umständen mehrere Plots aus einer Datenmenge generiert werden sollen. Um zu spezifizieren, welche Daten (Spalten) grafisch veranschaulicht werden sollen, ist der „using“-Befehl sehr hilfreich.

Using: Dieser Befehl wird verwendet, um gnuplot mitzuteilen, welche Spalten der Inputdatei für den plot-Befehl verwendet werden sollen [Crawford., 2010, Seite 79]. Je nach Art des Diagramms kann es eine unterschiedliche Anzahl von Spalten geben, die gnuplot für die Darstellung benötigt. Soll beispielsweise ein Liniendiagramm dargestellt werden, erwartet gnuplot eine x- und eine y-Koordinate im Datensatz. Wird kein „using“-Befehl angegeben, wird die Standardeinstellung verwendet, d.h. für die Koordinate x wird die 1. Spalte und für die Koordinate y die 2. Spalte verwendet.

Zu beachten ist, dass bei einer vollständigen Inputdatei, wie im vorigen Beispiel, die Befehle „plot data.dat with lines“, „plot data.dat using 1:2 with lines“ und „plot data.dat using (1) : (2) with lines“ die identen Ergebnisse liefern. Dies ist aber nicht der Fall, wenn die Daten unvollständig sind, wie etwa in folgendem Beispiel:

```
#x Häufigkeitet
1     1
2     4
3     2
```

```
4
5       1
6       6
```

Die drei Aufrufe liefern nun unterschiedliche Ergebnisse, wie die Abbildungen 4.3 bis 4.5 zeigen.

Der Grund dafür liegt in der unterschiedlichen Interpretation der Daten, wenn diese unvollständig sind: Im ersten Beispiel (siehe Abbildung 4.3) wird die fehlerhafte Zeile (Wert 4) als y-Wert der vorigen x-Koordinate (3) interpretiert [Crawford., 2010, Seite 80].

Mit der Option „using 1:2“ wird dieser fehlerhafte Datensatz ignoriert, d.h., dass der Datensatz davor mit dem danach verbunden wird [Crawford., 2010, Seite 80]. Somit ergibt sich in diesem Fall für den x-Wert 4 die Häufigkeit 1,5 (siehe Abbildung 4.4).

Die Verwendung von „using ($1):($2)“ bewirkt, dass der unvollständige Datensatz auf einen undefinierten Wert gesetzt wird [Crawford., 2010, Seite 80]. Deshalb kann dieser nicht angezeigt werden, wie in Abbildung 4.5 verdeutlicht wird.

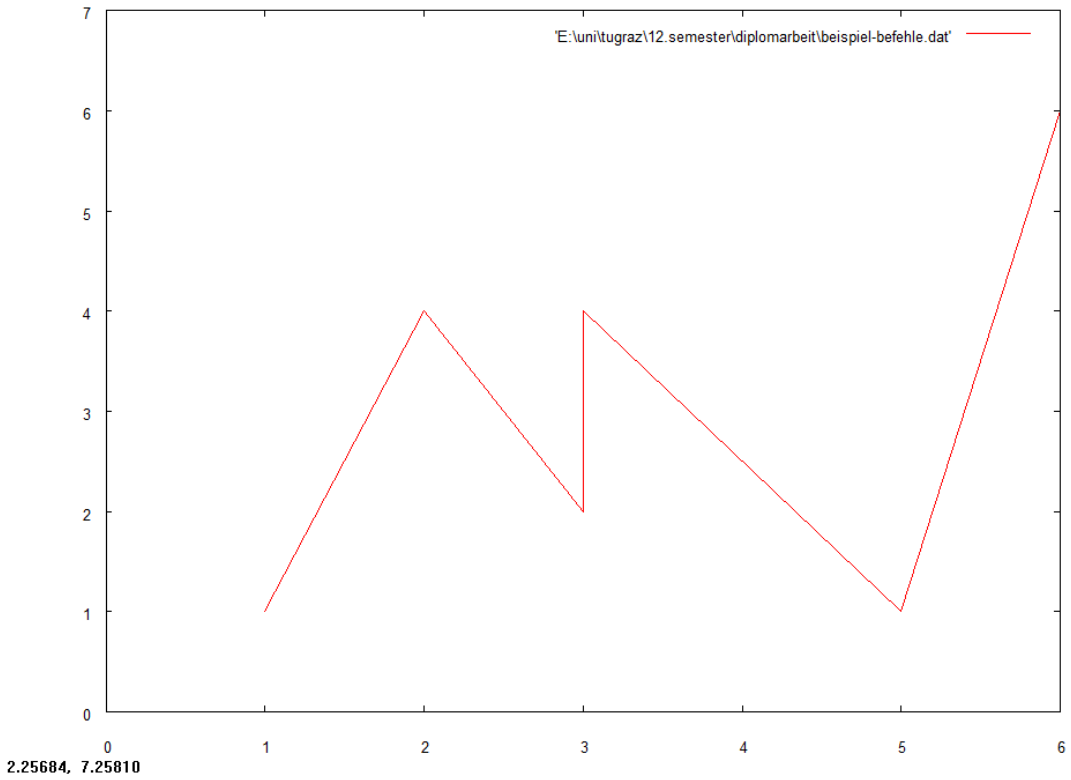

Abbildung 4.3: Output der Datei („data.dat“) mit „default“-Using.

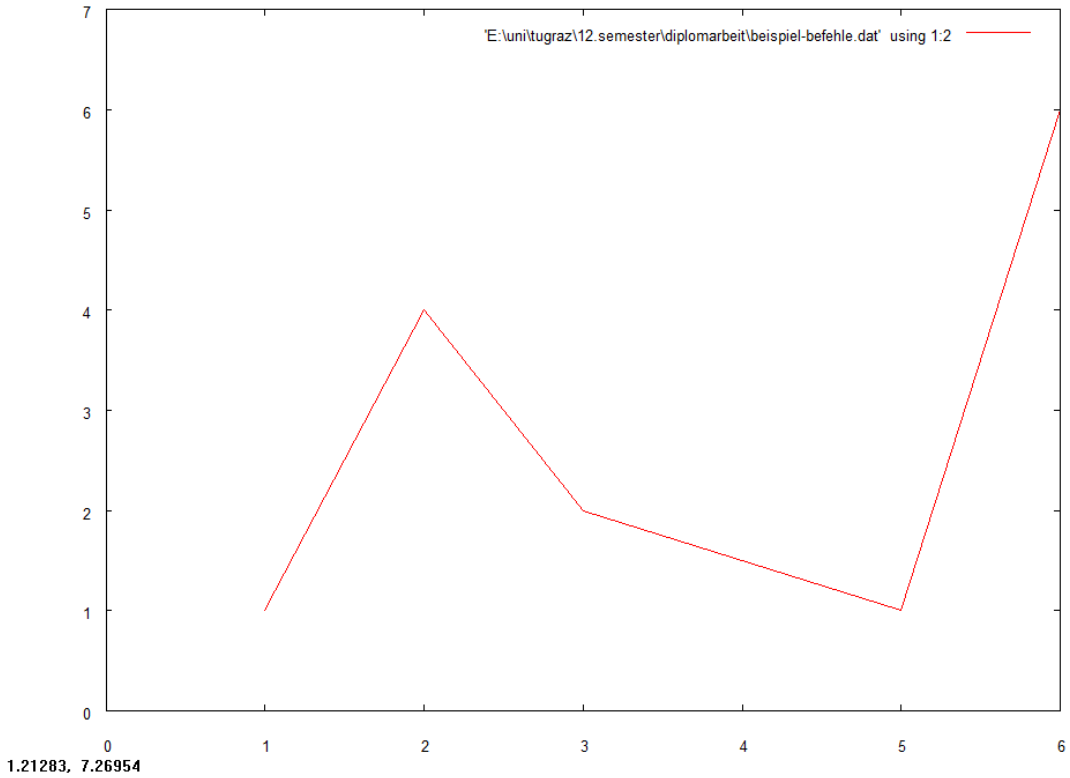

Abbildung 4.4: Output der Datei („data.dat“) mit Using 1:2.

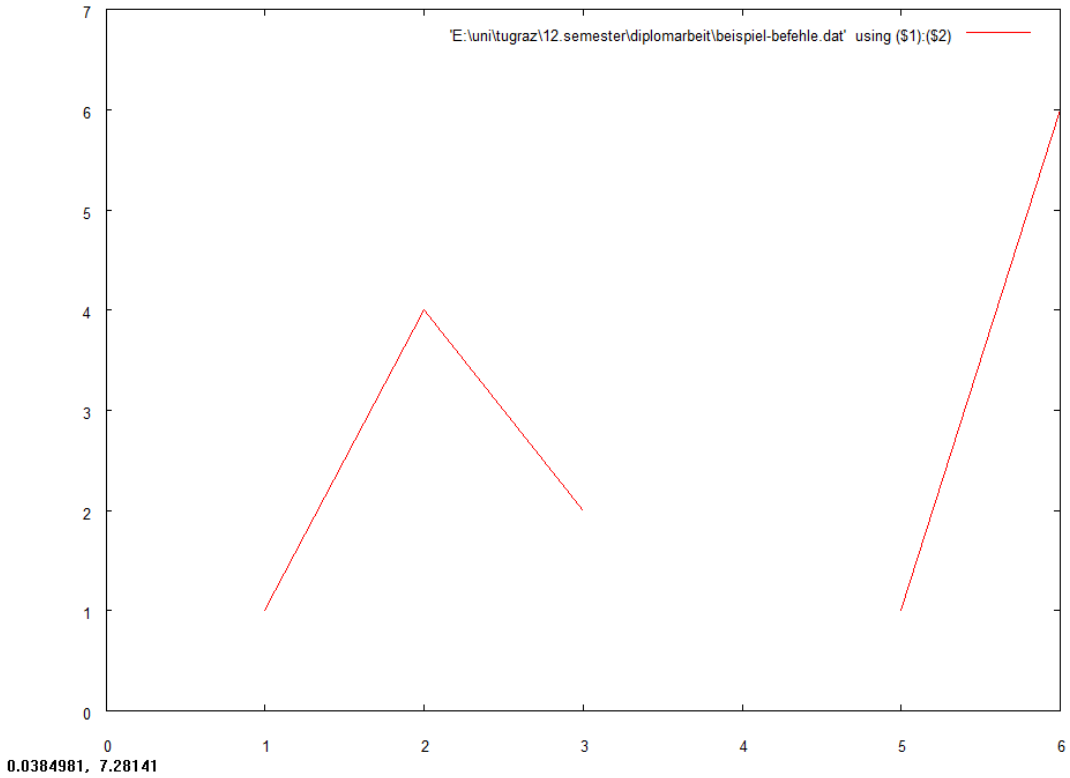

Abbildung 4.5: Output der Datei („data.dat“) mit Using $1:$2.

Kapitel 5

Webanwendung

Dieser Abschnitt beschreibt die beiden verwendeten Technologien, JavaServer Faces und RichFaces, im Detail. Zum Inhalt gehören u.a. Begriffsdefinitionen, Funktionsweise und technische Anforderungen. In Abschnitt 6.2 wird der praktische Teil der Webanwendung detaillierter betrachtet. Dieser beinhaltet u.a. die verwendeten Komponenten inklusive Beschreibung, Informationen zum Ablauf und einige Screenshots der Applikation.

5.1 JavaServer Faces

5.1.1 Was ist JSF

[Durocher, 2005] definiert JSF folgendermaßen:

> „Javaserver Faces sind ein standardisiertes serverseitiges Framework, welches die Entwicklung der Präsentationsschicht von Webanwendungen vereinfacht."

[Katz, 2008, Seite 1] definiert JSF etwas detaillierter:

JSF ist ein Java-Framework zur Erstellung Browser-basierender User Interfaces. Ein Hauptbestandteil sind die wiederverwendbaren UI (User Interface) Komponenten, bei deren Einbindung der Benutzer (Programmierer) nicht mit HTML-code in Berührung kommt [Katz, 2008, Seite 1]. Mehr Informationen zur Funktionsweise von JSF sind im Abschnitt 5.1.4 zu finden.

Um zu zeigen, dass JSF nicht etwas komplett Neues ist, möchte ich auf eine Definition von [Bosch, 2004, Seite 63]verweisen:

> „JSF definiert jetzt nicht eine komplett neuartige Technologie, sondern ist eher als konsequente Weiterentwicklung bestehender und bereits etablierter Technologien zu verstehen. Es mussten zuerst notwendige Basistechnologien vorhanden sein, mit denen ein höherwertiges Framework entwickelt werden konnte. Dabei ist das Wort höherwertig nicht dahingehend fehlzuinterpretieren, dass JSF etwas Besseres als Servlets oder Tag-Bibliotheken sei. Viel-

> mehr nutzt JSF die vorhandenen technischen Möglichkeiten, um auf dieser Basis ein umfangreiches und zum Teil auch komplexes Rahmenwerk bereitszustellen."

Abbildung 5.1 verdeutlicht noch einmal, dass JavaServer Faces nicht neu erfunden wurde, sondern bereits existierende Technologien vereinigt und auf einer höheren Ebene abstrahiert.

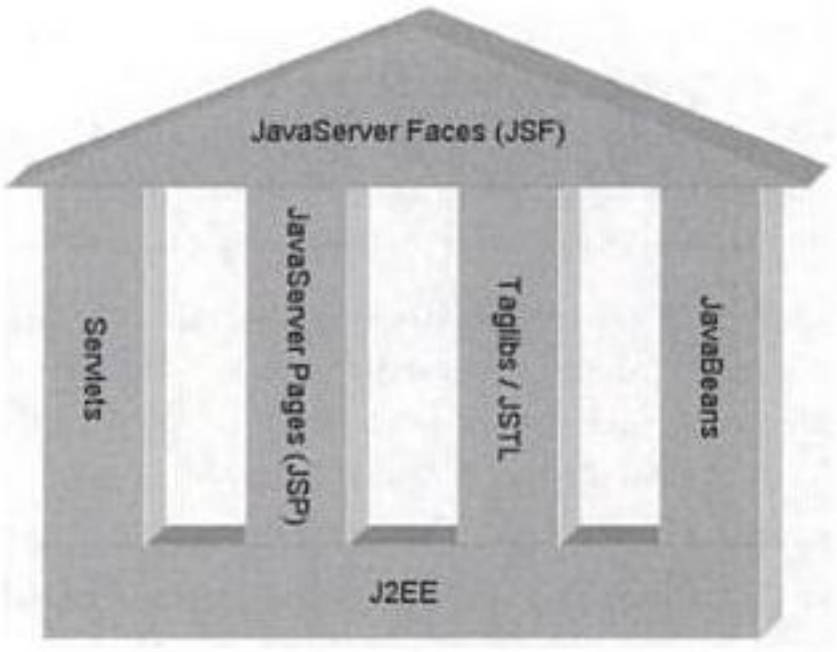

Abbildung 5.1: JSF Framework [Bosch, 2004, Seite 64].

[Durocher, 2005, Seite 2] liefert meiner Meinung nach die beste Definition zur Frage „Was ist JSF?". Er definiert den Begriff JSF wie folgt:

> „JSF sind ein standardisiertes serverseitiges Framework, welches die Entwicklung der Präsentationsschicht von Webanwendungen vereinfacht. Entwickler können wiederverwendbare UI-Komponenten zusammenstellen, um Webseiten zu erstellen, diese Komponenten mit der Datenquelle der Anwendung verknüpfen und clientseitige Ereignisse mit serverseitigen Event-Handlern verarbeiten."

Zusammenfassend lässt sich JSF als ein Framework, das die Entwicklung komplexer Java Web-Anwendungen durch Standardisierung erleichtern soll, beschreiben. Dies wird dadurch erreicht, indem dem Benutzer Standardaufgaben abgenommen werden. Laut [Katz, 2008, Seite 4] gehören zu diesen Standardaufgaben etwa:

- Daten aus einem Request auslesen
- Definieren von Navigation
- u.v.m.

5.1.2 Entwicklung von JSF

Dieser Abschnitt beschreibt die Entwicklungsstufen bis zur Entwicklung von JSF.

5.1.2.1 HTML

Zu Beginn der Web-Entwicklung 1991 gab es nur HTML (Hypertext Markup Language) als einfache Sprache zur Darstellung von Textteilen. Durch den großen Anwendungsbereich bekam auch das Layout immer mehr Bedeutung. Aufgrund der Komplexität der HTML-Dateien wurde mit CSS (Cascading Style Sheets) eine eigene Layoutsprache entwickelt. Begleitend dazu wurde auch die Einbindung dynamischer Aspekte immer wichtiger, was zur Entwicklung der Programmiersprache JavaScript führte [Marinschek et al., 2007, Seite 1].

5.1.2.2 Server

Neben den client-seitigen Fortschritten wurden auch am Server einige Scriptsprachen entwickelt, wie etwa:

- Perl
- Python
- PHP
- Java

Das Resultat der Vielfalt an Technologien war, dass die Erstellung großer, dynamischer Web-Anwendungen wichtiger und dadurch auch immer komplexer wurde [Marinschek et al., 2007, Seite 2].

5.1.2.3 Servlets

Im Java-Bereich war die Servlet-Technologie 1997 die erste Möglichkeit HTML Seiten dynamisch am Server zu generieren und diese an den Client zurück zu schreiben. Mit der wachsenden Komplexität der Anwendungen ist es leicht vorstellbar, dass der dabei zu generierende Code sehr schnell unübersichtlich wurde [Marinschek et al., 2007, Seite 2].

5.1.2.4 JSP - Java Server Pages

Ziel der JavaServer Pages war es in HTML sogenannte Scriptlets - Aufrufe der Java Methoden zur Ausgabe dynamischer Teile - in die HTML-Seite einzubinden. Was ursprünglich positiv gedacht war, hatte eine negative Entwicklung. Immer mehr Entwickler neigten dazu, noch mehr Code in die einzelnen Seiten einzubinden. Das Ergebnis war oft eine komplexe Mischung aus HTML und Java Code, die ähnlich schwierig zu warten war, wie die HTML-Generierung im zuvor erwähnten Servlet-Code. Ein zweiter Kritikpunkt war, dass viele Fehler erst zur Laufzeit auftraten, die bei der Erstellung von Java Klassen

bereits im Sourcecode beseitigt worden wären. Der Grund dafür ist, dass bei der Verwendung von Java Server Pages der Code erst beim Anwendungsstart im Applikationsserver kompiliert wurde. Je mehr Sourcecode in die JSP Seite eingebunden wird, desto negativer kann sich dieses Problem auswirken [Marinschek et al., 2007, Seite 4].

5.1.2.5 Web Frameworks

Um die bei JSP beschriebenen Probleme in den Griff zu bekommen, wurden Web-Frameworks entwickelt. Bei der Verwendung eines Frameworks wird der Benutzer wieder auf den ursprünglichen Gedanken des HTML zurückgeführt. Der Entwickler soll möglichst viele Teile der Layoutbeschreibung in einer Markup-Sprache - wie etwa JSP - erstellen und wenig Logik in dieser Sprache einfügen [Marinschek et al., 2007, Seite 4].

Aufgrund dieser Trennung findet hier das MVC (Model-View-Controller) Pattern Anwendung und speziell bei Web-Frameworks hat sich eine leicht abgeänderte Version des ursprünglichen MVC etabliert, das Model 2. Abbildung 5.2 soll dieses Pattern grafisch veranschaulichen.

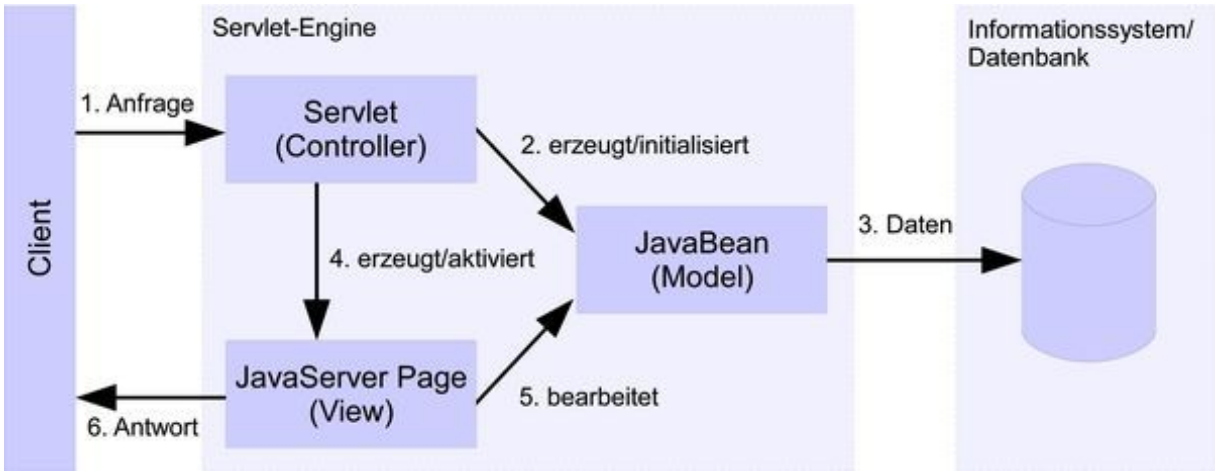

Abbildung 5.2: MVC Pattern - Model2 [Kurz et al., 2009].

Wie Abbildung 5.2 zeigt, dient das Servlet als Controller der Steuerungslogik. Dieser initialisiert das Modell, welches beispielsweise aus Java Klassen (Beans) bestehen kann. Für die Ansicht (View) können unterschiedliche Sprachen verwendet werden, wie etwa JSPs bei JSF oder etwa XML bei Cocoon [Marinschek et al., 2007, Seite 5].

