university of applied sciences
gegr. 1970 fachhochschule hamburg
FACHBEREICH ELEKTROTECHNIK UND INFORMATIK

Diplomarbeit

Matthias Stolt

Ein Verwaltungssystem für mittelgroße Websites auf Basis von Open Source Komponenten

Studiengang Technische Informatik
Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. rer. nat. Christoph Klauck
Abgegeben am 28. Januar 2002

Berliner Tor 3 · D-20099 Hamburg

Ein Verwaltungssystem für mittelgroße Websites auf Basis von Open Source Komponenten

Stichworte
Content Management, Verwaltungssystem, Website, datenbankgestütztes Publizieren, Framework, Publizieren, CMS, PHP, Open Source.

Zusammenfassung
Es wird zunächst eine Begriffsklärung rund um Websites und deren Informationsarchitektur gegeben. An Hand der entwickelten Begriffe werden die Anforderungen an ein an Kundenwünsche anpassbares Verwaltungssystem für Websites definiert und diese in einem Framework als Baukastensystem realisiert.

Das Framework ist objektorientiert und basiert auf der serverseitigen Open Source Applikationssprache *PHP*. Die Inhalte der Websites werden in SQL-Datenbanken gehalten. Ein Schwerpunkt im Design des Frameworks liegt auf der Trennung von Funktionalität, Inhalt und Gestaltung/Layout, sowie der Wiederverwendbarkeit von entwickelten Anwendungen. Die Administration der Inhalte der Sites erfolgt browserbasiert.

A System for the Management of middle-sized Websites based upon Open Source Components

Keywords
Content management, administration system, website, database driven publishing, framework, publishing, CMS, PHP, open source.

Abstract
Introductory an explanation of the terms around the field of websites and their information architecture will be given. Based on these terms the requirements for a custom-made administration system for websites are developed and a framework as modular construction kit will be implemented.

The framework is object oriented and based upon the serverside application language *PHP* which is an open source project. The content of the sites is stored in SQL databases. The main focus is on the separation of logic, content and layout, as well as on the re-use of developed applications. The administration of content is browserbased.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

Seitdem Tim Berners-Lee für das World Wide Web im Jahr 1989 am CERN[1] den Grundstein gelegt hat, wurden unzählige Web-Projekte aufgesetzt und ins Netz gestellt. Die meisten dieser Projekte waren bis vor wenigen Jahren Sammlungen von HTML-Seiten, die von engagierten Menschen ins Leben gerufen wurden. Mittlerweile jedoch ist das World Wide Web auch zu einem kommerziell interessanten Markt geworden. Es gibt eine Vielzahl von Geschäftsmodellen, von virtuellen Shops über Informationsbörsen für geschlossene Nutzerkreise bis zu werbefinanzierten Angeboten der vielfältigsten Art. Mit all diesen Angeboten hat die Zahl der Nutzer des Internets — womit heute meist das World Wide Web gemeint ist — stetig zugenommen[2].

Im Zuge der Ausweitung des World Wide Web haben sich die von den Nutzern geforderten Formen der Websites verändert und weiterentwickelt, entsprechend mußten und müssen sich auch die Produktionsmethoden für Websites weiterentwickeln. Dies gilt sowohl für die initiale Produktion der Site als auch für die laufende redaktionelle Produktion bzw. Bearbeitung.

Heute ist es kaum noch möglich, eine erfolgreiche Website zu betreiben, ohne auf zunehmend komplexere Software und wachsende Kenntnisse zurückzugreifen. In vielen Agenturen wurden und werden die einzelnen Projekte jeweils als weitgehend unabhängige Projekte gefahren, das heißt, es wurden immer wieder Lösungen für die gleichen Aufgabenstellungen neu entwickelt. Dieses Vorgehen führte zu sehr hohen Kosten für das einzelne Projekt. Der Bedarf an generalisierten Methoden und der Einführung eines dazu passenden Vorgehensmodells war und ist vorhanden. Mittlerweile existieren eine ganze Reihe von Content Management und Redaktionssystemen[3], die hierfür angeboten werden. Das in sie investierte Know-how muß allerdings gut bezahlt werden. Für nicht kommerzielle Projekte oder solche mit einem niedrigen Budget muß nach Alternativen gesucht werden. In dieser Arbeit wird eine solche Alternative für ein individuell erstelltes Verwaltungssystem auf Basis von Open Source Komponenten entwickelt.