Wie hier schon erkennbar wird, gibt es eine Vielzahl von Web-Frameworks. Um nur einige zu nennen seien an dieser Stelle die folgenden kurz erwähnt [Marinschek et al., 2007, Seite 5]:

- Cocoon
- Velocity
- Struts

Mittlerweile hat nicht allein die Trennung der Schichten Priorität bei den Web-Frameworks, sondern u.a auch die Wiederverwendbarkeit von Komponenten, die den Entwickler bei der Arbeit unterstützen soll [Marinschek et al., 2007, Seite 5].

5.1.2.6 JSF

JSF definiert einen Standard, der die vielfältigen und weit auseinander laufenden Ansätze zur Entwicklung von Web-Anwendungen in Java bündeln soll. Des Weiteren ist JSF auch eine Art Nachfolger des zuvor weit verbreiteten Struts-Framework [1], das versucht, die Vorteile von Struts zu übernehmen und dessen Schwächen auszubessern. Einer dieser Kritikpunkte ist die starke Verflechtung zwischen Struts und der Seitenbeschreibungssprache JSP. In dieser Hinsicht bleibt JSF offen und lässt beliebige Sprachen zur Beschreibung der Seiten zu [Marinschek et al., 2007, Seite 6].

5.1.3 Vorteile von JSF

Im diesem Abschnitt seien die positiven Eigenschaften von JavaServer Faces erwähnt, welche meine Entscheidung, das Projekt mit JSF umzusetzen, bestärkten. Nach [Anand, 2008, Seite 1] und [Burns et al., 2010, Seite 4] ergeben sich folgende Vorteile bei der Verwendung dieser Technologie:

- Ist nach einer kurzen Lernphase leicht zu erlernen
- Es gibt viele Libraries für Komponenten, die eingebunden und verwendet werden können
  - Diese nimmt dem Entwickler viel Arbeit im UI-Management ab
- Eine klare Trennung zwischen Inhalt und Präsentation
- Gibt Unterstützung in den Bereichen
  - Komponenten-Status verwalten
  - Komponenten-Daten verarbeiten
  - Benutzereingaben validieren
  - Event-Handling bei Interaktionen durch den Benutzer (Aufruf einer Methode am Server)
- Robuster Event-Handling Mechanismus
- Render-Kit unterstützt unterschiedliche Arten von Clients und deren Dateiformate
  - Für den Browser werden die Formate HTML, SGL[2], WML[3], XML[4] unterstützt
- Der Entwickler muss sich nicht um den Status der Komponenten bei Mehrfachzugriff von mehreren Clients kümmern; diese Aufgabe wird vollständig von JSF übernommen
- Client unabhängige Entwicklung von Web UIs
- Beschleunigung des Entwicklungsprozesses, indem der Benutzer sich auf Komponenten, Events, Backing-Beans und deren Interaktionen konzentrieren kann und nicht bezüglich Requests/Responses und Markup seine Entwicklungszeit aufwenden muss [Mann, 2004, Seite 9].

[1] http://struts.apache.org/
[2] http://www.sgl.sourceforge.net/
[3] http://www.w3schools.com/wap/default.asp
[4] http://www.de.selfhtml.org/xml/

5.1.4 Funktionsweise von JSF

Um die Funktionsweise von JSF detailliert betrachten zu können, ist es notwendig einige wichtige Begriffe zu definieren, die häufig in JSF vorkommen.

5.1.4.1 Allgemeine Begriffe in JSF

Komponente (UIComponent)

Eine Komponente (z.B.: „h:outputText“) sorgt dafür, dass die darzustellenden Werte aus dem Modell entnommen werden. Anschließend werden diese je nach Ausgabetechnologie in ein entsprechendes Format gebracht und an den Client zur Darstellung übertragen. Auch die umgekehrte Richtung muss von der Komponente unterstützt werden. Gibt der Benutzer einen Wert in ein Input-Feld ein, muss dieser in das Modell gespeichert werden [Marinschek et al., 2007, Seite 28].

Ein wichtiger Bestandteil der Komponente ist die Komponentenklasse. Diese beinhaltet die Eigenschaften der Komponente, die in der Seitenbeschreibungssprache (z.B. JSP) vom Entwickler gesetzt wurden [Marinschek et al., 2007, Seite 28].

Komponentenbaum (UITree)

Der Komponentenbaum beinhaltet alle Komponenten einer Seite und stellt diese hierarchisch dar. Alle Anfragen an die Anwendung werden an die Wurzel des Baums (UIRoot) gesendet, welche diese rekursiv an die Kinder weiterleitet. JSF arbeitet nie mit dem „HTML-Tree“, sondern immer mit dem Komponentenbaum. HTML-Code wird erst beim Rendern aus dem UITree erzeugt [Marinschek et al., 2007, Seite 28].

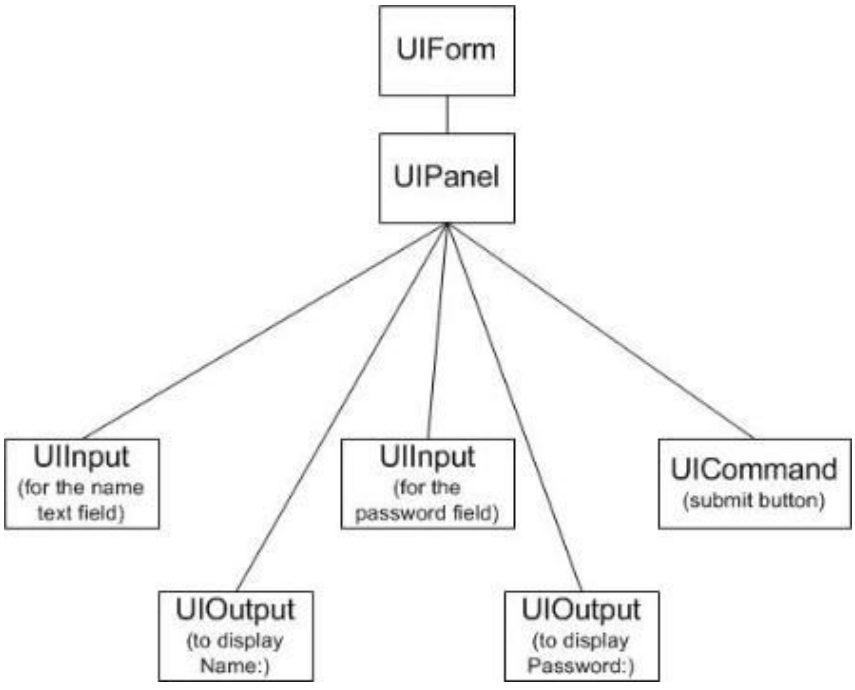

Abbildung 5.3: Beispiel für einen Komponentenbaum [Intertech, 2008].

Renderer

Die Darstellung der Komponente inklusive ihrer Daten am Client wird vom Renderer übernommen. Es gibt die Möglichkeit, mehr als einen Renderer für eine Komponente zu spezifizieren. Ist dies der Fall, wird je nach Ausgabetechnologie der dafür zuständige ausgewählt. Wird keiner angegeben, ist - wie zuvor im Absatz „Komponente“ beschrieben - diese selbst für die Darstellung verantwortlich [Marinschek et al., 2007, Seite 28].

Validator

Nicht selten kommt es vor, dass bei einer Internet-Anwendung ungültige Werte vom Benutzer in ein Feld eingegeben werden. Um dies zu verhindern, gibt es die Validatoren. Diese können vom Entwickler selbst geschrieben werden, es gibt aber auch bereits vordefinierte (Bsp.: <f:validateLength>). Ungültige Werte werden nicht ins Modell zurückgeschrieben [Marinschek et al., 2007, Seite 29].

Ein Beispiel für einen eigenen Validator wäre, zu überprüfen, ob in einem Input-Feld zur Eingabe der e-mail Adresse ein „@“ Zeichen enthalten ist. Detaillierte Informationen zur Erstellung eigener Validatoren sind unter [Patel, 2009] zu finden.

Converter

Browser können nur Strings verarbeiten. Aus diesem Grund werden Converter eingesetzt, um die Datentypen der Anwendungslogik in eine Zeichenkette zu konvertieren. Auch hier muss wieder in zwei Richtungen gearbeitet werden. Einerseits vom Java-Datentyp in eine Zeichenkette, wenn Daten gelesen und im Browser angezeigt werden sollen. Andererseits muss eine Benutzereingabe, welche am Client getätigt wurde, in den zum Modell kompatiblen Java Datentyp konvertiert werden. Dies ist notwendig um diese Daten auch im Modell speichern zu können [Marinschek et al., 2007, Seite 29].

Es gibt einige bereits vordefinierte Converter in JSF (z.B.: <f:convertDateTime>), die häufig verwendet werden. Laut [Marinschek et al., 2007, Seite 89] gibt es die Möglichkeit, eingene Converter für beliebige Datentypen zu schreiben (siehe Abschnitt 5.1.4.2, Phase 3).

Managed-Beans

Als Managed-Beans werden die Klassen im Modell bezeichnet, welche die Werte für die Komponenten speichern sollen. Anforderungen an eine Managed-Bean ist ein Public-Konstruktor und Private-Variablen mit dazugehörigen getter - bzw. setter Methoden [Marinschek et al., 2007, Seite 29].

Ein wichtiger Faktor bei den Managed-Beans ist der Gültigkeitsbereich (Scope), welcher auf folgende Werte gesetzt werden darf und nach [Lim, 2007] folgende Eigenschaften besitzt:

- None
  - Objekte mit Scope „None“ sind auf keiner JSF-Seite sichtbar. Findet sich in der Konfigurationsdatei (siehe Abschnitt 5.1.4.3) eine Managed-Bean mit Gültigkeit „None“, ist das ein

Indikator dafür, dass dieses Objekt von einer anderen Klasse in der Applikation verwendet wird.

- Request
  - Objekte mit „Request-Scope“ sind von Beginn eines Requests bis zu dessen Ende gültig, wobei das Ende mit dem Senden der Antwort (Response) zum Client gleichzusetzen ist. Wird ein Request weitergeleitet, ist das Objekt auch auf dieser weitergeleiteten Seite sichtbar. Managed-Beans mit Request-Scope können andere Objekte mit Sichtbarkeit „none“, „request“, „session“ oder „application“ beinhalten.
- Session
  - „Session-Scope“-Objekte sind in allen Request/Response-Zyklen innerhalb einer Session sichtbar und solange gültig, bis das Objekt oder die Session ungültig bzw. zerstört wird. Managed-Beans mit Session-Scope können andere Objekte mit Sichtbarkeit „none“, „session“ oder „application“ beinhalten.
- Application
  - „Application-Scope“-Objekte sind in allen Request/Response-Zyklen für alle Clients in der Applikation sichtbar und gültig, bis die Applikation nicht mehr aktiv ist. Managed-Beans mit Application-Scope können andere Objekte mit Sichtbarkeit „none“ oder „application“ beinhalten.

Wie die Definition der Sichtbarkeit von Managed-Beans in der Konfigurationsdatei „faces-config.xml“ funktioniert, soll das Beispiel in Abschnitt 5.1.4.3 zeigen.

Ereignisbehandlung

Ein Ereignis in JSF tritt dann auf, wenn z.B. eine Schaltfläche betätigt, eine Checkbox angehakt, oder generell ein Wert verändert wird. Dies sind die wichtigsten und zentralsten Elemente in JavaServer Faces. Wird beispielsweise eine Schaltfläche betätigt, wird eine dazu spezifizierte Methode am Server aufgerufen. Prinzipiell kann jede Komponente Ereignisse auslösen und jede Managed-Bean kann als Listener (Interessent) eines Ereignisses registriert werden [Marinschek et al., 2007, Seite 29].

Navigation und Aktionen

Navigiert wird in JSF über Navigationsregeln (navigation-rules), welche in der Konfigurationsdatei „faces-config.xml“ definiert werden müssen. Ausschlaggebend dafür, wohin navigiert wird, ist der Rückgabewert der Action-Methode. Navigationsregeln können global, d.h. von allen Seiten aus, oder nur für einzelne Seiten definiert werden [Marinschek et al., 2007, Seite 30]. Das Beispiel in Abschnitt 5.1.4.3 soll die Funktionsweise der Navigation in JSF verdeutlichen.

Tag Library

In JSP können Befehle über tags zugänglich gemacht werden, um den Anteil von Java-Code in der JSP-Seite zu minimieren. Dabei kann es sich um vordefinierte tags oder um selbst erzeugte tags handeln. Eine Tag-Library ist eine Ansammlung von „eigenen“ tags [Microsystems, 2000], („What is a Tag Library“).

Unified Expression Language

Die Unified Expression Language wurde entwickelt, um dem Entwickler die Möglichkeit zu geben, das User-Interface mit den dahinter liegenden Daten aus dem Modell dynamisch zu verbinden. Begrenzt wird der UEL Ausdruck vorne von einer Raute und einer öffnenden geschwungenen Klammer und am Ende von einer schließenden geschwungenen Klammer. Innerhalb dieser Begrenzung kann auf Objekte und deren Werte oder Methoden zugegriffen werden. Daneben sind auch Operatoren in UEL erlaubt, wie etwa Vergleichsoperatoren (==,!=), arithmetische Operatoren (+,-) und sogar ternäre Operatoren (bedingung ? dann : sonst) [Marinschek et al., 2007, Seite 40f].

Wie der Einsatz der UEL in der Praxis funktioniert, soll anhand einiger Beispiele verdeutlicht werden:

```
<h:outputText id="output" value="#{user.name}"
   rendered="#{user.name != null}"/>
```

In diesem Beispiel wird der Name des Benutzers aus der Bean „user“ über die Methode „getName“ ausgelesen und angezeigt. Dargestellt wird das Textfeld allerdings nur, wenn auch wirklich ein Name vorhanden ist. Zu beachten ist, dass die Bean unbedingt in der „faces-config.xml“ definiert sein muss. Wie die Definition der Beans funktioniert, wird im Beispiel in Abschnitt 5.1.4.3 gezeigt.

```
<h:outputText value ="#{order.costs * 1.20}"/>

<h:outputText value="#{order.costs >= 1000 ?
     order.costs * 0.95 : order.costs}/>
```

Hat der Kunde über einen Wert von 1000 bestellt, wird ein Rabatt von 5 % gewährt.

```
<h:commandButton action="#{order.save}"/>
```

Die Signatur der „action“-Methode ist wie folgt:

- String save() - der String als Rückgabewert dient auch zur Navigation

```
<h:commandButton actionListener="#{order.deleteProduct}"/>
```

Der Unterschied zur Action ist, dass beim ActionListener ein ActionEvent als Parameter mitgegeben wird. Dadurch ist es z.B. möglich, einen aus einer Liste ausgewählten Datensatz zu löschen oder den ausgewählten Knoten in einem Baum zu identifizieren.

```
<h:outputText value="#{order.products[4]}"/>
```

Hier wird auf den fünften Wert der Liste „products“ in der Bean „order“ referenziert.

In diesem Abschnitt wurden einige essentielle Begriffe für die Verwendung von JSF definiert und anhand einiger Beispiele veranschaulicht. Abschnitt 5.1.4.2 und 5.1.4.3 beschreiben noch zwei weitere Aspekte, welche für den Einsatz von JSF unbedingt gelesen und verstanden werden sollten.

5.1.4.2 Lebenslauf einer HTTP-Anfrage in JSF

Bei manchem Leser ist vielleicht die Frage aufgetreten, wann die zuvor beschriebenen Converter, Validator, etc. eigentlich zum Einsatz kommen. Um diese Frage zu beantworten, ist es notwendig, einen Blick auf den Lebenslauf einer HTTP-Anfrage in JSF zu werfen. Dieser beschreibt die genaue Reihenfolge der durchzulaufenden Schritte, vom Senden der Anfrage (Request) bis zum Erhalt der Antwort (Response). Abbildung 5.4 soll eine grafische Unterstützung zum Verständnis des Ablaufs bieten [5]. Im Anschluss werden die einzelnen Phasen detaillierter betrachtet.

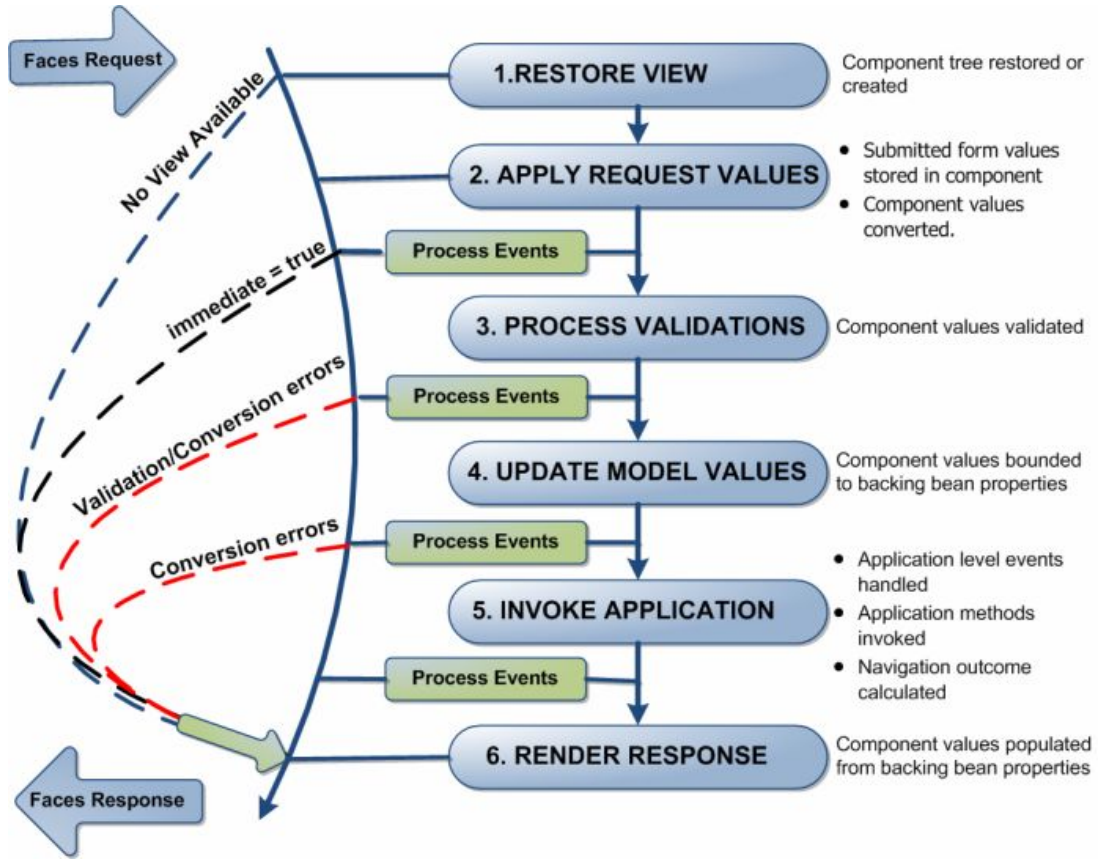

Abbildung 5.4: Ablauf einer HTTP-Anfrage in JSF [Anand, 2008].

Phase1: Restore View

In der Restore View-Phase sind JSF Klassen dafür verantwortlich, den Komponentenbaum für die darzustellende Seite (= eingehender Request) aufzubauen. Im Hintergrund versucht der JSF Framework-Controller über die ID der Ansicht (meist der Name der Seite), die Komponenten für diese zu laden. Sind keine vorhanden (initial request), erzeugt der JSF Controller einen neuen Komponentenbaum, sonst wird der bereits existierende Baum geladen (postback) [Anand, 2008]. Für den Durchlauf der Phasen bedeuten diese zwei Typen von Anfragen unterschiedliche Pfade:

Handelt es sich bei einer Anfrage um einen Initial-Request werden nur die Phasen Restore View und Render Response ausgeführt, weil es keine Input-Daten oder auszuführende Aktionen geben kann. Ist die Anfrage allerdings ein Postback-Request, wird in der Restore View-Phase die Ansicht über die gespeicherten Statusinformationen geladen und der Durchlauf aller Phasen wird eingeleitet [Anand, 2008].

[5] Anm.: Der Standard-Ablauf wird in „durchgehenden“ Linien dargestellt. Die „gestrichelten“ Linien sind Alternativpfade, die bei diversen Fehlern (siehe Abbildung 5.4) durchlaufen werden.

Phase2: Apply Request Values

In der Apply Request Values-Phase wird das sogenannte Decoding durchgeführt. Dabei werden die Request-Parameter ausgelesen und die Werte an die entsprechenden UI-Komponenten übergeben. Tritt ein Fehler auf, wird dieser in den FacesContext geschrieben und bei der Render Response-Phase ausgegeben [Anand, 2008].

Ist bei einer Komponente der aktuellen Seite die Eigenschaft „immediate“ auf „true“ gesetzt , werden für diese die Schritte Validierung, Konvertierung und Ereignisbehandlung bereits in dieser Phase ausgeführt und nicht wie üblich in der Process Validation-Phase. Dies kann beispielweise bei einem „Abbrechen“ Button verwendet werden, da dieser keine Werte des Modells benötigt [Anand, 2008].

Phase3: Process Validations

Die vom Benutzer eingegebenen Werte, welche in Phase zwei an die Komponenten übergeben wurden, werden in dieser Phase konvertiert und validiert. Wie bereits in Abschnitt 5.1.4.1 bei der Definition des Begriffs „Konverter“ erwähnt wurde, arbeitet der Browser grundsätzlich immer mit Zeichenketten. Damit eingegebene Daten in Phase vier erfolgreich ins Modell geschrieben werden können, müssen diese in die entsprechenden Java-Datentypen konvertiert werden. In vielen Fällen ist der von JSF bereits vordefinierte Standardkonverter ausreichend. Ist dies nicht der Fall, gibt es die Möglichkeit, eigene Konverter zu implementieren. Dabei gilt es, die folgende Vorgehensweise einzuhalten:

- Die benutzerdefinierte Konverter-Klasse muss von javax.faces.convert.Converter abgeleitet sein
- Die Methoden getAsObject() und getAsString() müssen überschrieben werden

Zur Verwendung des selbst definierten Konverters, kann das Attribut „converter“ für die jeweilige UI-Komponente gesetzt werden [Marinschek et al., 2007, Seite 35].

Wurde der Wert erfolgreich konvertiert, kommt der Validator zum Einsatz. Auch hier kann es sich um einen bereits vordefinierten oder um einen selbst implementierten handeln. Verwendet wird ein Validator mittels Hinzufügen des Kindelements „validator“ zur entsprechenden Komponente. Wurde der Wert erfolgreich konvertiert und validiert, wird dieser in der value-Eigenschaft der Komponente gespeichert und in Phase vier ins Modell geschrieben. Tritt bei der Konvertierung oder Validierung ein Fehler auf, werden die entsprechenden Fehlermeldungen generiert. Da es bei einer fehlerhaften Verarbeitung der Daten zu keiner Speicherung im Modell kommen darf, werden die Phasen zur Modell-Aktualisierung und Ausführung der Applikationslogik übersprungen und die Antwort wird sofort angezeigt [Marinschek et al., 2007, Seite 35f].

Phase 4: Update Model Values

Wie schon im Abschnitt 5.1.4.1 erwähnt, gibt es in JSF die Möglichkeit, Werte von Komponenten über die Unified Expression Language mit dem Modell zu verknüpfen. Abhängig von diesen definierten UEL-Ausdrücken, werden in dieser Phase die Daten vom Komponentenbaum ins Modell gespeichert. Deshalb kommen auch hier die Konverter ins Spiel, um zwischen den String-Repräsentationen der Komponentendaten in die zum Modell passenden Java-Datentypen konvertieren zu können. Tritt bei der Konvertierung

ein Fehler auf, wird sofort die Render Response-Phase ausgeführt. Erneutes Laden der Seite zeigt diese Fehlermeldungen am Browser an [Anand, 2008].
Der Unterschied zu Phase zwei ist der, dass die Werte von den Komponeneten ins Modell geschrieben werden, während in Phase zwei die Werte vom Request in die UI-Komponenten geschrieben werden [Anand, 2008].

Phase 5: Invoke Application

In dieser Phase werden alle Events der Applikations-Ebene behandelt. Neben den ausgelösten Events sind auch die aufzurufenden Methoden von großer Bedeutung, wie etwa die ActionListener einer Schaltfläche. Diese wird nach dem Auslösen des ActionEvents durch den Klick auf einen Button aufgerufen. Wichtig wird in dieser Phase auch das Thema Navigation, weil der „Navigation outcome“ berechnet wird. Dies passiert im Falle einer Schaltfäche, durch das Aufrufen der Action Methode, welche einen String („Navigation outcome“) zurückliefert. Entsprechend den definierten Navigationsregeln in der „faces-config.xml“ wird der Benutzer auf die jeweilige Seite weitergeleitet [Anand, 2008].

Phase 6: Render Response

Ziel dieser Phase ist es, die Antwort des Servers im Browser darzustellen. Dabei müssen zuerst die Werte inklusive derer vorgenommener Modifikationen aus dem Modell an die Komponenten übertragen werden. Neben der Präsentation der Daten speichern die UI-Komponenten auch deren Status, welcher standardmäßig als verstecktes Eingabefeld in den HTML-Code geschrieben wird [Anand, 2008].

Treten bei einem Postback request in irgendeiner Phase Fehler auf, wird die ursprüngliche Seite angezeigt. Durch das Einbinden des <h:message>tags werden alle generierten Fehlermeldungen auch dem Benutzer angezeigt [Anand, 2008].

Process Events

In dieser Phase werden alle in der vorhergehenden Phase aufgetretenen Events (z.B. Validierungs-Fehler) behandelt, um der Applikation einen „sicheren“ Ausstieg zu ermöglichen [Anand, 2008].

5.1.4.3 Konfigurationsdateien

In JSF gibt es die Möglichkeit, zentrale Konfigurationseinstellungen in zwei unterschiedlichen Dateiein festzulegen. Zu diesen Einstellungen gehören laut [Marinschek et al., 2007, Seite 49f]:

- Konfiguration von Managed-Beans (faces-config.xml)
- Navigationsregeln (faces-config.xml)
- Internationalisierung der Applikation (faces-config.xml)
- Allgemeine Einstellungen zur Applikation (web.xml)

Web.xml

Wie bei jeder Java Web-Applikation dient die Konfigurationsdatei web.xml zur Spezifikation der zentralen Einstellungen, die prinzipiell in zwei Arten unterteilt werden können und folgende Eigenschaften besitzen [Marinschek et al., 2007, Seite 50ff]:

- Performance-relevante Parameter
  - Speichern des Status der Applikation
    - Beim ersten Aufruf einer Seite wird der Komponentenbaum erzeugt. Der Status dieses Baums wird gespeichert, um bei erneutem Zugriff auf die Seite diese schneller laden zu können (es muss nicht bei jedem Aufruf der Komponentenbaum neu erzeugt werden). Über den Kontextparameter STATE_SAVING_METHOD kann spezifiziert werden ob der Status am Server oder am Client gespeichert werden soll. Besteht kein zwingender Grund, den Client zu verwenden, wird aus Performance-Gründen empfohlen, auf den Server als Speichermedium zurückzugreifen.
  - Speichern der letzten Komponentenbäume in der Session
    - Gibt die Anzahl der zu speichernden Komponentenbäume an. Dieser Parameter funktioniert allerdings nur, wenn als STATE_SAVING_METHOD der Wert „Server“ gesetzt wurde. Zu beachten ist, dass ein bereits gespeicherter UITree zu schnelleren Ladezeiten der Seite führt. Dadurch wird allerdings auch mehr Speicher am Server benötigt.
  - u.v.m.

  Für detaillierte Informationen zur Performance-Steigerung bietet [Myfaces Wiki, 2009] eine umfangreiche Übersicht.

- Nicht Performance-relevante Parameter
  - Konfigurationsdateien
    - Über den Kontextparameter CONFIG_FILES können die Konfigurationsdateien angegeben werden. Wird kein Parameter angegeben, wird standardmäßig die faces-config.xml verwendet [Marinschek et al., 2007, Seite 51].
  - Filter
    - Es gibt eine Vielzahl bereits vordefinierter Filter. Ein durchaus sehr nützlicher ist der MyFacesExtensions Filter, welcher u.a. das Größenlimit einer auf den Server zu ladenden Datei angibt [Marinschek et al., 2007, Seite 143f].
  - Servlet
    - In der „web.xml“ muss ein FacesServlet angegeben sein, welches direkt über die Referenz (javax.faces.webapp.FacesServlet) spezifiziert werden kann. Andere Möglichkeiten wären das Servlet von „org.apache.myfaces.webapp.MyFacesServlet“ abzuleiten oder dieses selbst zu implementieren. Neben der Angabe des Servlets muss auch ein Servletmapping vorhanden sein [Marinschek et al., 2007, Seite 52].
    - Ein Beispiel für ein Servlet inklusive dessen Mapping ist unter [McGraw-Hill, 2008] zu finden.
  - Welcome File

Faces-config.xml

Diese Datei dient zur Registrierung von Komponenten, Renderern, Validatoren und Konvertern (für Entwickler von Komponenten) und zur Konfiguration von Managed-Beans und Navigationsregeln, wie das folgende Beispiel verdeutlichen soll:

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config version="1.2" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">


<managed-bean>
<managed-bean-name>loginBean</managed-bean-name>
<managed-bean-class>beans.LoginBean</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean


<navigation-rule>

    <from-view-id>/login.xhtml</from-view-id>

    <navigation-case>
      <from-outcome>success</from-outcome>
      <to-view-id>/admin.xhtml</to-view-id>
    </navigation-case>

    <navigation-case>
      <from-outcome>failure</from-outcome>
      <to-view-id>/login.xhtml</to-view-id>
    </navigation-case>

</navigation-rule>
```

Denkbar wäre in diesem Beispiel ein Szenario mit Username und Passwort als Input-Felder und einem Login-Button. Betätigt der Benutzer die Schaltfläche, wird die registrierte Action Methode aufgerufen, in der Benutzername und Passwort überprüft werden. Sind diese Eingaben korrekt, wird der String „success“ retourniert, andernfalls wird „failure“ zurückgegeben. Die Navigationsregel macht den Rest und navigiert den Benutzer bei erfolgreichem login auf die Seite „admin.xhtml“.

5.2 RichFaces

Wie der Name RichFaces bereits erahnen lässt, handelt es sich dabei um ein Framework zur Erstellung von Rich Internet Applications (RIAs). Was genau ist aber eine RIA?

5.2.1 Rich Internet Application

> „Rich Internet Application (RIA) sind Webapplikationen, die mit einer wesentlich interaktiveren Benutzerschnittstelle ausgestattet sind, als wir das bisher von den auf HTML-basierten poor ugly web applications (PUWA) gewohnt waren“ [Walter, 2008, Seite 333].

Ziel einer RIA ist es, dem Benutzer ein User Interface mit der Übersichtlichkeit, Flexibilität und Interaktivität einer lokalen Desktop-Applikation zur Verfügung zu stellen. Dadurch soll die Usability für den Benutzer von Webanwendungen gesteigert werden.

Aufgrund der zuvor genannten Eigenschaften wurden RIAs immer populärer. Gleich zu Beginn dieser Phase wurden einige Frameworks für die Entwicklung von Rich Internet Applications von namhaften Firmen wie Macromedia, Microsoft oder Sun Microsystems auf den Markt gebracht. Diese unterstützten zwar eine Vielfalt an UI-Komponenten, waren aber im Vergleich zu Desktop Anwendungen immer noch sehr langsam. Um dem Performance-Problem entgegen zu wirken etablierte sich bei den Entwicklern die Technologie AJAX [Paulson, 2005, Seite 14].

5.2.2 AJAX (Asynchronous JavaScript and XML)

AJAX ist eine Zusammensetzung von bereits in den 1990ern entwickelten Technologien. Dazu zählen laut [Paulson, 2005, Seite 14f]:

- Dynamic HTML
- XML (Extensible Markup Language)
- CSS (Cascading Stylesheets)
- DOM (Document Object Model)
- JavaScript
- XMLHttpRequest

Der Grundgedanke von AJAX ist, nur die Teile der Seite zu aktualisieren, deren Werte sich auch tatsächlich geändert haben und nicht wie ursprünglich bei jedem Request die ganze Seite neu zu laden [Katz, 2008, Seite 5].

Abbildung 5.5 stellt den Ablauf einen gewöhnlichen HTTP-Anfrage und Antwort im Gegensatz zu den unter Verwendung von AJAX generierten Prozess dar.

Wie die Abbildung zeigt, verwendet AJAX JavaScript als Laufzeitumgebung am Client und XML zum Datenaustausch zwischen Client und Server, um die zu aktualisierenden Komponenten herauszufinden

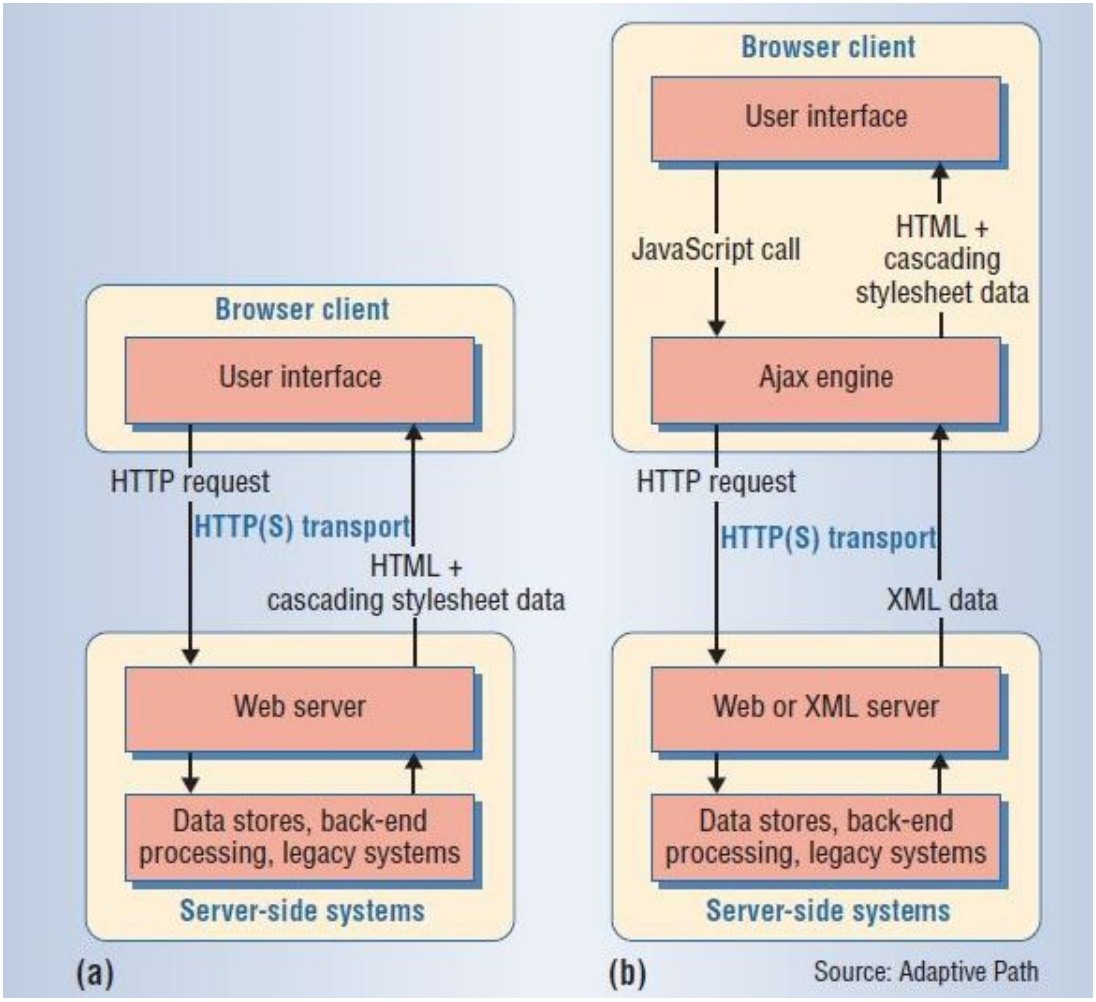

Abbildung 5.5: a) Stellt den Ablauf einer gewöhnlichen HTTP-Anfrage dar, während b) den Ablauf einer AJAX-Anfrage darstellt [Paulson, 2005, Seite 16].

und nur diese neu zu laden. Im Gegensatz dazu wird im Standard-Zyklus immer die komplette Seite neu aufgebaut.

5.2.2.1 Funktionsweise

Die AJAX-Engine fängt Benutzereingaben ab, stellt Daten dar und verarbeitet Interaktionen durch den Anwender auf der Client Seite. Werden zusätzliche Daten benötigt, wird ein XMLHttpRequest abgesetzt. Dieser holt sich die Daten vom Server im Hintergrund, sodass der Benutzer nichts von dem Request in Form einer Verzögerung wahrnimmt (Asynchron). Der Benutzer kann, anstatt auf die Antwort des Servers zu warten, weiter mit der Applikation interagieren [Paulson, 2005, Seite 16].

5.2.2.2 Vorteile von AJAX

Laut [Paulson, 2005, Seite 16] ergeben sich folgende Vorteile durch den Einsatz von AJAX:

- Performance

  - Anfragen werden im Hintergrund ausgeführt
  - Kein kompletter Seiten-Reload
- Traffic zwischen Client und Server wird minimiert
- Durch die vielseitige Anwendung steigt die Erfahrung der Anwender
- Plattformunabhängigkeit
  - Eine große Anzahl von Benutzern wird direkt erreicht

5.2.2.3 Nachteile von Ajax

Natürlich gibt es nicht nur Vorteile von AJAX. Deshalb sei an dieser Stelle kurz auf die Nachteile hingewiesen. Diese sind laut [Paulson, 2005, Seite 16f] und [Katz, 2008, Seite 6] folgende:

- Nachträgliches Hinzufügen von AJAX-Funktionalität in ein bereits vorhandenes, komplexes Projekt kann sehr kompliziert werden
- Je nach Browser wird JavaScript oft anders interpretiert
- Ältere Browser unterstützen AJAX nicht vollständig
- Kann nicht für Audio - oder Videowiedergabe eingesetzt werden
- Sicherheit
  - „Ajax funktioniert nicht ohne JavaScript. Das ist vor allem aus Sicherheitsgesichtspunkten das größte Problem. JavaScript ist das Einfallstor für alle Arten von Internet-Kriminalität.“ [empulse, 2010]
- Die Lernkurve in AJAX ist sehr flach
- Manuelle AJAX-Entwicklung wird sehr schnell komplex und kompliziert

Um relativ einfach mit Ajax arbeiten zu können, gibt es einige Frameworks, die diese Arbeit erleichtern sollen. RichFaces ist ein solches und da es im praktischen Teil zum Einsatz kommt, wird es im Abschnitt 5.2.3 detailliert beschrieben.

5.2.3 Definition von RichFaces

RichFaces ist eine Komponenten-Bibliothek, welche Komponenten mit AJAX-Unterstützung zur Verfügung stellt. Diese ersetzt NICHT JSF, sondern baut darauf auf. Eine ganze Menge zusätzlicher Komponenten werden mit bereits implementiertem JavaScript-Code angeboten. Dieses Framework bringt also die großen Vorteile, dass sich der Entwickler nicht mit zeitaufwändigen JavaScript-Aufgaben herumschlagen muss und dass sehr viele Komponenten in der Library enthalten sind [Katz, 2008, Seite 7].

Im Wesentlichen umfasst RichFaces zwei Tag-Libraries:

- Rich

- A4j

Die „rich“-Bibliothek unterstützt AJAX auf Komponentenebene. Das bedeutet, dass alle Komponenten der „rich-Library“ bereits AJAX-Funktionalität integriert haben. Um einer anderen bzw. mehreren Komponente(n) die Möglichkeit zu geben, einen AJAX Request abzuschicken, wird die „a4j“-Bibliothek benötigt [Katz, 2008, Seite 7].

Eine Übersicht der von RichFaces zur Verfügung gestellten Komponenten inklusive kurzer Beispiele kann unter [jboss.org, 2010] oder [RichFaces] nachgelesen werden.

In den Abschnitten 5.2.3.1 und 5.2.3.2 werden noch zwei weitere wichtige Aspekte, die bei der Verwendung von RichFaces von Bedeutung sein können, beschrieben.

5.2.3.1 Skins

Ein weiteres wichtiges Thema in RichFaces ist das Aussehen der Applikation. Dazu gibt es in RF die Möglichkeit, bereits vordefinierte oder selbst erzeugte Skins zu verwenden. Diese bieten die Möglichkeit, das Aussehen in einer zentralen Stelle zu definieren. Skinnability kann als Abstraktion von CSS betrachtet werden, unterstützt aber auch normale CSS-Deklarationen [Katz, 2008, Seite 219].

Im Zuge des Projekts wird ein bereits vordefinierter Skin verwendet. Deshalb möchte ich hier nicht im Detail auf das Thema „Skinning“ eingehen. Wer ergänzende Informationen zu diesem Thema haben möchte, sei auf [jboss.org, 2010, Seite 62ff] verwiesen.

5.2.3.2 Component Development Kit

Aus Gründen der Vollständigkeit sei erwähnt, dass es in RichFaces die Möglichkeit gibt, eigene Komponenten zu entwickeln. Da zur Umsetzung des praktischen Teils die vordefinierten Komponenten ausreichten, möchte ich dieses Thema hier nicht im Detail betrachten. Wie die Entwicklung eigener Komponenten funktioniert, kann unter [jboss.org, 2009] nachgelesen werden.

5.2.4 Technische Anforderungen an RichFaces

In diesem Abschnitt möchte ich auf die technischen Anforderungen zur Verwendung von RichFaces eingehen. Es soll ein Überblick über die Grundvoraussetzungen vermittelt werden, die erfüllt sein müssen, um überhaupt mit RF (Version 3.3.3) arbeiten zu können. Diese sind laut [jboss.org, 2010, Seite 3f] die folgenden:

- Java
  - JDK (Java Development Kit) 1.5 oder höher
- Unterstützte JavaServer Faces Implementierungen
  - **Sun JSF-RI** 1.2_x (1.2_14 wird empfohlen), 2.x
  - MyFaces 1.2.x (1.2.5 recommended), 2.x

  - Facelets 1.1.x
  - Seam 2.x
- Java application server oder servlet container
  - **Apache Tomcat** 5.5 - 6.0
  - BEA WebLogic 9.1 - 10.0
  - Resin 3.1.x
  - Jetty 6.1.x
  - Sun Application Server 9 (J5EE)
  - Glassfish V2, V3
  - JBoss 4.2.x - 5
  - Websphere 7.0. oder höher
  - Geronimo 2.0 oder höher
- Browser (auf der Client Seite)
  - Firefox 3.0 oder höher
  - Google Chrome
  - Internet Explorer 6.0 oder höher
  - Opera 9.5 oder höher
  - Safari 3.0 oder höher
- RichFaces Framework

Dieser Abschnitt sollte einen Überblick über die technischen Vorraussetzungen zur Verwendung von RichFaces geben. Informationen zur Installation der verwendeten Entwicklungsumgebung und der Erstellung eines RichFaces-Projekts in Eclipse sind im Anhang zu finden.

Kapitel 6

Umsetzung der Applikation

6.1 Generierung der Statistiken in Java

Den Kern zur Generierung der Statistiken in Java bildet die bereits existierende Library „jgnuplot“. Es werden zwar nicht alle Output-Terminals von gnuplot unterstützt, aber sowohl das Terminal „png“ als auch „postscript“ sind in der library implementiert [Sourceforge.net].

Aufgrund der Anforderungen an das Projekt (siehe Kapitel 3) sind diese beiden Output-Typen ausreichend. Ich habe mich dazu entschlossen, jgnuplot zu verwenden, weil damit bereits eine funktionierende library zur grafischen Darstellung mittels gnuplot existiert und der Source code sehr übersichtlich und relativ einfach gehalten ist. Ein weiterer Entscheidungsfaktor für jgnuplot ist, dass viele Einstellungsmöglichkeiten unterstützt werden, wie etwa Output-Typen, Style, Linientyp, etc. Im folgenden Abschnitt wird die Funktionalität und die Funktionsweise von jgnuplot detailliert betrachtet.

6.1.1 Jgnuplot - Das Grundgerüst

Das Klassendiagramm in Abbildung 6.1 zeigt die Klassen von jgnuplot mit einigen Member-Variablen und Funktionen. Es dient ausschließlich zur Veranschaulichung und dazu, Überblick zu bekommen. Da aus Gründen der Übersichtlichkeit nicht alle Eigenschaften dargestellt werden können, habe ich ein paar wichtige und im Projekt verwendete Befehle ausgewählt, die in der Grafik angezeigt werden.

Vorweg sei erwähnt, dass viele Klassen nur Konstanten beinhalten. Diese dienen ausschließlich dazu, diverse Einstellungen für die plots zu spezifizieren, wie etwa Art des Diagramms, Linientyp, Punkttyp, etc. Die Funktionalität befindet sich ausschließlich in den Klassen „Graph“ und „Plot“. Im Anschluss werde ich die einzelnen Klassen detaillierter betrachten und beschreiben.

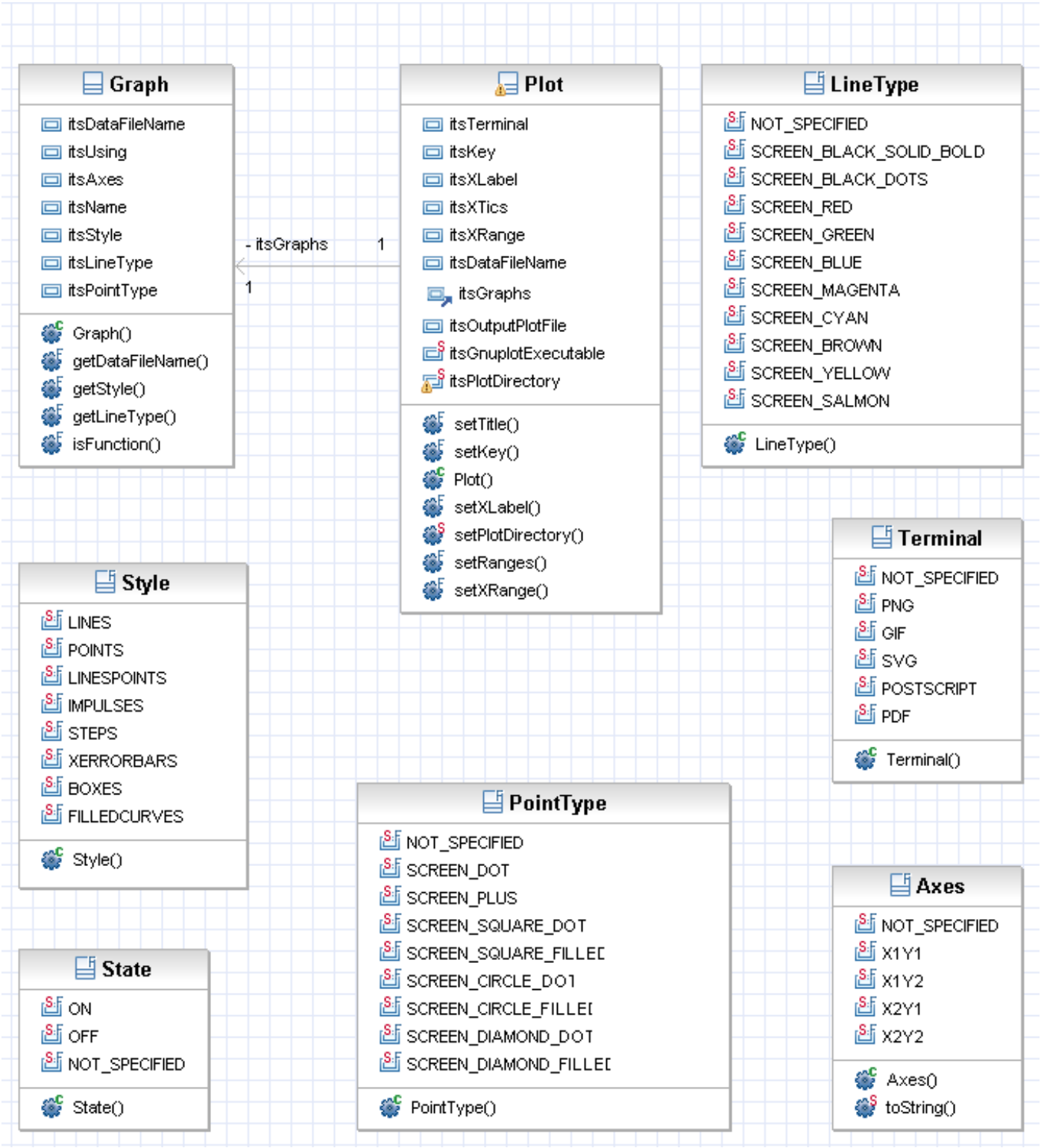

Abbildung 6.1: Klassendiagramm von „Jgnuplot“.

6.1.1.1 Style

Mithilfe der Klasse „Style“ wird die Art des Diagramms festgelegt, z.B. Liniendiagramm, Punktdiagramm, etc. Eine Übersicht über die einzelnen Stile sowie deren Aussehen und Verwendung ist in der

gnuplot-Dokumentation [Crawford., 2010, Seite 40ff] dargestellt.

6.1.1.2 LineType

Um die Linienart (Farbe und Form) zu definieren, kann auf die Klasse „LineType“ zugegriffen werden. Diese enthält einige Konstanten hierzu. Welche dies genau sind, kann in der api unter [Sourceforge.net] - „LineType“ nachgelesen werden. Wird kein Typ gesetzt, wird eine durchgehende Linie (solid) gezeichnet.

6.1.1.3 PointType

Wie bei einem Liniendiagramm der „LineType“, kann für ein Punktdiagramm der „PointType“ festgelegt werden. Mögliche Werte sind Circle, Star, Square, etc. Die genaue Liste der unterstützen Punkttypen kann auch hier wieder in der api [Sourceforge.net] nachgelesen werden.

6.1.1.4 Terminal

Wie bereits in Abschnitt 4.2.4.6 erwähnt wurde, unterstützt gnuplot eine Vielzahl von Grafikgeräten. Die Klasse „Terminal“ besitzt unterschiedliche Verarbeitungsmöglichkeiten um den gewünschten Output zu erzeugen [Sourceforge.net].

Im Falle dieses Projekts werden die Dateien immer gespeichert. Verwendet werden dazu die Terminals „Png“ (zum Erzeugen von png-Dateien) und „Postscript“ (für das Speichern von eps-Dateien).

6.1.1.5 State

„State“ dient ausschließlich zum Speichern des Status, ähnlich einer Bool’schen Variable. Der Unterschied besteht darin, dass „State“ drei Werte annehmen kann (ON, OFF, NOT_SPECIFIED) [Sourceforge.net]

6.1.1.6 Axes

Wie bereits im Abschnitt 4.2.4.2 erwähnt wurde, gibt es in gnuplot (auf zweidimensionaler Ebene) vier Achsen. Mit „Axes“ kann spezifiziert werden, welche beiden verwendet werden sollen. Standard ist X1Y1, also die Achse unten und die Achse links [Sourceforge.net].

6.1.1.7 Graph

In der Klasse „Graph“ wird über den Konstruktor ein Objekt angelegt und als Parameter können einige Eigenschaften übergeben werden, um diese für den jeweiligen Graphen zu setzen. Grundsätzlich gibt es die folgenden zwei Varianten zur Erzeugung eines Graphen:

1. Funktion als erster Parameter
2. Input Data-File als erster Parameter

Zu jeder der zwei Möglichkeiten gibt es wieder unterschiedliche Konstruktoren hinsichtlich der zu setzenden Werte. Eine Übersicht zu den verschiedenen Möglichkeiten, einen Graphen anzulegen, ist unter der „jgnuplot"-API [Sourceforge.net] - Graph, zu finden. Für das Projekt wird hauptsächlich der folgende Konstruktor verwendet:

```
public Graph(String theDataFileName, String theUsing,int theAxes,
             String theName, int theStyle)
```

„TheDataFileName" gibt den zu zeichnenden Datensatz (.dat Datei) an. Mit dem Parameter „theUsing" wird spezifiziert, welche Spalte der Inputdatei als x- bzw. y-Wert dargestellt wird (siehe Abschnitt 4.2.5.9). „TheAxes" gibt an, welche Achsen (Standard:X1Y1) verwendet werden. „TheName" setzt den Namen des Graphen und „theStyle" legt fest, welche Art von Diagramm erstellt wird.

6.1.1.8 Plot

„Plot" beinhaltet die Logik von jgnuplot und einige Parameter, die für jeden Plot gesetzt werden können. Zwei Werte müssen unbedingt angegeben werden, damit jgnuplot fehlerfrei auf gnuplot zugreifen kann:

1. itsGnuplotExecutable (gibt an, wo sich die ausführbare gnuplot.exe-Datei befindet)
2. itsPlotDirectory (legt den Output-Ordner für die erzeugten Grafiken fest) [1]

Neben diesen gibt es aber auch noch andere wichtige Parameter, die für jeden „Plot" gesetzt werden können:

- setKey → die Legende
- setXLabel → Beschriftung der x-Achse
- setXRange → sichtbarer Bereich der x-Achse
- setXTics → Abstand auf der x-Achse
- setOutput → Name der zu generierenden Output-Datei
- u.v.m. (siehe [Sourceforge.net])

Wie Abbildung 6.1 zeigt, gibt es eine Verbindung (Assoziation) zwischen zwei Klassen. Plot beinhaltet als Member-Variable einen Stack von Graphen. Mit der Funktion

```
pushGraph (Graph graph)
```

[1] Wie diese Befehle von der Web-Anwendung gesetzt werden, wird im Abschnitt 6.2.5.3 beschrieben.

können zuvor erzeugte Graphen zum Plot hinzugefügt werden.

An dieser Stelle möchte ich die „plot()“-Funktion genauer betrachten, weil diese für die Visualisierung der Daten verantwortlich ist und somit das Herzstück von jgnuplot darstellt. Der Ablauf lässt sich wie folgt darstellen:

- Es wird eine „.plt.txt“-Datei erzeugt, die alle Anweisungen zur Erzeugung des Plots beinhalten wird und als Input für gnuplot dient. Diese wird später durch den „load“-Befehl, welcher bereits in Abschnitt 4.2.5.8 ausführlich beschrieben wurde, geladen und zeilenweise von gnuplot ausgeführt.
- Alle Eigenschaften der „Plot“-Klasse werden dahingehend überprüft, ob Werte gesetzt wurden oder nicht. Wenn ja, wird der entsprechende gnuplot-Befehl mit diesem Wert als Parameter in die Datei geschrieben, wie das folgende Beispiel zeigt.

```
if (itsTitle != null) {
  itsOutputPlotFile.write("set title \"" + itsTitle + "\"");
  if (itsCommandsOnOneLine)
    itsOutputPlotFile.write("; ");
  } else {
  itsOutputPlotFile.write("\n");
  }
}
```

  - Es wird überprüft, ob ein Titel gesetzt wurde und gegebenenfalls wird dieser in das OutputPlotFile geschrieben und ein Zeilenumbruch eingefügt. Ist kein Titel vorhanden, passiert nichts. Auf diese Art und Weise werden alle Eigenschaften der Plot-Klasse abgearbeitet.

- Sind alle Grundeinstellungen für den Plot gesetzt, wird „plot“ in die Datei geschrieben. Wie dieser Befehl funktioniert, wurde bereits in Abschnitt 4.2.5.9 detailliert beschrieben.
- Anschließend iteriert die „plot()“-Funktion alle zu zeichnenden Graphen mit deren Eigenschaften durch
  - Resultat davon ist, dass die zu zeichnende Input-Datei der Graphen hinter den Plot-Befehl geschrieben wird.
  - Danach werden die Einstellungen, die den Graphen betreffen, in die Datei geschrieben (z.B.: „using 1:2“).
- Damit ist die Erzeugung der Plot-Datei abgeschlossen. Diese könnte nun an gnuplot übergeben und abgearbeitet werden. Der Inhalt ist im Prinzip eine Aneinanderreihung mehrerer gnuplot-Befehle, wie das folgende Beispiel verdeutlichen soll.

```
set title "Plot Frequency"
unset grid
unset key
unset parametric
unset polar
set xlabel "Random Value"
set ylabel "Frequency"
set terminal postscript
set output "E:/documents/frequency_plot.eps"
```