Ursprung und Umgebung des Themas Entstanden ist die Idee zu der Arbeit bei der Tätigkeit des Autors im Verlagswesen. Gefordert waren Sites für viele Nutzer und mit kurzen Entwicklungszeiten. Die Sites mußten häufig an neue Forderungen von Seiten der Nutzer, der Redakteure oder auch des behandelten Themas selbst angepaßt werden. Für die Auftraggeber waren niedrige Kosten, sowohl Investitions- als auch Betriebskosten wichtig, denn die Projekte sollten sich zumindest selbst finanzieren.

[1] Abkürzung für französisch **C**onseil **E**uropéen pour la **R**echerche **N**ucléaire, eine Kernforschungseinrichtung in Genf.

[2] Mitte 1998 zwischen 40 und 80 Millionen Benutzer weltweit — mit stark steigender Tendenz [Brockhaus 2000].

[3] In dieser Arbeit wird die neue deutsche Rechtschreibung eingesetzt.

Wie kam es zu der Aufgabe? Aus der Erkenntnis, daß sehr viel Aufwand in die ständige erneute Bearbeitung bereits gelöster Problemstellungen gesteckt wurde, entstand der Wunsch, eine Art Baukastensystem zu besitzen und als Lösungspool parat zu haben. Dies führte zu einer ersten Version eines noch prozeduralen Baukastens für Verwaltungssysteme von Websites, der allerdings bei der Migration von einem Projekt zum nächsten deutliche Schwächen aufzeigte. So entstand der Bedarf nach einem flexibleren Nachfolger dieses Baukastens.

Welches Ziel wird angestrebt? Mit den gewonnenen Erfahrungen aus den verschiedenen vorhergehenden Projekten soll ein nun objektorientiertes Framework als Baukastensystem zur Erstellung von Verwaltungssystemen für Websites entstehen. Diese Arbeit wird durch den Wunsch getrieben, eine Verringerung der Entwicklungszeiten durch Wiederverwendung bei gleichzeitiger Steigerung der Qualität der Sites und erhöhter Flexibilität zu erreichen.

Das Framework soll für kleine und mittlere Projekte geeignet sein, die eine schnelle Umsetzung, möglichst einfache Pflege und niedrige Investitions- und Betriebskosten benötigen. Der Schwerpunkt liegt hier auf Sites, die primär Informationen publizieren wollen.

In späteren Ausbauphasen soll ebenfalls die Entwicklung von interaktiven Sites möglich sein, wie z.B. Communities und E-Commerce-Sites (siehe auch Abschnitt 1.1).

Was erwartet den Leser? In dieser Arbeit wird das Framework entwickelt und komplett beschrieben. Der Leser bekommt eine Einführung in die Begrifflichkeiten und Konzepte, die hinter dem Framework stehen, sowie einen Überblick über die Organisation des Frameworks und an Hand eines exemplarisch durchgeführten Projektes eine Handlungsanweisung für den Einsatz des Frameworks. Auf der beigelegten CD[4] befinden sich: das Framework und seine API-Dokumentation, die exemplarische Website und die notwendigen Programme zum Betrieb des Frameworks.

Wie ist die Beschreibung aufgebaut? Jedem größeren Abschnitt ist, zur Orientierung für den Leser, eine kurze Zusammenfassung vorangestellt.

- Kapitel 1 *Gegenstandsbestimmung*
  Da das Thema Web Content Management sich noch in seiner Anfangs- und Findungsphase befindet, werden hier in aller Kürze die für das Verständnis der Arbeit notwendigen Begriffe erklärt.

- Kapitel 2 *Das Framework*
  Hier werden die Anforderungen an das Framework, die zu Grunde lie-

[4] Aktualisierte Fassungen des Frameworks und elektronische Versionen dieser Arbeit finden sich unter `http://www.stolt.de/wck` [wwwCite:WCK].

gende Philosophie und die Grundentscheidungen im Design des Frameworks dargelegt. Auch die vorhandenen Basisklassen und deren Zusammenspiel wird in großen Zusammenhängen erläutert[5].

- Kapitel 3 *Exemplarischer Einsatz des Frameworks*
  In diesem Kapitel wird an Hand der Entwicklung einer Alternative zur bisherigen Website des IKS-Projektes der *Hochschule für Angewandte Wissenschaften Hamburg* das entwickelte Framework an einer praxisnahen Aufgabe getestet. Hier sind ebenfalls viele Hinweise auf die Einsatzmöglichkeiten des Frameworks enthalten.