```
unset logscale
set autoscale
set xrange [0:10]
set yrange [0.0:5.0]
plot "E:/documents/frequency_plot.dat" using 1:2 title
"frequency graph" with impulses
```

- Der letzte Schritt in der „plot()“-Funktion ist der Aufruf von gnuplot mit der zuvor generierten Input-Datei als Parameter. Dazu wird ein Prozess über den „ProcessBuilder“ in Java mit den Parametern „GnuplotExecutable“ und „tempPlotFile“ angelegt. Das führt dazu, dass „gnuplot.exe“ aufgerufen und die zuvor erzeugte „.plt.txt“-Datei als Parameter mitgegeben wird. Damit die Plots auch gespeichert werden, ist es notwendig, dass die Eigenschaften „terminal“ und „output“ gesetzt wurden, wie im zuvor gezeigten Beispiel ersichtlich ist.

6.1.2 Generierung der Statistiken

Wie die Plots mit jgnuplot in Java erzeugt werden, wurde zuvor im Abschnitt 6.1.1 ausführlich beschrieben. Wer aber übernimmt diese Generierung und weiß, welche Statistiken generiert werden sollen? Abbildung 6.2 zeigt, welche Klassen dazu erzeugt wurden und wie diese mit jgnuplot interagieren.

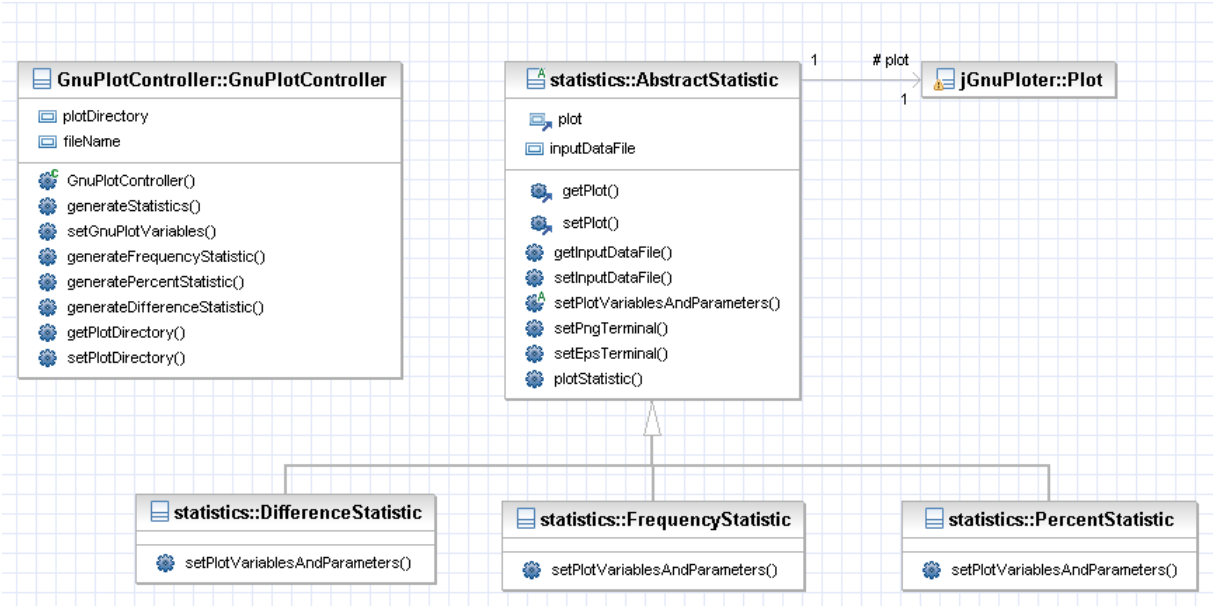

Abbildung 6.2: Klassendiagramm zur Generierung der Statistiken (Interaktion mit jgnuplot).

6.1.2.1 Arten und Eigenschaften der Statistiken

Zum jetzigen Zeitpunkt des Projekts gibt es drei unterschiedliche Statistiken, wie in Abbildung 6.2 ersichtlich ist:

- DifferenceStatistic
- FrequencyStatistic

- PercentStatistic

Um das Hinzufügen zusätzlicher Statistiken möglichst generell und einfach zu halten, habe ich mich dazu entschlossen, eine AbstractStatistic anzulegen. Diese beinhaltet alle Daten und Funktionen, die alle Statistiken gemeinsam haben, wie etwa das „InputDataFile“, welches die darzustellenden Werte für die Statistik enthält oder den zu generierenden Plot. Die Funktionen „setPngTerminal“ bzw. „setEpsTerminal“ rufen lediglich die Funktion „setOutput“ aus der Klasse „Plot“ auf. Parameter sind der Terminaltyp und die Output-Datei. Auch diese beiden Funktionen sind essentiell für die Generierung einer Statistik, da sowohl eine eps-Datei als auch eine png-Datei erzeugt werden soll. PlotStatistic ruft „plot()“ mit inputDataFile als Parameter auf. Dieser dient ausschließlich dazu, die Plot-Datei mit dem gleichen Namen generieren zu können.

Je nachdem, welche Statistik generiert wird, werden unterschiedliche Eigenschaften im Plot gesetzt. Da sich diese unterscheiden, werden sie in den jeweiligen konkreten Statistiken gesetzt, wie z.B.:

- xlabel, ylabel
- title
- xRange, yRange
- usw.

Ebenfalls wird in der konkreten Statistik das Graph-Objekt angelegt und auf den Stack des Plots gelegt. Der Grund dafür ist, dass die Graphen nicht für alle Statistiken die gleichen Parameter und Eigenschaften haben. Alle statistikspezifischen Einstellungen werden in der Funktion „setPlotVariablesAndParameters“ in der jeweiligen konkreten Ausführung implementiert.

6.1.2.2 GnuplotController - Die Logik

Der GnuPlotController hat die Aufgabe, die notwendigen gnuplot-Variablen zu setzen (GnuPlotExecutable und PlotDirectory) und überprüft, welche Statistik generiert werden soll und leitet die dazu notwendigen Schritte ein.

Um zu wissen, wo die fertig generierten Dateien gespeichert werden sollen, kommuniziert der Controller mit der Web-Applikation. Bei einem FileUpload (hochladen der .dat-Datei) muss ein Ordner ausgewählt werden, in dem diese abgespeichert wird. Damit die Grafik-Dateien im selben Ordner mit der .dat-Datei und der .plt.txt-Datei liegen, wird bei jedem Upload ein neuer Controller angelegt, der den ausgewählten Ordner und den Dateinamen der hochgeladenen Datei als Parameter mitbekommt. Danach wird in der Funktion „setGnuPlotVariables“ genau dieser Ordner als plotDirectory gesetzt. Detailliertere Informationen zur Webapplikation sind im Abschnitt 6.2 zu finden.

Damit sichergestellt ist, dass auch die richtige Statistik generiert wird, ist der Dateiname, der (in der Web-Applikation) hochgeladenen Datei entscheidend. Es werden nur bestimmte Dateinamen vom Controller akzeptiert. Diese sind als Konstanten im Programm festgelegt. Um eine „Frequency-Statistik“ zu generieren, muss eine „.dat“-Datei mit der Bezeichnung „frequency_plot.dat“ auf den Server geladen werden. Bei der „Percent-Statistik“ muss diese „percent_plot.dat“ und bei der „Difference-Statistik“, „difference_plot.dat“ heißen. Nur wenn einer der zuvor erwähnten Dateinamen hochgeladen wird, kann der Controller die entsprechenden Schritte einleiten, um eine Statistik fehlerfrei zu erzeugen. Im Falle von

Erweiterungen der Applikation um andere Statistiken ist dies unbedingt zu beachten und im GnuPlotController entsprechend zu ergänzen bzw. anzupassen. Der Grund für diese Einschränkung der Dateinamen ist, dass laut Anforderungen nur ein Frequency-Plot pro Ordner erlaubt ist. Dasselbe gilt auch für die anderen Statistiken.

6.2 Umsetzung der Webapplikation

Aufgrund der in Abschnitt 5.2.3 beschriebenen Vorteile von RichFaces und der in Abschnitt 5.1.3 genannten Vorteile von JSF habe ich mich dazu entschlossen, die Web-Applikation mit diesen Technologien umzusetzen. Ein Nachteil dabei ist der große Aufwand, um mit dieser Technologie vertraut zu werden und die Hintergründe der Abläufe nachvollziehen zu können. Das kann ich nur bestätigen. Anfangs ist viel Zeit in die Lernphase zu investieren, um das Framework zu verstehen um effektiv damit arbeiten zu können. Dieser Aufwand rentiert sich aber, weil die Entwicklung der Webapplikation danach relativ schnell durchführbar ist.

Wie bereits bei der Definition von RF (siehe Abschnitt 5.2.3) beschrieben, gibt es eine Menge vordefinierter Komponenten. Diese unterstützten mich bei der Umsetzung meines Projekts sehr. Eine Menge an Zeitersparnis bringt auch der MVC-Ansatz hinsichtlich sich verändernder Anforderungen. Diese können mit verhältnismäßig wenig Aufwand korrigiert werden. Ein weiterer wichtiger Aspekt für die Verwendung von RichFaces in meinem Projekt war die sehr gute Unterstützung von AJAX; es gibt sowohl bereits vordefinierte „AJAX-fähige“ Komponenten, zusätzlich aber auch die Möglichkeit, allen anderen Komponenten AJAX-Unterstützung hinzuzufügen [jboss.org, 2010, Seite 1].

In den folgenden Abschnitten möchte ich vertiefend darauf eingehen, wie die einzelnen Anforderungen an die Web-Applikation im Projekt umgesetzt wurden. Dies beinhaltet die verwendeten Komponenten mit deren Einstellungen und Eigenschaften. Zusätzlich werden einige essentielle Funktionen näher betrachtet, wie etwa der FileUpload, u.a.

6.2.1 Generelle Einstellungen für die Webapplikation - Properties

Um die WebApplikation fehlerfrei ausführen zu können, müssen zuerst einige Einstellungen vorgenommen werden. Dazu gibt es im Ordner JavaSource ein Property-File („gnuplot.properties“). Dieses beinhaltet neben Benutzername und Passwort für den Admin-Bereich auch den Wurzel-Ordner für das anzuzeigende Dateisystem (Bsp.: “C:/tomcat/webapps/network-projekt/WebContent/“) auf dem Server. Des Weiteren ist ein Unterordner für diese Wurzel anzugeben, welche den Root-Pfad komplettiert (Bsp.: „/documents“). Ausgehend von diesem Pfad („C:/tomcat/webapps/network-projekt/WebContent/documents“), werden dem Benutzer alle Unterordner und Dateien angezeigt. Der Unterordner wird benötigt, um bei Auswahl eines Ordners nicht immer den kompletten Pfad anzeigen zu müssen. Ein weiterer Pfad der unbedingt gesetzt werden muss, ist der Pfad zur ausführbaren „gnuplot“-Datei, um auch Statistiken via „gnuplot“ generieren zu können (z.B. „C:/gnuplot/bin/wgnuplot.exe“ für die Windows Version).

6.2.2 Konstanten

Aktuell gibt es drei Statistiken, die der Benutzer generieren kann:

1. Frequency Statistik

2. Percent Statistik
3. Difference Statistik

Es werden nur bestimmte Dateinamen als Inputdatei von der Applikation akzeptiert. Diese Dateinamen lauten wie folgt:

- „frequency_plot“
- „percent_plot“
- „difference_plot“

Zu beachten ist auch, dass nur Inputdateien des Typs „.dat“ verarbeitet werden. Eine gültige Datei wäre z.B. „frequency_plot.dat“. Eine ungültige hingegen wäre z.B. „frequency.dat“.

Detaillierte Informationen zu den einzelnen Statistiken, wie etwa die unterschiedlichen Arten der Statistiken, die im Moment unterstützt werden oder wie diese generiert werden, sind in den Abschnitten 6.1.2 und 6.2.5.3 zu finden.

Eine weitere Konstante der Applikation ist eine Liste von Strings, welche alle zweidimensionalen Styles von gnuplot enthält („boxes“, „dots“, „filledcurves“, „impulses“, etc.)

Der Benutzer kann beim Hochladen der Datei auswählen, in welchen Style die zu generierende Statistik gezeichnet werden soll. Diese Konstanten und deren Einsatz in der Anwendung sind im Falle einer Erweiterung des Projekts sehr wichtig und unbedingt zu beachten.

6.2.3 Beans

Für die Bereiche „Login“, „Dateisystem“, „Datei hochladen“ (inklusive Generierung der Statistiken) und „Statistiken betrachten“ gibt es jeweils eine eigene Bean. Diese beinhaltet alle benötigten Eigenschaften für die korrekte Darstellung und die Ereignisbehandlungen.

Alle Beans im Projekt besitzen Session-Scope, d.h. sie existieren von Beginn bis Ende einer Session. Dieser Scope wurde gewählt, weil ich zugunsten der Usability darum bemüht war, möglichst viele Komponenten nur anzuzeigen, wenn der Benutzer diese auch tatsächlich verwenden darf. So ist beispielsweise der Upload einer Datei nur verfügbar, wenn der Anwender auch einen Ordner in der Applikation ausgewählt hat, in den er die Datei hochladen möchte. Ist kein Ordner selektiert, wird der Upload auch nicht angezeigt. Um Komponenten ein - bzw. ausblenden zu können, wird eine Variable der Bean an das „rendered“-Attribut der Komponente gebunden. Diese Variable muss vom Scope „Session“ sein [2]. Wichtige Eigenschaften bzw. Funktionen der einzelnen Beans werden in den folgenden Abschnitten detailliert beschrieben.

6.2.4 User Login

Laut Anforderungen (siehe Abschnitt 3) gibt es zwei unterschiedlich Arten von Benutzern:

[2] http://community.jboss.org/message/32368

1. Administrator
2. Standard-Anwender

Je nach Benutzer sollen gewisse Funktionalitäten verfügbar sein. Der Administrator hat alle Rechte und darf Ordner im Dateisystem anlegen, Dateien hochladen (Statistiken generieren) und die im Dateisystem vorhandenen Statistiken betrachten.

Der Standard-Anwender darf hingegen nur die vorhandenen Statistiken ansehen. Abbildung 6.3 zeigt die Login-Maske der Web-Applikation.

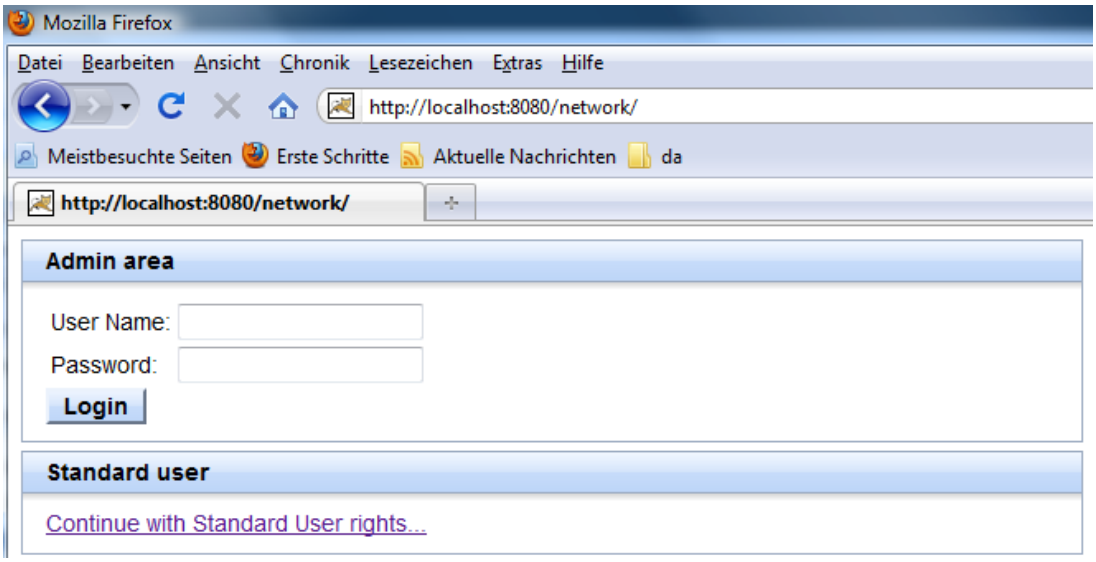

Abbildung 6.3: Die Login-Maske der Applikation.

Wird vom Benutzer der Standard-User ausgewählt, wird dieser über eine in der „faces-config.xml“ definierten Navigationsregel auf die Seite zur Betrachtung der Statistiken weitergeleitet. Diese wird im Abschnitt 6.2.5.4 detailliert beschrieben.

Gibt der Benutzer Benutzername und Passwort ein, wird durch Klick auf den Login-Button die Funktion „checkAdminUser“ der LoginBean aufgerufen. Dabei werden die Eingaben des Benutzers mit dem Admin-Benutzernamen und dem Admin-Passwort verglichen. Stimmt die Eingabe überein, wird „success“ retourniert und der Benutzer als Administrator auf die Seite „admin.xhtml“ weitergeleitet. Ist eine der Eingaben falsch („failure“), wird erneut die Login Seite angezeigt. Diese beiden Fälle werden auch über Navigationsregeln in der „faces-config.xml“ behandelt, wie folgender Auszug der Datei verdeutlichen soll:

```
<navigation-rule>
   <from-view-id>/start.xhtml</from-view-id>

      <navigation-case>
         <from-outcome>success</from-outcome>
         <to-view-id>/admin.xhtml</to-view-id>
      </navigation-case>

      <navigation-case>
         <from-outcome>failure</from-outcome>
```

```
         <to-view-id>/start.xhtml</to-view-id>
      </navigation-case>

      <navigation-case>
         <from-outcome>standardUser</from-outcome>
         <to-view-id>/dnd.xhtml</to-view-id>
      </navigation-case>

</navigation-rule>
```

Die Admin Funktionalitäten werden im Abschnitt 6.2.5 detailliert beschrieben.

6.2.5 Der Admin-Bereich

6.2.5.1 Tab-Panel

Aus Gründen der Übersichtlichkeit und der Usability habe ich den Admin-Bereich in zwei „Tabs“ (rich:tabPanel) aufgeteilt; ein Tab mit den Inhalten „Dateisystem“ und „FileUpload“ und einer zum Anzeigen der Statistiken (siehe Abbildung 6.4).

6.2.5.2 Dateisystem

Zur Darstellung des Dateisystems wurde im Projekt ein „rich:tree“ mit einem „recursiveTreeNodeAdopter“ verwendet. Die Verwendung dieser beiden Komponenten wird im folgenden Teil detaillierter betrachtet.

Tree
Folgender Code zeigt, wie die „rich:tree“-Komponente im Projekt verwendet wird.

```
<rich:tree id="tree" style="width:80%" ajaxRendered="true"
   ajaxSubmitSelection="true" switchType="ajax"
   nodeSelectListener="#{recursiveTreeNodeAdopterBean.processSelection}">

   <rich:recursiveTreeNodesAdaptor
      roots="#{recursiveTreeNodeAdopterBean.sourceRoots}" var="item"
      nodes="#{item.nodes}" />

   <rich:treeNode value="#{item.nodes}" />
</rich:tree>

<a4j:outputPanel id="selectedNode1">
   <h:outputText escape="false" id="selectedNode"
      rendered="#{recursiveTreeNodeAdopterBean.selectedNode != null}"
      value="Selected Node: #{recursiveTreeNodeAdopterBean.selectedNode}"/>
```

```
</a4j:outputPanel>
```

Grundvoraussetzung für den zuvor gezeigten Code ist die Registrierung der „recursiveTreeNodeAdopterBean“ in der „faces-config.xml“.

Wichtig bei der Tree-Komponente ist der „switchType“, welcher die Werte „server“, „client“ oder „ajax“ annehmen kann [jboss.org, 2010, Seite 263f]. In diesem Fall wurde „ajax“ und „ajaxSubmitSelection=true“ verwendet. Bei jedem Klick auf den Tree wird also ein Ajax-Request gesendet. Dies wird benötigt, um bei Selektion eines Knoten den ausgewählten Knoten in einem Textfeld anzuzeigen. Somit weiß der Anwender immer, welcher Ordner gerade ausgewählt ist, womit er bei den Aktionen „Unterordner anlegen“ und „FileUpload“ visuell unterstützt wird. Dieses Output-Feld ist von einem <a4j:outputPanel>mit der Eigenschaft „ajaxRendered=true“ umgeben. Dieses Attribut bewirkt, dass die Komponenten innerhalb des <a4j:outputPanels>bei jedem Ajax-Request neu gerendert werden [jboss.org, 2010, Seite 34].

Damit auch immer der aktuell ausgewählte Knoten aus dem Modell ausgelesen wird, setzt die Methode „processSelection“ die Variable „selectedNode“ [3] auf den Wert des selektierten Ordners, was unten stehender Code-Ausschnitt verdeutlichen soll. Wie der Name „nodeSelectListener“ bereits erahnen lässt, wird diese Methode bei jeder Knotenauswahl aufgerufen:

```
public void processSelection(NodeSelectedEvent event) {
   UITree tree = (UITree) event.getComponent();
   FileSystemNode node = (FileSystemNode) tree.getRowData();
   selectedNode = node.getPath();
}
```

Der Wert „selectedNode“ wird aktualisiert und das <a4j:outputPanel>wird mit dem neuen Wert angezeigt. Auch hier wird aus Gründen der Usability das Textfeld nur angezeigt, wenn auch tatsächlich ein Knoten ausgewählt wurde.

RecursiveTreeNodeAdopter

Der „rich:recursiveTreeNodeAdopter“ ist eine Komponente, die es ermöglicht, Daten-Modelle zu erzeugen bzw. zu verwenden, Knoten einzubinden und diese rekursiv abzuarbeiten [jboss.org, 2010, Seite 275f]. Die Dokumentation [jboss.org, 2010] empfehle ich in Kombination mit dem Beispiel des recursiveTreeNodesAdopter von [RichFaces] um die Funktionsweise dieser Komponente zu verstehen.

Kurz sei nur erwähnt, dass zuerst die Knoten der ersten Ebene ausgelesen und anschließend nach Kindern durchsucht werden. Wurde ein Kind gefunden, wird es in zweiter Ebene wiederum nach Kindern durchsucht. Dieser Vorgang wird für alle Knoten wiederholt und nennt sich rekursiv. Die Pfade aller Knoten werden als <rich:Treenode>angezeigt.

Abbildung 6.4 zeigt das Ergebnis der Zusammenarbeit dieser Komponenten und spiegelt die Ordnerstruktur ausgehend vom Wurzelordner wieder:

[3] Anm.: selectedNode ist eine Membervariable der recursiveTreeNodeAdopterBean.

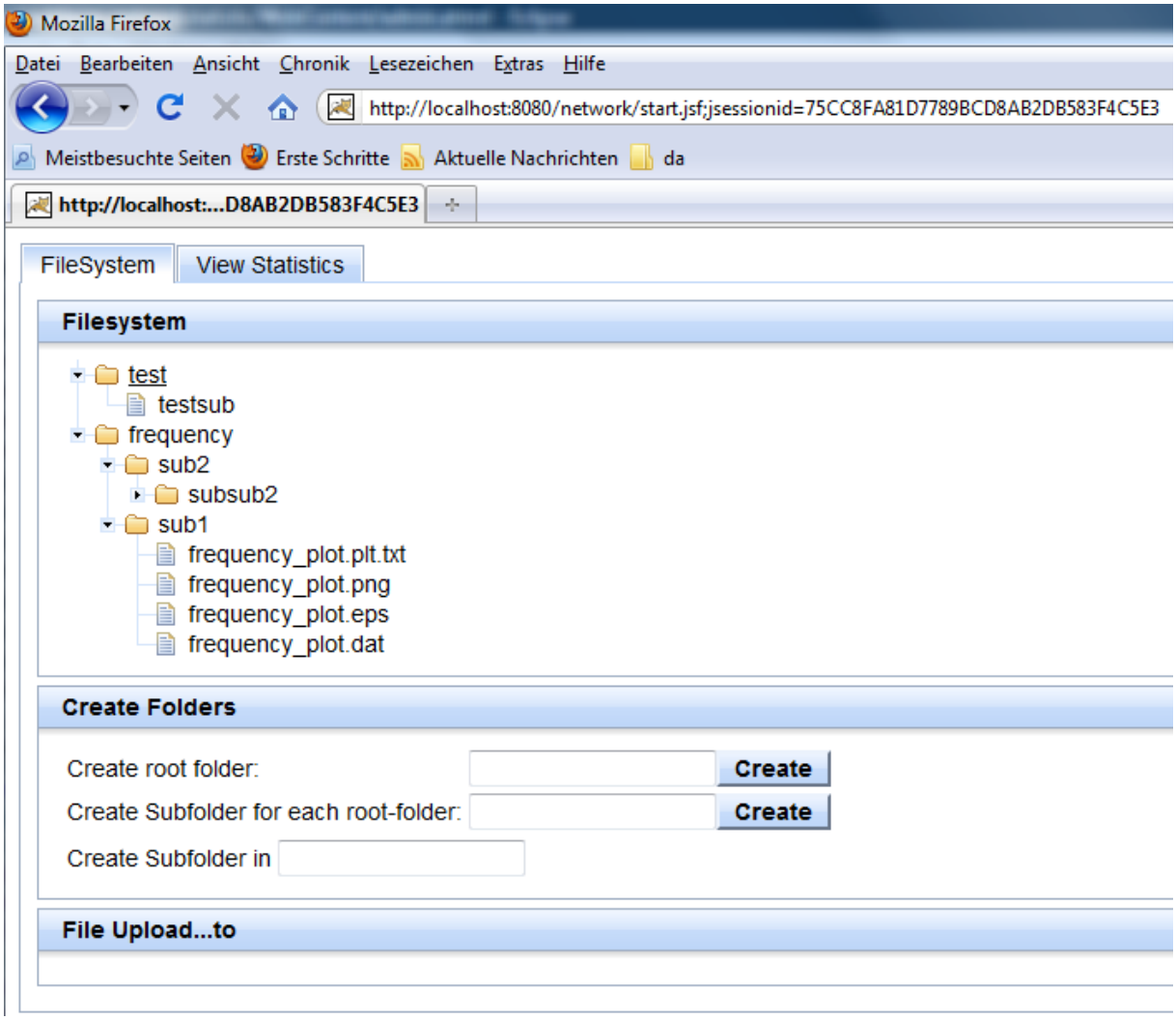

Abbildung 6.4: Der Admin-Bereich der Applikation.

Die Ordner „test“ und „frequency“ sind eine Ebene unter dem im Property-File spezifizierten Root-Ordner zu finden. Die Abbildung 6.4 zeigt auch, dass beliebig viele Ebenen erzeugt werden können („frequency”→ „sub2”→ „subsub2“).

Im Bereich unterhalb des Baumes gibt es die Möglichkeit, Ordner anzulegen. Dabei gibt es drei unterschiedliche Varianten:

1. Hauptordner anlegen
   (a) Gibt der Benutzer einen Ordnernamen (im Feld neben „create root folder“) ein und klickt anschließend auf den „Create“-Button, wird dieser Ordner als Hauptordner angelegt, d.h. auf derselben Ebene wie die Ordner „test“ und „frequency“.
2. Unterordner in allen Hauptordnern
   (a) Gibt der User im zweiten Textfeld einen Ordnernamen ein und betätigt er die daneben liegende Schaltfläche, wird in jedem Hauptordner ein Unterordner mit diesem Namen erzeugt.
   (b) Wird z.B. „subfolder“ in das Feld eingegeben und klickt der Benutzer anschließend auf „Create“ wird dieser in den Ordnern „test“ und „frequency“ als Unterordner angelegt, nicht allerdings in den Ordnern „sub1“, „sub2“ und „subsub2“.

3. Unterordner in ausgewähltem Ordner

   (a) Aus Gründen der Usability soll diese Schaltfläche nur verfügbar sein, wenn ein Knoten vom Benutzer ausgewählt wurde, in dem er einen Unterordner anlegen möchte. Dieses Verhalten wird auch hier wieder mit einem <a4j:outputPanel>erreicht. Ist ein Knoten ausgewählt, kann ein Unterordner in diesem angelegt werden (siehe Abbildung 6.5).

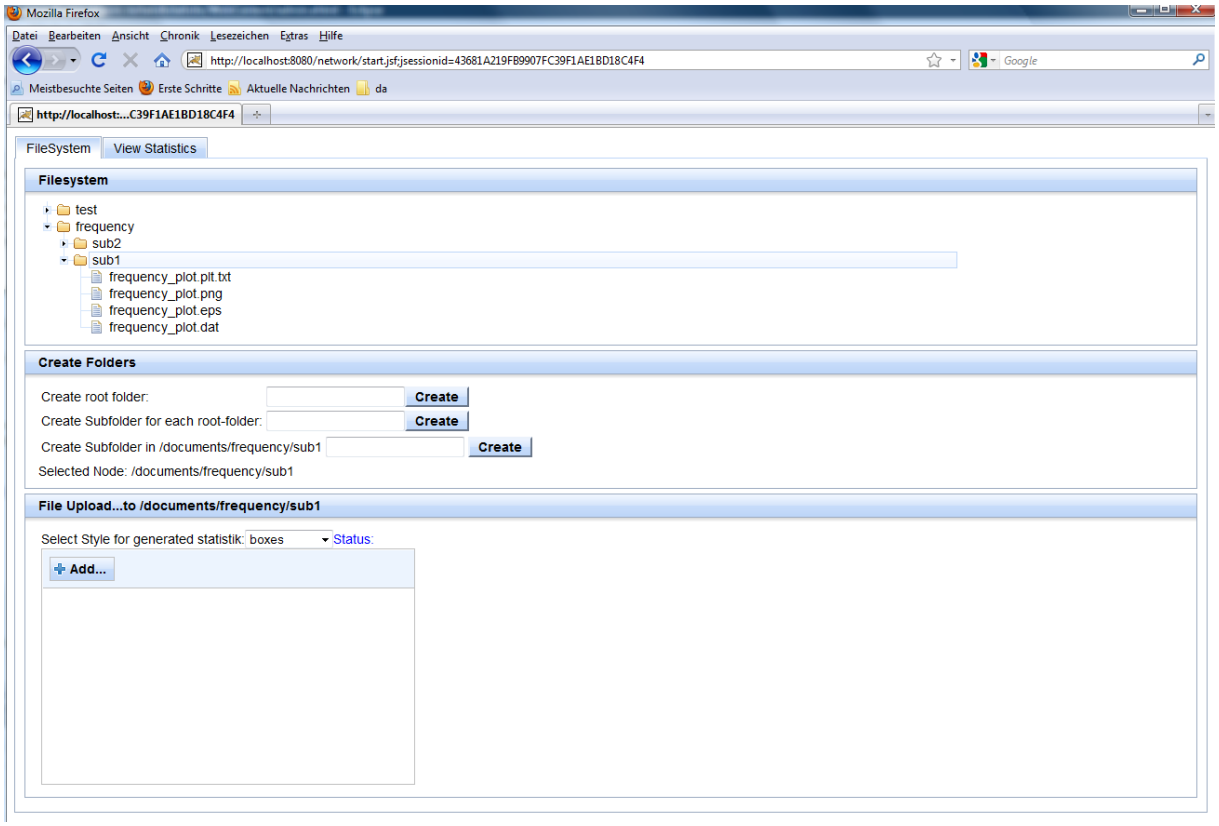

Abbildung 6.5: Unterordner anlegen ist nur verfügbar, wenn ein Ordner vom Benutzer ausgewählt wurde.

In Abbildung 6.5 wird auch deutlich, dass jetzt die FileUpload Komponente verfügbar ist. Auch diese ist an den „selectedValue“ gebunden und wird nur angezeigt, wenn ein Knoten ausgewählt wurde. Zusätzlich wird der aktuell selektierte Ordner auch im Kopf der FileUpload Komponente angezeigt.

Status Meldung

Wurde ein Ordner angelegt oder eine Datei hochgeladen, wird eine entsprechende Statusmeldung angezeigt. Tritt ein Fehler auf, wird auch dieser dargestellt. Beim FileUpload kann beispielsweise eine Fehlermeldung auftreten, wenn der Pfad zur ausführbaren gnuplot-Datei nicht stimmt. Wird ein Status angezeigt, muss dieser durch Klick auf den „OK“-Button bestätigt werden. Bevor dieser nicht betätigt wird, kann der User keine weiteren Aktionen ausführen. Dies soll sicherstellen, dass der Benutzer immer sofort ein Feedback über die aktuell ausgeführte Aktion erhält und er diese auch zur Kenntnis genommen hat (siehe Abbildung 6.6).

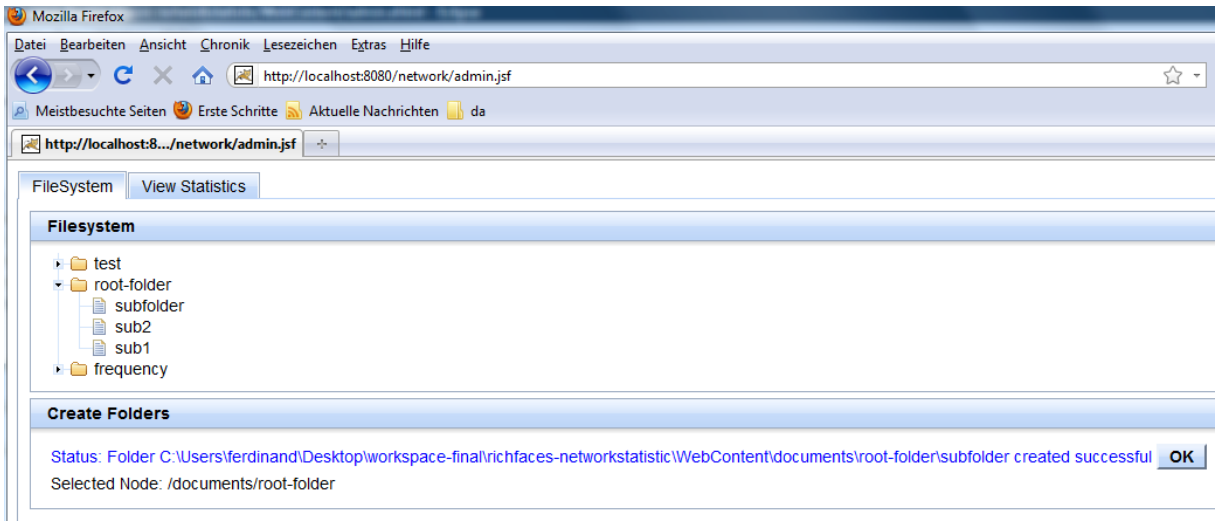

Abbildung 6.6: Beispiel einer Statusanzeige nach Anlegen eines Ordners.

6.2.5.3 Dateien hochladen (Generierung der Statistik)

Die Funktion „Hochladen einer Datei“ wurde im Projekt mit Hilfe der <rich:fileUpload>Komponente umgesetzt, welche laut [jboss.org, 2010, Seite 383] folgende Grundeigenschaften besitzt:

- Fortschritt des Uploads wird angezeigt
- Einschränkung der hochzuladenden Datei in Bezug auf Dateityp, Dateigröße und Anzahl der Dateien
- Flash wird unterstützt
- Möglichkeit, Uploads abzubrechen
- Automatische Uploads („immediateUploads=true“)
- Das Aussehen kann individuell angepasst werden

Neben der maximalen Dateigröße muss laut [jboss.org, 2010, Seite 383f] auch das Attribut createTempFiles in der „web.xml“ eingefügt werden, um mit der <rich:fileUpload>- Komponente zu arbeiten (siehe folgendes Beispiel):

```
<filter>
   <display-name>Ajax4jsf Filter</display-name>
   <filter-name>ajax4jsf</filter-name>
   <filter-class>org.ajax4jsf.Filter</filter-class>
   <init-param>
      <param-name>createTempFiles</param-name>
      <param-value>true</param-value>
   </init-param>
   <init-param>
      <param-name>maxRequestSize</param-name>
```

```
      <param-value>10000000</param-value>
   </init-param>
</filter>
```

Die Angaben der Dateigröße erfolgen in Byte. Dieses Beispiel würde also Dateien bis zu einer Größe von ca. 10MB akzeptieren.

Im folgenden Bereich wird der Code der <rich:fileUpload>-Komponente aus dem Projekt näher betrachtet. Sämtliche Eigenschaften für den FileUpload im Projekt sind in der Datei „FileUploadBeanRich“ zu finden wie etwa:

- String statusMessage
  - Beinhaltet den Status nach dem Upload (z.B.: „Plot generated successful“), aber auch evtl. aufgetretene Fehler
- List<File>files
  - eine Liste der hochzuladenden Dateien
- int uploadsAvailable
  - Anzahl der Dateien ist auf „1“ gesetzt, somit kann immer nur eine Datei pro Request auf den Server geladen werden
- boolean autoUpload
  - Ist auf false gesetzt → Der Benutzer muss noch auf den „Upload“-Button klicken, um diesen zu starten
- boolean useFlash
  - Wurde auf „true“ gesetzt, um die Flashunterstützung zu aktivieren
- String selectedNode
  - Beinhaltet den aktuell im Baum ausgewählten Ordner
  - Dieser wurde als Request Parameter an diese Bean übergeben (<f:param>) und wird in der „listener“-Methode ausgelesen bzw. dann gesetzt
- List<SelectItem>gnuplotStyles
  - Beinhaltet alle zweidimensionalen Gnuplot-Styles, die aktuell unterstützt werden
  - Der Benutzer kann einen Stil auswählen, in dem die Statistik generiert werden soll
- String selectedStyle
  - Wird im DropDown-Feld ein Wert geändert, wird dieser in die Variable „selectedStyle“ geschrieben
  - Diese wird an den „gnuplotController“ übergeben, damit dieser weiß, in welchem Stil die Statistik generiert werden soll

Wichtig ist hier auch die Schnittstelle zu gnuplot. Die „fileUploadBeanRich“ beinhaltet ein Objekt des „GnuPlotControllers“ als Member-Variable. Diesem werden die vom Benutzer gesetzten Werte (ausgewählter Ordner, Dateiname und der vom Anwender ausgewählte Stil) übergeben, damit dieser die weiteren Schritte zur Generierung der Statistiken einleiten kann (siehe Abschnitt „Generierung der Statistik“ auf Seite 84).

Abbildung 6.7 und der folgende Code-Auszug zeigen die FileUpload-Komponente im Projekt.

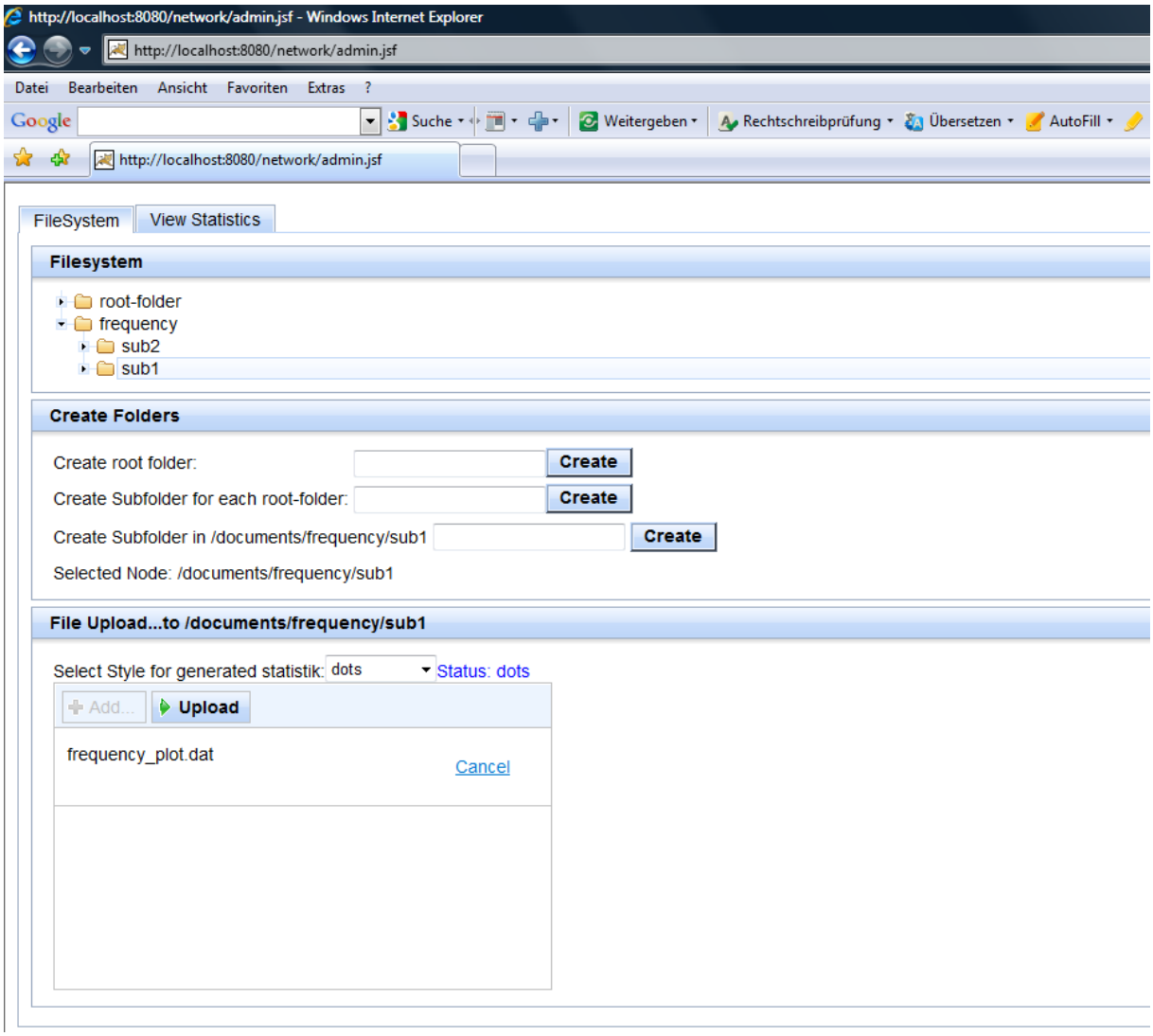

Abbildung 6.7: Die FileUpload-Komponente auf der Admin-Seite.

```
<rich:fileUpload id="upload"
   fileUploadListener="#{fileUploadBeanRich.listener}"
   maxFilesQuantity="#{fileUploadBeanRich.uploadsAvailable}"
   immediateUpload="#{fileUploadBeanRich.autoUpload}"
   acceptedTypes="dat" allowFlash="#{fileUploadBeanRich.useFlash}">

   <f:param name="selectedNode"
         value="#{recursiveTreeNodeAdopterBean.selectedNode}" />
</rich:fileUpload>
```

Der FileUploadListener

Eine der wichtigsten Funktionen des Projekts ist der fileUploadListener. Aus diesem Grund möchte ich diese hier detaillierter betrachten. Der Ablauf in dieser Methode ist wie folgt:

1. Da der komplette Pfad für die Erzeugung der Datei am Server bekannt sein muss und „selectedNode“ nur die Subordner (ausgehend vom Root-Ordner) beinhaltet, wird aus dem Property-File die „filesystem.root“ ausgelesen
2. Das UploadItem wird erzeugt
   (a) UploadItem item = event.getUploadItem();
3. Aus dem Request wird der zuvor erwähnte Parameter (selectedNode) ausgelesen und in die Eigenschaft „selectedNode“ der „fileUploadBeanRich“ geschrieben
4. Es wird überprüft, ob der Dateiname der hochzuladenden Datei korrekt ist (siehe Abschnitt 6.2.2)
5. Ist dies der Fall, wird die Datei im selektierten Pfad erzeugt und mit den Werten der lokalen Datei geschrieben
6. Wurde die Input-Datei für die Statistiken erfolgreich auf den Server geladen, werden nun die Statistiken generiert.
   (a) Dieser Prozess wird im folgenden Abschnitt detailliert betrachtet

Generierung der Statistik

Wie bereits zuvor erwähnt, werden die Statistiken im Zuge des FileUploads generiert. Dazu wird der „GnuplotController“ mit den folgenden Parametern erzeugt [4]:

1. PlotDirectory
   (a) Der aktuell ausgewählte Ordner, in dem der Plot gespeichert werden soll
2. FileName
   (a) Der gleiche Dateiname wie bei der hochzuladenden „.dat“ Datei
   (b) Die Dateiendungen werden entsprechend angepasst (z.B.: „.png“)
3. SelectedStyle
   (a) Der vom Benutzer ausgewählte Stil für die zu erzeugende Statistik (DropDown-Feld)
4. Properties
   (a) Um den Pfad der ausführbaren gnuplot-Datei auslesen zu könnnen

[4] Anm.: Ist bereits ein Objekt vom Typ „GnuPlotController“ vorhanden, werden nur die entsprechenden Werte gesetzt und kein neues Objekt erzeugt.

Beim Hochladen der Datei wird die Funktion „generateStatistics“ vom „GnuPlotController“ aufgerufen, welche wie folgt vorgeht:

- Setzen der allgemeinen GnuPlot-Variablen („gnuplotExecutable“ und „plotDirectory“)
  - „GnuplotExecutable“ wird aus dem Property-File ausgelesen
  - Die Variable „plotDirectory“ wird beim FileUpload auf den Wert des aktuell ausgewählten Ordners gesetzt und an den „GnuplotController“ übergeben
- Anhand des Dateinamens wird überprüft, welche Statistik generiert werden soll
  - Ein konkretes Exemplar dieser Statistik wird erzeugt (z.B.: „FrequencyStatistik“)

Alle Statistiken sind von der Klasse „Abstract Statistik“ abgeleitet, welche die Grundfunktionalitäten und Eigenschaften zusammenfasst. Dies soll vor allem die Erweiterung der Applikation um zusätzliche Statistiken erleichtern.

Spezifische Einstellungen, wie etwa Beschriftung der Achsen, Graphen, Abstände auf den Achsen, etc. werden in den konkreten Statistiken gesetzt. Sind alle Einstellungen entsprechend eingestellt, werden zwei Plots erzeugt - eine „.eps“- und eine „.png“-Datei.

Dazu wird in den Funktionen zur Generierung zuerst das „PngTerminal“ gesetzt und die Outputdatei über die Funktion „plotStatistic“ [5]erzeugt. Anschließend wird das Terminal auf „Eps“ gesetzt und wiederum die Funktion zum Plotten aufgerufen. Detaillierte Informationen zur „plot“-Funktion sind unter Abschnitt 6.1.1.8 nachzulesen.

Wurden diese Funktionen abgearbeitet, wird dem Benutzer eine Statusmeldung angezeigt. Im Optimalfall teilt ihm diese mit, dass die Statistiken erfolgreich generiert wurden. Andernfalls wird eine entsprechende Fehlermeldung angezeigt. Nach der erfolgreichen Generierung und Speicherung der grafischen Darstellungen stehen diese zur Betrachtung zur Verfügung. Auch dies wird von der Webapplikation unterstützt und wird im folgenden Abschnitt näher betrachtet.

6.2.5.4 Statistiken betrachten

Laut Anforderungen (siehe Abschnitt 3) darf sowohl der Administrator als auch der „Standard“-Benutzer die generierten Statistiken betrachten. Aus diesem Grund gibt es zwei nahezu identische „.xhtml“-Seiten im Projekt - die „images.xhtml“ und die „dnd.xhtml“. Der einzige Unterschied zwischen diesen beiden Dateien ist, dass alle Tags im „images.xhtml“ von einem <h:form>-Tag umgeben sind. In der „dnd.xhtml“ wird dieser nicht benötigt, weil diese Seite in einen Tab der „admin.xhtml“ eingebunden wurde und alle Tags der Admin-Seite bereits innerhalb eines <h:form>-Tags sind.

Anfangs hatte ich nur eine Seite für beide Benutzer angelegt und diese sowohl dem Standard - als auch dem Admin-Benutzer angezeigt. Daraus folgte im Admin-Bereich ein verschachteltes Formular (nested Form), was dazu führte, dass unter Internet Explorer keinerlei Aktionen ausgeführt werden konnten. Dies war der Grund für eine Aufteilung in zwei Dateien.

Alle Funktionen und Eigenschaften für diese Seite finden sich in der „ImagesBean“. Um die folgenden Erklärungen der einzelnen Funktionalitäten besser verständlich zu machen, möchte ich zu Beginn die Eigenschaften der Bean zeigen und beschreiben, wozu diese benötigt werden:

[5]Anm.: PlotStatistic ruft die „plot“-Funktion von „jgnuplot“ mit der Input-Datei als Parameter auf.

- List <String>imagePaths
  - Beinhaltet alle (short)-Pfade zu den Statistiken. Der Anwender soll eine Übersicht erhalten, welche Statistiken überhaupt verfügbar sind
- List <String>imagesDetailView
  - Beinhaltet alle Pfade von Statistiken, die aktuell betrachtet werden (d.h. sich in der Detailansicht befinden)
- List <SelectItem>numberOfColumns
  - Mögliche Anzahlen von Spalten für die Detailansicht
- Integer selectedNumberOfColumns = 2
  - Die aktuell gewählte Anzahl von Spalten für die Detailansicht

Zu allen zuvor erwähnten Variablen gibt es entsprechende Getter- bzw. Setter-Methoden, um die Werte auszulesen bzw. zu verändern.

Abbildung 6.8 soll einen Überblick über die Seite und deren Funktionalitäten vermitteln. Im Anschluss werden die verwendeten Komponenten mit deren Einstellungen detailliert betrachtet.

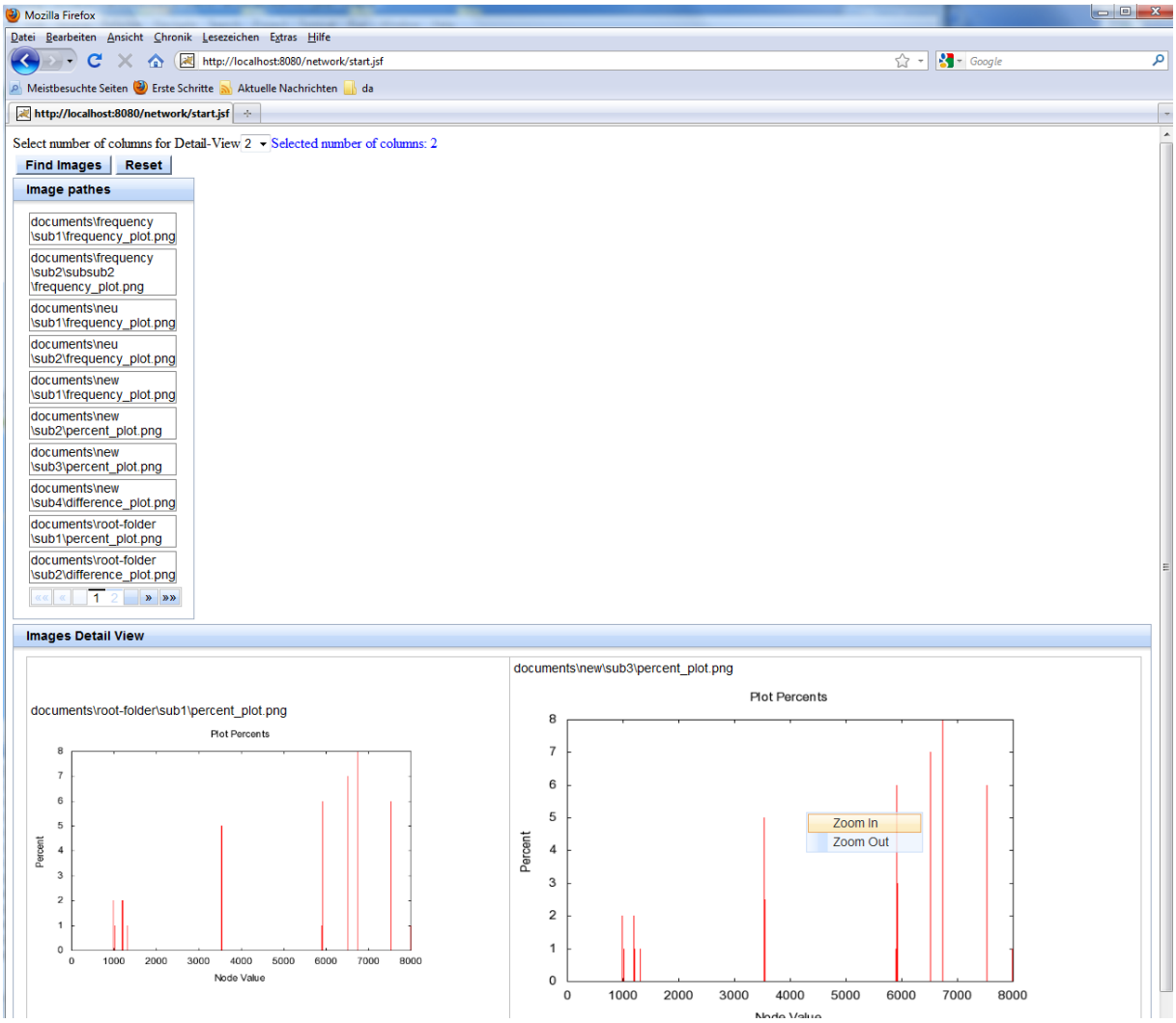

Abbildung 6.8: Überblick der Funktionalitäten der „View Statistics“-Seite.

Wie aus Abbildung 6.8 ersichtlich wird, stehen für den Benutzer folgende Funktionalitäten zur Verfügung:

Anzahl der Spalten für Detailansicht einstellen

Als DropDown-Feld wurde die Komponente <h:selectOneMenu>verwendet. Grundlage für die Werte ist die Liste „numberOfColumns“. Wird die Seite aufgerufen, bewirkt das den Aufruf der Methode „getNumberOfColumns“, welche die „SelectItem“-Elemente generiert und zur Liste hinzufügt (aktuell die Werte 2 bis 6). Ändert der Benutzer den Wert, wird der „valueChangeListener“ (die Methode „numberOfColumnsChanged“) aufgerufen, wodurch die Variable „selectedNumberOfColumns“ den vom Benutzer neu ausgewählten Wert erhält. Diese Variable wird via Unified Expression an den Wert „columns“ der Detailansicht gebunden.