- Kapitel 4 *Zusammenfassung und Ausblick*
  Den Abschluß bildet eine Zusammenfassung der Erfahrungen mit dem Framework und ein Ausblick auf Erweiterungen und besondere Themenstellungen, die im Rahmen dieser Arbeit nicht erschöpfend bearbeitet werden konnten.

[5] Die konkrete Dokumentation der API der einzelnen Klassen und der Implementierung ist auf der beigelegten CD als HTML-Set vorhanden, ebenso der Quelltext des Frameworks sowie derjenige der exemplarischen Anwendung.

Kapitel 1

Gegenstandsbestimmung

„Es ist alles so schön bunt hier, ich kann mich gar nicht entscheiden...“
Nina Hagen

Die Technik des World Wide Webs bringt einen ganzen Strauß verschiedener Arten von Websites zu Tage. All die verschiedenen Arten erfordern für ihren Aufbau und die Pflege ihrer Inhalte unterschiedliche Vorgehensweisen und Methoden. Wenn in dieser Arbeit ein Framework zur Erstellung von Verwaltungssystemen für Websites erarbeitet werden soll, dann müssen zunächst einige Begrifflichkeiten, Unterschiede und Gemeinsamkeiten verschiedenster Websites untersucht und erklärt werden, um dann eine Strategie und ein Einsatzfeld des Frameworks festlegen zu können[1].

Im Abschnitt 1.1 werden zunächst die Archetypen von Websites vorgestellt, um dann im Abschnitt 1.2 genauer auf die im Web möglichen Inhaltstypen und im Abschnitt 1.3 auf die typischen Bestandteile von Websites einzugehen.

Im Abschnitt 1.4 werden die an der Erstellung und dem Betrieb einer Website beteiligten Rollen vorgestellt.

Im Abschnitt 1.5 werden die drei häufigsten Qualitätsanforderungen an Websites aufgeführt und deren Wechselwirkungen untereinander und mit den im Abschnitt 1.6 vorgestellten Werkzeugen erläutert. Bei der Untersuchung der Werkzeuge wird vor allem auf die Möglichkeit der Bewältigung der nötigen Arbeiten eingegangen.

Als Abschluß der Gegenstandsbestimmung werden im Abschnitt 1.7 Erfahrungen aus vorhergehenden Projekten, die in die Frameworkentwicklung eingeflossen sind, vorgestellt und im Abschnitt 1.8 die aus allen vorhergehenden Abschnitten resultierenden Anforderungen an das Framework als grobe Richtungsgebung zusammengefaßt.

[1] Vertiefte Darstellungen finden sich bei [Bauer 2001], [Büchner et. al. 2001] und [Kim 2001], wobei dort zum Teil unterschiedliche Diktionen benutzt werden.

1.1 Archetypen von Websites

Es gibt einige Archetypen von Websites, diese unterscheiden sich vor allem in ihrer inhaltlichen Ausprägung und dem Grad der interaktiven Nutzung. Die wichtigsten Archetypen und ihre Eigenheiten werden für die Diskussion in dieser Arbeit kurz vorgestellt. Zu jedem Archetyp werden auch Beispiele genannt, wobei jedoch zu berücksichtigen ist, daß dies reale Websites sind und reale Websites eigentlich immer Mischungen unterschiedlicher Archetypen sind.

Imagesite Bei diesem Typus ist die werbende Präsentation für eine Firma, ein Produkt oder eine Veranstaltung treibende Kraft. Es handelt sich um mit aufwendigen grafischen Mitteln gestaltete Seiten mit meist geringem Umfang. Häufig werden großflächige Grafiken eingesetzt oder Plugins wie Flash/Shockwave[2] und andere mediale Elemente wie Ton und Video. Oft sind diese Sites Begleiter von Werbekampagnen in den „klassischen“ Medien wie Print, Funk und Fernsehen. Es finden auf solchen Sites kaum Aktualisierungen statt, eher wird ein vollständiger Relaunch durchgeführt.

Eine Vertreterin, die diesem Archetypus nahe kommt, ist die unten abgebildete Site `http://www.lucy-planning.de` [wwwExample:LucyPlanning] oder auch die Site der Agentur *Springer und Jacobi* [wwwExample:SuJ].