```
<h:selectOneMenu id="numberColumns"
      valueChangeListener="#{imagesBean.numberOfColumnsChanged}"
      onchange="submit()">
         <f:selectItems value="#{imagesBean.numberOfColumns}" />
</h:selectOneMenu>
```

Ordner nach Bildern durchsuchen

Beim Klick auf den Button „Find Images“ wird die Funktion „findAllImages“ aufgerufen. Diese liest zuerst aus dem Property-File den Wurzelordner aus und übergibt diesen als Parameter an die Funktion „listDir“. Diese arbeitet wie folgt:

1. Alle Dateien, die sich im aktuellen Ordner befinden, werden in eine Liste gespeichert
2. Diese wird in einer Schleife iteriert
   (a) Handelt es sich beim aktuellen Element um einen Ordner, wird die Funktion „listDir“ mit diesem Ordner als Parameter aufgerufen (rekursiv)
   (b) Andernfalls wird überprüft, ob die Datei vom Typ „png“ ist
      i. Ist dies der Fall, wird der Pfad zur Datei [6] - sofern dieser noch nicht vorhanden ist - auf die Liste „imagePaths“ gelegt
      ii. Ist dies nicht der Fall wird die aktuelle Datei ignoriert

Alle Pfade dieser gefundenen Bilder werden in einer Tabelle angezeigt. Aus Gründen der Übersichtlichkeit habe ich die Anzahl von den Reihen auf 10 beschränkt und einen Seiten-Scroller (<rich:datascroller>) hinzugefügt (siehe Folgender Codeausschnitt).

```
<rich:panel style="width:200px">
   <f:facet name="header">
      <h:outputText value="Image pathes" />
   </f:facet>
   <h:dataTable id="images" columns="1" value="#{imagesBean.imagePaths}"
         var="img" footerClass="footerClass" rows="10">

      <h:column>
         <a4j:outputPanel style="width:160px;border:1px solid gray;
                  padding:1px" layout="block">

            <h:outputText value="#{img}"/>

            <rich:dragSupport dragType="item" dragValue="#{img}">
       <rich:dndParam name="label" value="#{img}" />

            </rich:dragSupport>

         </a4j:outputPanel>
      </h:column>
```

[6] Anm.: Um die Anzeige des Pfades etwas zu verkürzen, wird dieser nach dem Wurzelordner abgeschnitten. Dieser „Short-Path“ wird dem Benutzer angezeigt.

```
      <f:facet name="footer">
         <rich:datascroller id="datascroller" />
      </f:facet>

   </h:dataTable>

</rich:panel>
```

Die Komponente <rich:dragSupport>wird im Abschnitt „Statistiken betrachten“ auf Seite 89 eingehender beschrieben.

Reset

Diese Funktion dient dazu, alle gefundenen Statistiken bzw. alle aktuell in der Detailansicht vorhandenen Statistiken zu löschen. Ist die Seite überladen, soll der Anwender die Möglichkeit haben, diese neu aufzubauen.

Statistiken betrachten

Mancher Leser mag sich vielleicht schon gefragt haben, wie es überhaupt möglich ist, die Statistiken zu betrachten und warum es eigentlich diese Übersicht aller Pfade gibt. Grund dafür ist, dass ich das Hinzufügen zur Detailansicht mit „Drag&Drop“ umgesetzt habe. Die Tabelle („Image paths“) soll dem User eine Übersicht der vorhandenen Statistiken geben. Diese Pfade kann er einzeln in die Detailansicht ziehen, wo ihm dann das entsprechende Bild hinter dem Pfad angezeigt wird.

```
<rich:panel header="Images Detail View" id="dropPanel">

   <rich:dropSupport dropListener="#{imagesBean.processDrop}"
         acceptedTypes="item" reRender="dropArea">

      <rich:dataGrid id="dropArea"
         columns="#{imagesBean.selectedNumberOfColumns}" width="100%"
         value="#{imagesBean.imagesDetailView}" var="img">

         <h:outputText value="#{img}" />
         <h:graphicImage id="pic" value="#{img}" width="100%"/>
      </rich:dataGrid>
   </rich:dropSupport>

</rich:panel>
```

Auffällig hier ist die Komponente <rich:dropSupport>. Diese gehört unzertrennlich mit der zuvor erwähnten Komponente <rich:dragSupport>zusammen. Der Drag-Support ermöglicht das Verschieben der einzel-

nen Zeilen der Übersichtstabelle. Der aktuelle Wert wird dabei als Parameter mit übergeben. Der Drop-Support fängt diese Komponenten auf und ruft dabei eine entsprechende Methode auf. Wichtig dabei ist, dass der „dragType“ ident mit den „acceptedTypes“ des Drop-Supports ist, um verschobene Elemente auffangen zu können. Wurde ein Pfad in die Detailansicht gezogen, wird der „dropListener“ aufgerufen. Dieser beinhaltet im Wesentlichen nur die folgenden zwei Zeilen:

```
public void processDrop(DropEvent event) {

   if (!imagesDetailView.contains(event.getDragValue().toString())) {
      imagesDetailView.add(event.getDragValue().toString());
   }
}
```

Befindet sich also der aktuell verschobene Pfad noch nicht in der Detailansicht, wird dieser hinzugefügt und der Drop-Bereich neu gerendert.

Größe einzelner Statistiken verändern

Um dem Anwender mehr Flexibilität in Bezug auf die Betrachtung der Statistiken zu geben, wurde noch die Funktionalität zur Veränderung der Größe einzelner Bilder hinzugefügt. Durch einen Rechtsklick mit der Maus auf eine Statistik wird ein Kontextmenü mit den Einträgen „Vergrößern“ oder „Verkleinern“ eingeblendet.

```
<rich:contextMenu event="oncontextmenu" attachTo="pic"
   submitMode="none">

   <rich:menuItem value="Zoom In"
      onclick="enlarge(this.id)" id="zin"></rich:menuItem>

   <rich:menuItem value="Zoom Out"
   onclick="decrease(this.id);" id="zout"></rich:menuItem>
</rich:contextMenu>
```

Wie im Code ersichtlich, werden beim Klick auf einen Menüeintrag JavaScript-Funktionen aufgerufen. Der folgende Codeausschnitt zeigt die Funktion zum Verkleinern der Statistik:

```
function decrease(menuid){
  var image_id = menuid.replace(/zout/g, "pic");

  document.getElementById(image_id).width=document.getElementById(image_id).
  width*0.9;

  document.getElementById(image_id).height=document.getElementById(image_id).
```

```
  height*0.9;
}
```

Weil IDs immer eindeutig sein müssen, bestehen diese aus dem kompletten Pfad inklusive den IDs derer Eltern usw. Die ID des Kontextmenüs der ersten Zeile und ersten Spalte in der Tabelle wäre z.B. “form:j_id32:dropArea:0:zout“. Das Kontextmenü der Statistik in der ersten Zeile und zweiten Spalte hätte dann die ID „form:j_id32:dropArea:1:zout“.

Je nachdem von welchem Bild aus das Kontextmenü aufgerufen wird, besitzt dieses eine andere ID. Die JavaScript Funktion generiert ausgehend von der Menü-ID die passende Image-ID für das zu verändernde Bild → „zout“ bzw. „zin“ wird ersetzt durch „pic“. Danach wird die zu verändernde Statistik von der Seite ausgelesen und dessen Höhe und Breite verkleinert bzw. vergrößert.

Kapitel 7

Zusammenfassung und Schlussfolgerung

Das Ziel dieser Masterarbeit war die Implementierung eines Tools, welches dem Anwender die Möglichkeit bietet, von ihm generierte Data-Files über eine Webapplikation auf den Server zu laden und daraus Statistiken (Plots) zu generieren. Diese Statistiken sollten über gnuplot generiert werden und die Webapplikation sollte in Java geschrieben sein. Neben der Generierung war die Darstellung aller am Server liegenden Statistiken ein wichtiger Aspekt, um dem Benutzer die Möglichkeit zu geben, Statistiken miteinander zu vergleichen. Die Analyse einer Vielzahl von Daten und Informationen sollte über diese Webanwendung auf mehrere Benutzer verteilt werden.

Im Moment gibt es nur die Möglichkeit, drei unterschiedliche Arten von Statistiken („Frequency“, „Percent“, „Difference“) zu generieren. Wichtig beim Konzept der Arbeit war, einen Ansatz zu finden, welcher ein einfaches Hinzufügen neuer Statstiken ermöglicht. Dazu wurde eine abstrakte Statistik-Klasse angelegt, welche die Grundeinstellungen beinhaltet. Soll eine neue zweidimensionale Statistik hinzugefügt werden, muss lediglich von der abstrakten Statistik abgeleitet und die spezifischen Einstellungen in der konkreten Statistik, gesetzt werden. Um dreidimensionale Statistiken zu generieren, muss von der „Plot“ Klasse abgeleitet werden. Die neue Klasse muss um die dreidimensionalen Einstellungen ergänzt werden (z.B. zlabel, zrange, usw.).

Um der Aufgabenstellung gerecht zu werden, wurde zu Beginn der Arbeit Recherche über die Themen World Wide Web, Statistiken und Visualisierung von Daten angestellt. Es konnte dabei aufgezeigt werden, wie komplex die Analyse von großen Datensätzen werden kann. Dazu zählt sowohl die graphische Darstellung, um aus dem „Datenwald“ Informationen herauslesen zu können, aber auch die statistische Analyse der Daten, um Gesetzmäßigkeiten zu finden und Wissen zu generieren.

Nach den Anforderungen an die Applikation und den theoretischen Grundlagen fokussierte ich die Arbeit auf die Tools, die Teil der praktischen Umsetzung waren. Die Graphen sollten mit gnuplot generiert werden und es sollte sich um eine Java-Webapplikation handeln.

Schnell erkannte ich die Komplexität von gnuplot. Es gab eine riesige Anzahl von Befehlen, Einstellungsmöglichkeiten und Möglichkeiten zur Generierung eines Plots. Nach intensiver Bearbeitung und Auseinandersetzung mit dem Thema „gnuplot“, musste das Design zur Integrierung von gnuplot in das Projekt, festgelegt werden.

Der Benutzer übergibt seine Input-Daten an die Applikation. Ausgehend von den Einstellungen zur Ge-

nerierung des Plots und der Inputdatei wird eine temporäre Plot-Datei erzeugt, die sämtliche gnuplot-Befehle beinhaltet und diese über einen Prozess an gnuplot übergibt. Um der Anforderung der Erweiterbarkeit im Bereich der Statistiken gerecht zu werden, gibt es eine abstrakte Statistik, die Eigenschaften beinhaltet, welche alle Statistiken gemeinsam haben, wie etwa das Zeichnen der Statistik. Davon abgeleitet gibt es die jeweiligen konkreten Statistiken, welche die Statistik spezifischen Eigenschaften beinhalten, wie etwa Beschriftung, usw.

Im Bereich der Webapplikation wurden im Zuge der Recherche einige Frameworks zur Umsetzung aufgelistet. Danach habe ich mich auf Grund einiger Vorteile (MVC, AJAX-Unterstützung, viele vordefinierte Komponenten, etc.) für RichFaces mit JSF entschieden. Dieses wurde anschließend im Detail inklusive einiger Beispiele beschrieben.

Im letzten Teil der Masterarbeit wurde die Umsetzung der Webapplikation beschrieben. Diese beinhaltet Screenshots der Benutzeroberfläche und die von mir eingesetzten Komponenten sowie detaillierte Beschreibungen zu den wichtigsten Funktionen der Applikation (FileUpload, etc.). Der letzte Punkt war die Verknüpfung der Webapplikation mit der gnuplot-Schnittstelle. Bei der Entwicklung des Frontends war das Thema „Usability“ sehr wichtig. Da nicht nur die Dateien hochgeladen werden können, sondern auch das Ordnersystem (ausgehend von einem spezifizierten Ordner) angezeigt und bearbeitet werden kann, war es mir sehr wichtig, immer nur diese Komponenten für den Benutzer anzuzeigen, die er im Moment auch tatsächlich benutzen darf. So kann beispielsweise eine Datei nur hochgeladen werden, wenn zuvor ein Ordner ausgewählt wurde, in dem diese gespeichert und die Statistik generiert werden soll.

Laut Anforderungen war das Ziel des Projekts, ein Grundgerüst zu schaffen, welches ein einfaches Hinzufügen neuer Statstiken ermöglicht. Das Grundgerüst ist fertig. Aktuell ist die Funktionalität der Statistikgenerierung allerdings dahingehend eingeschränkt, dass lediglich ein Plot für den ganzen Wertebereich der Inputdaten erzeugt werden kann („autoscale“). Besteht die Inputdatei aus sehr vielen Daten, kann es passieren, dass der Benutzer die gewünschten Informationen nicht herauslesen kann.

Durch die Arbeit wurde verdeutlicht, wie komplex die statistische Analyse sowie die Visualisierung von großen Datenmengen sein können. Diese Themen wurden im praktischen Teil außer Acht gelassen und bieten noch viel Platz für Erweiterungen und Verbesserungen.

Eine Querverbindung zu diesem Projekt könnte später das Wissensmanagement werden. Ist das Projekt so weit, dass nicht nur Statistiken generiert, sondern diese auch optimal visualisiert und statistisch analysiert werden können, bietet sich die Möglichkeit, daraus Wissen zu generieren. Dieses Wissen könnte dann mit Wissensmanagement-Systemen interagieren.

Anhang A

Installation der Entwicklungsumgebung

Prinzipiell wäre es möglich mit einem gewöhnlichen Texteditor eine RichFaces Applikation zu entwickeln. Es gibt aber zahlreiche Plugins für unterschiedliche IDEs, die dem Programmierer Arbeit abnehmen können bzw. alles etwas übersichtlicher darstellen. Aus diesen Gründen habe ich mich bei der Entwicklung meines Projekts für Eclipse mit einigen unterstützenden Plugins entschieden. Dieser Abschnitt soll zeigen, was ich vor dem eigentlichen Start meines Projektes installiert habe.

A.1 Eclipse

Wie zuvor erwähnt, wurde Eclipse als Entwicklungsumgebung zur Umsetzung des Projekts gewählt. Vor der Installation muss allerdings sichergestellt sein, dass ein JDK (Java Development Kit) am Rechner installiert ist. Eine detaillierte Anleitung zur Installation des JDK ist auf [Horn, 2008] zu finden.

Wurde diese erfolgreich durchgeführt, kann Eclipse installiert werden. Dabei sollte unbedingt darauf geachtet werden, dass die IDE für Java EE Entwickler (Eclipse WTP) ausgewählt wird, da diese bereits einige Plugins zur Entwicklung von Web-Applikationen beinhaltet und sie somit nicht nachträglich installiert werden müssen. Eclipse kann unter `http://www.eclipse.org` heruntergeladen werden. In diesem Fall gibt [Horn, 2010] eine hilfreiche Unterstützung beim Installationsprozess.

A.2 JBossTools

Als zusätzliches Plugin wurde JBossTools installiert, welche unter `http://www.jboss.org/tools/download` heruntergeladen werden können. Dieses Plugin stellt u.a. einen grafischen Editor zur Verfügung, der dem Entwickler Arbeit abnimmt. Der Editor unterstützt beispielsweise eine Vorschau der aktuellen Seite, eine automatische Vervollständigung des Codes und eine Toolbox mit den von RichFaces und JSf angebotenen Komponenten. Um detaillierte Informationen zur Installation zu erhalten sei an dieser Stelle auf [JBoss Community] verwiesen. Zu beachten ist, dass die JBossTools Version mit der verwendeten Eclipse Installation kompatibel ist. Auf der Seite `http://www.jboss.org/tools/download` findet sich eine Übersicht über die komplette Version.

A.3 Tomcat

Als lokaler Server wurde Tomcat installiert. Heruntergeladen kann dieser unter `http://tomcat.apache.org/` werden. Zur Installation gibt es zwei verschiedene Möglichkeiten:

Wurde die Archiv-Version heruntergeladen, muss diese ins Dateisystem entpackt werden. Die zweite Möglichkeit unter Windows ist eine Installationsdatei. Nach der Installation von Tomcat muss noch die Umgebungsvariable CATALINA_HOME gesetzt werden, welche auf die Wurzel des Tomcat-Ordners zeigen muss [Markgraf, 2008].

Wurde die Archivdatei heruntergeladen, muss noch die Datei „tomcat-users.xml“ angepasst werden, um auf die Administratorkonsole und die Administratorseite von Tomcat zugreifen zu können [Markgraf, 2008]. Das folgende Beispiel zeigt, wie diese „tomcat-users.xml“ aussehen könnte:

```
< ?xml version='1.0' encoding='utf-8'?>
< tomcat-users>
< role rolename="manager"/>
< role rolename="admin"/>
< user username="webadmin" password="webpassword" roles="admin,manager"/>
< /tomcat-users>
```

Gestartet bzw. gestoppt wird der Server über die entsprechenden Batch-Dateien, die sich im Ordner „CATALINA_HOME/bin“ befinden. Wurde die Installation erfolgreich durchgeführt, zeigt der Aufruf von `http://localhost:8080` im Browser die Tomcat-Startseite an. Über den Aufruf der Seite `http://localhost:8080/manager/html` wird nach erfolgreicher Eingabe von Benutzername und Passwort der Tomcat Web-Application-Manager angezeigt. In diesem ist es möglich, Web-Anwendungen zu laden (deploy) oder wieder zu löschen (undeploy).

A.4 Tomcat Eclipse Plugin

Um die zuvor erwähnten Vorgänge (Server starten, Web-Applikation hochladen, Server stoppen, etc.) nicht immer „per Hand“ erledigen zu müssen, gibt es für Tomcat ein Eclipse-Plugin. Dieses unterstützt laut [eclipsetotale.com, 2011] folgende Befehle:

- Starten und Stoppen von Tomcat
- Java-Projekte zum Tomcat-Klassenpfad hinzufügen
- Ein WAR (Web Application Archive)-Projekt erzeugen
- Exportieren eines Tomcat-Projekts in eine WAR-Datei

Um das Plugin zu verwenden, muss dieses von [eclipsetotale.com, 2011] heruntergeladen und in den „Plugins“-Ordner der Eclipse-Installation extrahiert werden. Nach einem Neustart von Eclipse ist das Tomcat-Plugin bereits aktiviert. Danach müssen noch folgende Grundeinstellungen vorgenommen werden, um damit arbeiten zu können:

Die Werte „Tomcat-HOME“ und „Tomcat-VERSION“ in Eclipse sind unter „Workbench –>Preferences –>Tomcat“ zu setzen. Das Plugin greift auf die Standard JRE zu. Damit der Server fehlerfrei funktioniert, muss unter „Window –>Preferences –>Java –>Installed JREs“ eine JDK angegeben werden [eclipsetotale.com, 2011].

Anhang B

Erstellung eines RichFaces-Projekts in Eclipse

Laut [jboss.org, 2010, Seite 5ff] gibt es zwei Möglichkeiten in Eclipse ein Projekt zu erstellen, dass RichFaces verwendet:

1. Manuell (einige Konfigurationsschritte durch den Entwickler sind erforderlich)
2. Automatisiert mit Maven[1] (erfordert Maven Kenntnisse)

B.1 Manuelle Erstellung des Projekts

Diese Methode ist für denjenigen zu bevorzugen, dessen Maven-Kenntnisse sich in Grenzen halten. Laut [jboss.org, 2010, Seite 5ff] gibt es einige Schritte, die abzuarbeiten sind, um RichFaces erfolgreich in einem Eclipse-Projekt zu verwenden:

1. Herunterladen von RichFaces
   (a) Die Releases von RF stehen zum Download unter `http://www.jboss.org/richfaces/download.html` bereit.
   (b) Eine Version herunterladen und entpacken. Ich empfehle zum jetzigen Zeitpunkt die Verwendung der Version JSF 1.2 und RichFaces 3.3.3, da Version 4 noch in den Kinderschuhen steckt und im RF-Forum schon einiges über Probleme mit dieser Version gepostet wurde.
2. Erstellung des Projekts
   (a) Das Projekt ist ein Standard JSF 1.2-Projekt
   (b) Voraussetzung für dessen Erstellung ist eine vorhandene JBoss-Tools-Installation (siehe Abschnitt A.2) in Eclipse –>nur dann ist es möglich, ein JSF Projekt zu erstellen.
3. RichFaces-Bibliotheken im Projekt integrieren

[1] `http://maven.apache.org/`

(a) Im zuvor entpackten RF-Ordner gibt es einen Unterordner „lib“. Dieser beinhaltet alle drei benötigten RichFaces-Bibliotheken - API, UI und Implementation.

(b) Diese drei Dateien sind ins Projekt in den Ordner „WEB-INF/lib“ zu kopieren.

(c) Zu beachten ist noch, dass noch einige andere Bibliotheken unbedingt im Projekt eingebunden sein müssen, wie etwa:

i. commons-beanutils-1.7.0.jar
ii. commons-collections-3.2.jar
iii. commons-digester-1.8.jar
iv. commons-logging-1.0.4.jar
v. jhighlight-1.0.jar

Es ist umbedingt im „lib“-Ordner des Projekts zu kontrollieren, welche dieser Bibliotheken bereits integriert sind (etwa durch die Erstellung des JSF Projekts) und welche noch fehlen. Die nicht vorhandenen Libraries einfach im Internet suchen, herunterladen und ins Projekt integrieren . Die folgenden Schritte beinhalten bereits die erste Test-Seite inklusive der vorzunehmenden Konfigurationen. Zur Veranschaulichung möchte ich hier auch die einzelnen Beispiel-Dateien zeigen.

4. RichFaces in der web.xml registrieren

   Hier ein Beispiel, wie die web.xml aussehen könnte. Unbedingt notwendig ist der Filter und das Servlet bzw. Servlet Mapping.