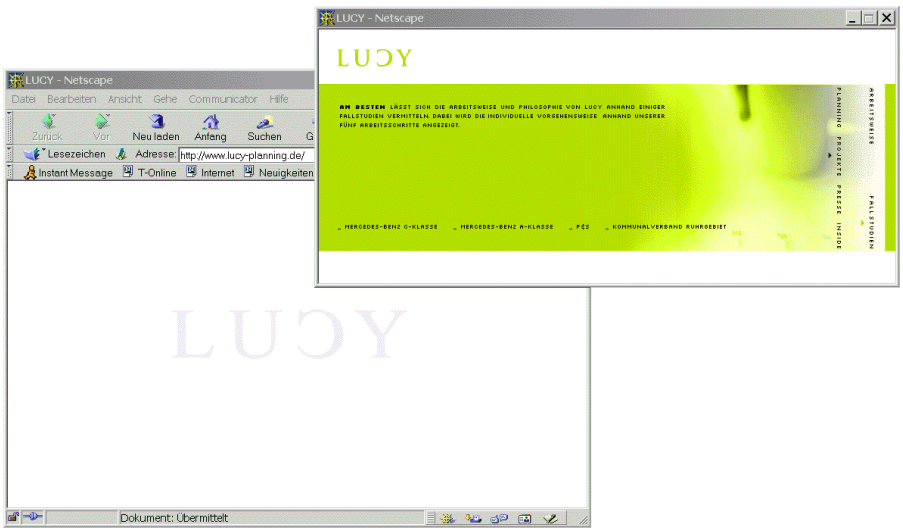

Abbildung 1.1: Beispiel für eine Imagesite

[2] Beides Produkte von Macromedia mit einer sehr hohen Marktdurchdringung, zu finden unter `http://www.macromedia.com` [wwwProduct:Macromedia].

Abbildung 1.2: Beispiel für einen Informationsanbieter

Informationsanbieter Der Informationsanbieter stellt aktuelle Informationen bereit. Dies können Nachrichten, Terminkalender, FAQ's[3], Pollenflugangaben und ähnliches sein. Gemeinsamkeit all dieser Möglichkeiten ist, daß die einzelnen *Informationsgruppen* (z.B. alle Nachrichten) aus *Informationseinheiten* (eine einzelne Nachricht) bestehen, die jeweils die gleiche Struktur besitzen. Im Fall von Nachrichten kann diese Struktur wie folgt aussehen: Schlagzeile, Aufhänger, Nachrichtentext, Datum der Veröffentlichung und Quelle. Alle Nachrichten bestehen aus diesen *Informationselementen.* Es gibt Übersichtsseiten, auf denen nur die aktuellen Schlagzeilen mit den Aufhängern jeweils verknüpft mit einer Folgeseite mit dem Nachrichtentext zu finden sind. Um den Benutzungskomfort zu erhöhen, kann auf den Seiten mit den vollständigen Nachrichten z.B. eine Liste der aktuellen Schlagzeilen geführt werden, so daß der Leser direkt via Schlagzeile von Nachrichtentext zu Nachrichtentext gelangen kann.

Die Aktualität ist ebenfalls eine gemeinsame Eigenschaft. Als Beispiel seien Termine genannt, die schon längst verstrichen sind oder der berühmte Zeitungsmacherausspruch: *„Nichts ist so alt wie die Nachricht von gestern!“*. Zwar ist es oft sinnvoll, die Informationen in einem Archiv weiter zur Verfügung zu stellen, aber den Leser interessieren meist nur die wirklich aktuellen Informationen.

Ebenfalls ist meist eine Kategorisierung der Informationen vorhanden, so kann der Nutzer die für ihn relevanten Informationen schneller finden. Durch

[3]Frequently Asked Questions, zu deutsch: häufig gestellte Fragen, zu denen die Antworten geliefert werden.

Kategorisierung ist es überhaupt erst möglich, eine umfangreiche Sammlung von Informationen zur Verfügung zu stellen.

Bei Informationsanbietern sind je nach Art des Angebotes feste Publikationszeiten möglich. Es kann zum Beispiel Tages- oder Wochenausgaben geben, die jeweils als komplettes Paket im Web veröffentlicht werden (siehe hierzu Abschnitt 1.7.2). Häufiger dürfte aber eine Art *„Just-In-Time“*-Publizierung vorliegen, in der Informationen wie z.B. Nachrichten sofort nach ihrem Erscheinen in das Angebot eingebaut werden.

Unter `http://www.sport1.de` [wwwExample:Sport1] präsentiert sich ein Sportinformationsanbieter, der hier exemplarisch für Informationsanbieter abgebildet wird. Auch Tageszeitungen sind gute Kandidaten für diesen Archetypus, hier sei stellvertretend die Financial Times Deutschland genannt `http://www.financialtimes.de` [wwwExample:FinancialTimes].

Shoppingsysteme Bei diesem Typus durchläuft der Nutzer mehrere Phasen in der Site (die Phasen werden sehr zum Bedauern der Betreiber häufig nicht vollständig durchlaufen). Zunächst durchsucht der Nutzer einen Katalog von Waren oder Dienstleistungen, dabei sammelt er die für ihn interessanten Positionen in einem virtuellen Warenkorb und begibt sich mit diesem an die „Kasse“. Dort identifiziert er sich und regelt die Formalitäten des Bezahlens. Anschliessend wird vom Betreiber der Site die gewünschte Ware oder Dienstleistung geliefert.

Abbildung 1.3: Beispiel für ein Shoppingsystem

In dieser Art von Site spielt das Sessionmanagement, also das Verfolgen der Aktivitäten eines Nutzers eine große Rolle. Nur so kann aus dem anonymen Nutzer ein zahlender Kunde werden.

Ebenso ist hier die Anbindung der Website an zur Firma gehörende Geschäftssysteme von großer Bedeutung. Es muß sichergestellt sein, daß sowohl die Auslieferung als auch das Inkasso für die gelieferten Waren bzw. Dienstleistungen erfüllt wird.

Der Versandhandel Otto sei hier als Beispiel für ein Shopsystem genannt, er ist unter `http://www.otto.de` [wwwExample:Otto] zu finden.

Kommunikationsplattformen Die Bildung von Kommunikationsplattformen ist der bislang letzte Archetyp, der sich im World Wide Web entwickelt hat[4]. An dieser Entwicklung ist deutlich der Wandel des World Wide Webs vom Broadcastmedium zum individuellen Kommunikationstool zu erkennen. Kommunikationsplattformen sind Treffpunkte für Menschen mit gleichen Interessen. Man trifft sich in einem virtuellen Raum und tauscht Gedanken, Meinungen, Informationen und dergleichen zu einem Thema aus. Hierzu stehen verschiedenste Werkzeuge bereit: Foren, Chats, Abstimmungstools, Profile bzw. Visitenkarten, persönliche Mailboxen und anderes.

Wesentlich an diesen Kommunikationsplattformen ist, daß die Mitglieder dieser Gemeinschaften sich selbst identifizieren können und ihre Persönlichkeit innerhalb dieser Kommunikationsplattform durch diese Identifikation eindeutig ist. Der Schwerpunkt des Nutzers liegt nicht mehr nur beim reinen Konsumieren, sondern bei der aktiven Beteiligung an der Fortentwicklung der Gemeinschaft.

Kommunikationsplattformen sind vom Wesen her hoch dynamisch. Damit sind keine festen Publikationszeiten — wie bei einigen Informationsanbietern — mehr möglich, denn die Nutzer stellen Informationen und Kommunikationen asynchron bereit.

Die von den Nutzern generierten Inhalte werden normalerweise vom Betreiber der Plattform kontrolliert, dies geschieht deshalb, weil der Betreiber meist ein bestimmtes Bild und ein Niveau seiner Kommunikationsplattform aufrechterhalten möchte[5]. Der Betreiber muß aus rechtlichen Gründen auch in der Lage sein, Inhalte zu entfernen, die illegalen oder zweifelhaften Gehalts sind, da er sonst für die Verbreitung dieser Inhalte verantwortlich gemacht werden kann [Zimmerling, Werner 2001].

[4] Obgleich Kommunikationsdienste die Basisdienste und Motivation des Internets sind. E-Mail, Newsserver und FTP sind mit die ältesten Dienste und stellen auch heute noch einen Großteil des Datenvolumens. Die Übernahme dieser Dienste von Websites ist eine Forderung und Folge der Vereinfachung des Umganges mit dem Netz durch die zunehmende Verbreitung von Netzzugängen in nicht technikaffinen Bevölkerungsschichten. Grundvoraussetzung für derartige Sites war die Verfügungbarkeit der Basistechnologie dynamischer Websites.

[5] Eine Anfrage eines Päderasten innerhalb einer Familienplattform ist beispielsweise sehr unerwünscht und rufschädigend.

Eine recht umfangreiche Kommunikationsplattform findet sich im deutschsprachigen Raum unter `http://www.redseven.de` [wwwExample:RedSeven] wieder (siehe auch Abbildung 1.4).