```
<?xml version="1.0"?>
<web-app version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

<display-name>Greeter</display-name>

<context-param>
   <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
   <param-value>server</param-value>
</context-param>

<context-param>
   <param-name>org.richfaces.SKIN</param-name>
   <param-value>blueSky</param-value>
</context-param>

<context-param>
   <param-name>org.richfaces.CONTROL_SKINNING</param-name>
   <param-value>enable</param-value>
</context-param>

<filter>
   <display-name>RichFaces Filter</display-name>
   <filter-name>richfaces</filter-name>
   <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>
```

```
<filter-mapping>
   <filter-name>richfaces</filter-name>
   <servlet-name>Faces Servlet</servlet-name>
   <dispatcher>REQUEST</dispatcher>
   <dispatcher>FORWARD</dispatcher>
   <dispatcher>INCLUDE</dispatcher>
</filter-mapping>

<listener>
   <listener-class>com.sun.faces.config.ConfigureListener
   </listener-class>
</listener>

<!-- Faces Servlet -->
<servlet>
   <servlet-name>Faces Servlet</servlet-name>
   <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
   <load-on-startup>1</load-on-startup>
</servlet>

<!-- Faces Servlet Mapping -->
<servlet-mapping>
   <servlet-name>Faces Servlet</servlet-name>
   <url-pattern>*.jsf</url-pattern>
</servlet-mapping>

</web-app>
```

[jboss.org, 2010, Seite 6ff]

5. Managed Beans erzeugen (siehe folgendes Beispiel)

```
package demo;

public class user {

   private String name="";

   public String getName() {
      return name;
   }

   public void setName(String name) {
   this.name = name;
   }
}
```

[jboss.org, 2010, Seite 8]

6. Bean in der faces-config.xml registrieren

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config version="1.2"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">

<managed-bean>
   <description>UsernName Bean</description>
   <managed-bean-name>user</managed-bean-name>
   <managed-bean-class>demo.user</managed-bean-class>
   <managed-bean-scope>request</managed-bean-scope>
   <managed-property>
      <property-name>name</property-name>
      <property-class>java.lang.String</property-class>
      <value/>
   </managed-property>
</managed-bean

</faces-config>
```

[jboss.org, 2010, Seite 8f]

In der „faces-Config.xml“ werden auch die Navigationsregeln definiert. Diese sind hier nicht inkludiert, da dieses kurze Beispiel nur eine Basis darstellen und einen Überblick vermitteln soll.

7. Im Ordner „Web Content“ eine Seite anlegen (Bsp.: index.jsp)

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<!-- RichFaces tag library declaration -->
<%@ taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
<%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>

<html>

<head>
   <title>RichFaces Greeter</title>
</head>

<body>
   <f:view>
      <a4j:form>
         <rich:panel header="RichFaces Greeter" style="width: 315px">
            <h:outputText value="Your name: " />
```

```
            <h:inputText value="#{user.name}" >
               <f:validateLength minimum="1" maximum="30" />
            </h:inputText>

            <a4j:commandButton value="Get greeting"
               reRender="greeting" />

            <h:panelGroup id="greeting" >
               <h:outputText value="Hello, "
                  rendered="#{not empty user.name}" />
               <h:outputText value="#{user.name}" />
               <h:outputText value="!"
                  rendered="#{not empty user.name}" />
            </h:panelGroup>

         </rich:panel>
      </a4j:form>
   </f:view>
</body>

</html>
```

[jboss.org, 2010, Seite 9]

8. Das Projekt am „Tomcat“ starten

   (a) Bei den Projekt Einstellungen müssen im Reiter „Tomcat“ einige Einstellungen vorgenommen werden, um die Applikation über das bereits installierte Tomcat-Plugin zu starten:

      i. „Ist ein Tomcat-Projekt“ muss angehakt werden
      ii. Die „Anwendungs-URI“ für den Aufruf im Browser ist zu setzen (Bsp.: /greeter)
      iii. „Unterverzeichnis, das als Root verwendet werden soll“ auf „/WebContent“ setzen

   (b) Über `http://localhost:8080/greeter/index.jsf` kann nun die Index-Seite des Projekts aufgerufen werden

B.2 Automatisierte Erstellung des Projekts

Wie bereits erwähnt, sollte diese Methode nur verwendet werden, wenn Kenntnisse über Maven vorhanden sind. Diese Methode wird hier nicht detaillierter beschrieben. Leser, die sich für diese Art der Projekt-Erzeugung interessieren, seien an dieser Stelle auf den Developer Guide von RichFaces [jboss.org, 2010, Seite 10ff] verwiesen. Dort ist eine schrittweise Anleitung für die automatisierte Erstellung des Projekts mit Maven zu finden.

Anhang C

CD-ROM

Literaturverzeichnis

[Anand 2008] ANAND: *JavaServer Faces (JSF) Tutorial*. 01 2008. – URL http://www.developersbook.com/jsf/jsf-tutorials/jsf-tutorials.php. – Zugriffsdatum: 23.12.2010 (Zitiert auf den Seiten vii, 51, 56, 57 und 58.)

[Anderson 2007] ANDERSON, Paul: What is Web 2.0: Ideas, technologies and implications for education / JISC. URL http://www.jisc.ac.uk/media/documents/techwatch/tsw0701b.pdf, 2007. – Forschungsbericht (Zitiert auf den Seiten 10 und 11.)

[Beal 2011] BEAL, Vangie: *The Difference Between the Internet and World Wide Web*. 01 2011. – URL http://www.webopedia.com/DidYouKnow/Internet/2002/Web_vs_Internet.asp. – Zugriffsdatum: 08.02.2011 (Zitiert auf Seite 3.)

[Berners-Lee] BERNERS-LEE, Sir T.: *Longer Biography*. – URL http://www.w3.org/People/Berners-Lee/Longer.html. – Zugriffsdatum: 08.02.2011 (Zitiert auf den Seiten 4 und 10.)

[Berners-Lee und Fischetti 2000] BERNERS-LEE, Tim ; FISCHETTI, Mark: *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. 1. Harper San Francisco, 2000. – 246 S. – ISBN 978-0062515872 (Zitiert auf den Seiten 1, 4, 5, 6, 7, 8, 9, 10 und 11.)

[Berners-Lee et al. 2001] BERNERS-LEE, Tim ; HENDLER, James ; LASSILA, Ora: The Semantic Web. In: *Scientific American* 284 (2001), Mai, Nr. 5, S. 34–43. – URL http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 (Zitiert auf den Seiten 11 und 12.)

[Bosch 2004] BOSCH, Andy: *Java Server Faces*. Addison-Wesley, 2004. – 456 S (Zitiert auf den Seiten vii, 47 und 48.)

[Burns et al. 2010] BURNS, Ed ; SCHALK, Chris ; GRIFFIN, Neil: *JavaServer Faces 2.0, The Complete Reference*. Mcgraw-Hill Professional, 02 2010. – 752 S (Zitiert auf Seite 51.)

[CERN 2008] CERN: *The website of the world's fierst-ever web server*. 2008. – URL http://info.cern.ch/. – Zugriffsdatum: 08.02.2011 (Zitiert auf den Seiten vii und 7.)

[Crawford. 2010] CRAWFORD., Dick: *gnuplot 4.4 An Interactive Plotting Program*. 05 2010. – URL http://www.gnuplot.info/docs_4.4/gnuplot.pdf. – Zugriffsdatum: 14.12.2010 (Zitiert auf den Seiten 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44 und 69.)

[Durocher 2005] DUROCHER, Marc: *Einführung in Javaserver Faces*. 12 2005. – URL http://www.zdnet.de/anwendungsentwicklung_einfuehrung_in_javaserver_faces_story-20000201-39139209-1.htm. – Zugriffsdatum: 22.12.2010 (Zitiert auf den Seiten 47 und 48.)

[Dutter 2010] DUTTER, Rudolf: *Statistik - Überblick und Definition*. 02 2010. – URL http://www.statistik.tuwien.ac.at/public/dutt/vorles/inf_bak/node6.html. – Zugriffsdatum: 30.01.2011 (Zitiert auf Seite 13.)

[e-teaching.org 2008] E-TEACHING.ORG: *Semantic Web*. 2008. – URL http://www.e-teaching.org/technik/vernetzung/semanticweb/. – Zugriffsdatum: 09.02.2011 (Zitiert auf Seite 12.)

[eclipsetotale.com 2011] ECLIPSETOTALE.COM: *Sysdeo Eclipse Tomcat Launcher plugin*. 2011. – URL http://www.eclipsetotale.com/tomcatPlugin.html. – Zugriffsdatum: 04.01.2011 (Zitiert auf den Seiten 96 und 97.)

[Elsner 2000] ELSNER, Frank: *Einführung in gnuplot-ein interaktives Programm zur grafischen Darstellung von Datenpunkten und math. Kurven und Flächen*. 03 2000. – URL http://www.rz.uni-osnabrueck.de/Zum_Nachlesen/Skripte_Tutorials/Gnuplot_Einfuehrung/pdf/gnuplot.pdf. – Zugriffsdatum: 14.12.2010 (Zitiert auf Seite 31.)

[empulse 2010] EMPULSE: *Rich Internet Applications & Ajax*. 2010. – URL http://www.empulse.de/2005/07/27/rich-internet-applications-ajax. – Zugriffsdatum: 04.01.2011 (Zitiert auf Seite 63.)

[Gavin 2008] GAVIN, Henri P.: *GNUPLOT 4.2 - A Brief Manual and Tutorial*. 12 2008. – URL http://www.duke.edu/~hpgavin/gnuplot.html. – Zugriffsdatum: 14.12.2010 (Zitiert auf Seite 32.)

[Hitzler et al. 2007] HITZLER, Pascal ; KRÖTZSCH, Markus ; SURE, York ; RUDOLPH, Sebastian: *Semantic Web: Grundlagen*. Springer, Oktober 2007. – ISBN 9783540339939 (Zitiert auf Seite 12.)

[Horn 2008] HORN, Torsten: *Installation des Java SDK / JDK*. 2008. – URL http://www.torsten-horn.de/techdocs/java-install.htm. – Zugriffsdatum: 04.01.2011 (Zitiert auf Seite 95.)

[Horn 2010] HORN, Torsten: *Eclipse: Erste Schritte*. 2010. – URL http://www.torsten-horn.de/techdocs/java-eclipse.htm. – Zugriffsdatum: 04.01.2011 (Zitiert auf Seite 95.)

[Intertech 2008] INTERTECH: *Component Tree*. 2008. – URL http://www.jsf-training-guide.com/component-tree.html. – Zugriffsdatum: 03.01.2011 (Zitiert auf den Seiten vii und 52.)

[JBoss Community] JBOSS COMMUNITY: *Eclipse Plugins for JBoss technology*. – URL http://www.jboss.org/tools/download/installation.html. – Zugriffsdatum: 04.01.2011 (Zitiert auf Seite 95.)

[jboss.org 2009] JBOSS.ORG: *RichFaces CDK Developer Guide*. 2009. – URL http://docs.jboss.org/richfaces/latest_3_3_X/en/cdkguide/html_single/. – Zugriffsdatum: 04.01.2011 (Zitiert auf Seite 64.)

[jboss.org 2010] JBOSS.ORG: *RichFaces Developer Guide*. 05 2010. – URL http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/pdf/richfaces_reference.pdf. – Zugriffsdatum: 04.01.2011 (Zitiert auf den Seiten 64, 74, 78, 81, 99, 101, 102 und 103.)

[Jürgen Hedderich 2006] JÜRGEN HEDDERICH, Lothar S.: *Angewandte Statistik. Methodensammlung mit R*. 12. Springer, 08 2006. – 702 S. – ISBN 978-3540321606 (Zitiert auf den Seiten 13, 14, 15 und 16.)

[Katz 2008] KATZ, Max: *Practical RichFaces*. Apress, 2008. – 245 S (Zitiert auf den Seiten 47, 48, 61, 63 und 64.)

[Keim 2002] KEIM, D.A.: Information visualization and visual data mining. In: *Visualization and Computer Graphics, IEEE Transactions on* 8 (2002), Nr. 1, S. 1 –8. – ISSN 1077-2626 (Zitiert auf den Seiten 1, 17 und 18.)

[Kurz et al. 2009] KURZ, Michael ; MARINSCHEK, Martin ; SCHNABL, Andrea ; MÜLLAN, Gerald: *Einführung in JavaServer Faces*. 11 2009. – URL http://jsfatwork.irian.at/semistatic/introduction.html. – Zugriffsdatum: 28.02.2011 (Zitiert auf den Seiten vii und 50.)

[Lim 2007] LIM, Ivan: *Java Bean Scopes in JSF*. 2007. – URL http://www.java-samples.com/showtutorial.php?tutorialid=472. – Zugriffsdatum: 03.01.2011 (Zitiert auf Seite 53.)

[Lohninger 2010] LOHNINGER, Hans: *Schließende Statistik*. 11 2010. – URL http://www.statistics4u.info/fundstat_germ/cc_inferential_stat.html. – Zugriffsdatum: 31.01.2011 (Zitiert auf Seite 16.)

[Mann 2004] MANN, K.D.: *JavaServer faces in action*. Manning, 2004 (In Action). – ISBN 9781932394122 (Zitiert auf Seite 51.)

[Marinschek et al. 2007] MARINSCHEK, Martin ; SCHNABL, Andrea ; MÜLLAN, Gerald: *JSF@Work*. dpunkt Verlag, 2007. – 325 S (Zitiert auf den Seiten 49, 50, 51, 52, 53, 54, 55, 57, 58 und 59.)

[Markgraf 2008] MARKGRAF, Mike: *Tomcat 6 installieren*. 01 2008. – URL http://opentutorial.blogspot.com/2008/01/tomcat-6-installieren.html. – Zugriffsdatum: 04.01.2011 (Zitiert auf Seite 96.)

[McGraw-Hill 2008] MCGRAW-HILL, Osborne: *Introduction to JavaServer Faces, Part 1*. 03 2008. – URL http://www.devshed.com/c/a/Java/Introduction-to-JavaServer-Faces-1/. – Zugriffsdatum: 03.01.2011 (Zitiert auf Seite 59.)

[Microsystems 2000] MICROSYSTEMS, Sun: *What is a Tag Library?* 2000. – URL http://java.sun.com/products/jsp/tutorial/TagLibrariesTOC.html. – Zugriffsdatum: 03.01.2011 (Zitiert auf Seite 54.)

[Myfaces Wiki 2009] MYFACES WIKI: *Performance*. 2009. – URL http://wiki.apache.org/myfaces/Performance. – Zugriffsdatum: 03.01.2011 (Zitiert auf Seite 59.)

[NETPLANET] NETPLANET: *Gopher*. – URL http://www.netplanet.org/dienste/gopher.shtml. – Zugriffsdatum: 08.02.2011 (Zitiert auf Seite 10.)

[NIST 2003] NIST: *Engineering Statistics Handbook*. 06 2003. – URL http://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm. – Zugriffsdatum: 30.01.2011 (Zitiert auf Seite 14.)

[Ferreira de Oliveira und Levkowitz 2003] OLIVEIRA, M.C. Ferreira de ; LEVKOWITZ, H.: From visual data exploration to visual data mining: a survey. In: *Visualization and Computer Graphics, IEEE Transactions on* 9 (2003), Nr. 3, S. 378 – 394. – ISSN 1077-2626 (Zitiert auf Seite 18.)

[Palmer 2001] PALMER, Sean B.: *The Semantic Web: An Introduction*. 09 2001. – URL http://infomesh.net/2001/swintro/. – Zugriffsdatum: 09.02.2011 (Zitiert auf Seite 11.)

[Patel 2009] PATEL, Viral: *JSF Validation Tutorial: Error Handling in JSF*. 03 2009. – URL http://java.dzone.com/articles/jsf-validation-tutorial-error. – Zugriffsdatum: 03.01.2011 (Zitiert auf Seite 53.)

[Paulson 2005] PAULSON, L.D.: Building rich web applications with Ajax. In: *Computer* 38 (2005), Nr. 10, S. 14 – 17. – ISSN 0018-9162 (Zitiert auf den Seiten vii, 61, 62 und 63.)

[Pruscha 2005] PRUSCHA, Helmut: *Statistisches Methodenbuch: Verfahren, Fallstudien, Programmcodes*. 1. Springer, 07 2005. – 412 S. – ISBN 978-3540260066 (Zitiert auf Seite 13.)

[RichFaces] RICHFACES: *RichFaces Live Demo*. – URL http://livedemo.exadel.com/richfaces-demo/richfaces/tabPanel.jsf. – Zugriffsdatum: 04.01.2011 (Zitiert auf den Seiten 64 und 78.)

[Semantic Focus 2007] SEMANTIC FOCUS: *Introduction to the Semantic Web Vision and Technologies - Part 3 - The Resource Description Framework*. 12 2007. – URL http://www.semanticfocus.com/. – Zugriffsdatum: 09.02.2011 (Zitiert auf den Seiten vii und 12.)

[SemiByte 2010] SEMIBYTE: *GnuPlot*. 2010. – URL http://pubs.cs.uct.ac.za/archive/00000275/01/www_2005_qti_final_edited_hs.pdf. – Zugriffsdatum: 15.12.2010 (Zitiert auf den Seiten ix, 32, 33 und 36.)

[Shan und Hua 2006] SHAN, Tony C. ; HUA, Winnie W.: Taxonomy of Java Web Application Frameworks. In: *e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on*, 2006, S. 378 –385 (Zitiert auf den Seiten vii, 24, 26 und 29.)

[Sourceforge.net] SOURCEFORGE.NET: *jgnuplot*. – URL http://jgnuplot.sourceforge.net/. – Zugriffsdatum: 20.12.2010 (Zitiert auf den Seiten 67, 69 und 70.)

[Tim O'Reilly 2005] TIM O'REILLY, Patrick H.: *Was ist Web 2.0?* 2005. – URL http://www.oreilly.de/artikel/web20_trans.html. – Zugriffsdatum: 09.02.2011 (Zitiert auf Seite 11.)

[TUBraunschweig 2008] TUBRAUNSCHWEIG: *Wichtige Gnuplot-Befehle*. 2 2008. – URL http://www.pci.tu-bs.de/agelstner/lehre/thc2-08/gnuplot-befehle-small.pdf. – Zugriffsdatum: 15.12.2010 (Zitiert auf Seite 33.)

[Tukey 1977] TUKEY, John W.: *Exploratory Data Analysis*. Addison Wesley Pob Co Inc, 06 1977. – 688 S. – ISBN 978-0201076165 (Zitiert auf Seite 14.)

[Udo Bankhofer 2007] UDO BANKHOFER, Jürgen V.: *Datenanalyse und Statistik: Eine Einführung für Ökonomen im Bachelor*. 1. Gabler, 11 2007. – 328 S. – ISBN 978-3834904348 (Zitiert auf den Seiten vii, 14, 15 und 16.)

[Vogel 2005] VOGEL, Friedrich: *Beschreibende und schließende Statistik: Formeln, Definitionen, Erläuterungen, Stichwörter und Tabellen*. 13. Oldenbourg, 03 2005. – 372 S. – ISBN 978-3486577761 (Zitiert auf Seite 13.)

[Walter 2008] WALTER, Hans-Dirk: Rich Internet Applications Eine perfekte Kombination benutzerfreundlicher Schnittstellen mit Webtechnologie. In: *Informatik-Spektrum* 31 (2008), S. 333–342 (Zitiert auf Seite 61.)