Abbildung 1.4: Beispiel für eine Kommunikationsplattform

Weitere Kommunikationsplattformen unterschiedlicher Art stellen die Site `http://www.slashdot.com` [wwwExample:Slashdot] oder auch die Clubs, Interestlists, Profile, Messenger und privaten Homepages der ehemals reinen Suchmaschine `http://www.yahoo.de` [wwwExample:Yahoo] dar.

1.2 Informationsstrukturen

Im vorherigen Abschnitt wurden verschiedene Archetypen vorgestellt. Von diesen Archetypen möchte ich nun abstrahieren und mich nur mit verschiedenen Typisierungen von Informationen beschäftigen.

Websites können als eine Menge zusammengehöriger Webseiten mit Webobjekten betrachtet werden. Eine einzelne Webseite wiederum besteht auf einer Ebene aus HTML-Text und eventuell eingebetteten Webobjekten (Bilder, Filme, Java-Applets, Flashprogramme, etc.), auf einer anderen Ebene kann man eine Webseite aber auch als eine Menge zusammengehöriger Informationseinheiten betrachten. Diese Informationseinheiten bilden den eigentlichen Inhalt einer einzelnen Webseite und in ihrer Summe auf den Seiten den Inhalt der gesamten Site.

1.2.1 Klassifikation von Inhalten

Zur Klassifikation von Inhalten können sehr unterschiedliche Aspekte herangezogen werden. Da wäre zunächst die Lebensdauer bzw. die Häufigkeit der Änderung, dann die Struktur der Information und die Funktion innerhalb der Site.

Lebensdauer

Bei den zu pflegenden Informationen können unter zeitlichen Gesichtspunkten drei Arten unterschieden werden.

- **Statische Informationen** Als — im wesentlichen — unveränderlich zu betrachtende Informationen, wie zum Beispiel der Lebenslauf einer schon verstorbenen Person.
- **Semidynamische Informationen** Dies sind Informationen, die eine lange Haltbarkeit aufweisen. Telefonverzeichnisse einer kleinen Firma mit langjährigen Mitarbeitern stellen gute Kandidaten für semidynamische Informationen dar.
- **Dynamische Informationen** Dies sind Informationen mit einer höheren Änderungsfrequenz, z.B. der aktuelle Tabellenstand der Fußballbundesliga oder der aktuellen Kassa-Kurs einer Aktie.

Struktur

Eine andere Einordnung der Informationen ist die nach dem Grad der Strukturierung der Daten.

- **Reguläre Strukturen** Eine Telefonliste (Informationsgruppe) besteht aus Datensätzen (Informationseinheiten) mit wohlstrukturierten Inhalten (Informationselementen) (z.B. Name, Telefonnummer, Standort, Funktion). Jeder Datensatz (Informationseinheit) beinhaltet an gleicher Stelle den gleichen Informationstyp.
- **Unstrukturiert** Webseiten, die per Hand gepflegt werden, sind gute Kandidaten für unstrukturierten Inhalt. Ein Autor schreibt seine Gedanken und Einfälle nieder und nutzt dabei die Möglichkeit des Hypertextes, um unterschiedliche Aspekte seines Themas frei mittels Verknüpfungen zu assozieren. Das später noch angesprochene WikiWeb [Leuf, Cunningham 2001] ist eine Umgebung, die für die Aufnahme unstrukturierter Informationen gut geeignet ist[6].

Die beiden obigen Positionen stellen die Extremwerte dar. Im Grunde genommen gibt es aber kaum echt unstrukturierte Informationen. In den meisten als unstrukturiert bezeichneten Informationen steckt eine irreguläre Struktur oder sie lassen sich mit regulären Metadaten beschreiben. Ein Beispiel für irreguläre Strukturen ist die Gruppe der Publikationen. Jede Publikation wird untergliedert in Absätze, Abschnitte, Kapitel, in Tabellen, Grafiken mit Untertiteln und so weiter. Allerdings ist die Reihenfolge und das Vorhandensein der möglichen Elemente von Publikation zu Publikation meist sehr unterschiedlich. Die Publikationen können aber auf einer Metaebene sehr wohl mit regulären Strukturen wie XML-Schemata beschrieben

[6] Ein WikiWeb ist eine Art „Freitext“-Datenbank.

oder mit ihren Metadaten sogar zu regulären Datensätzen werden (es erfolgt dann eine Erfassung nach Autor, Verlag, Erscheinungsort und -zeitpunkt, etc.). Auch die Struktur einer Website gehört als Graph zu den irregulären Strukturen.

Oft lassen sich in einer Site abgrenzbare individuelle Exemplare von Dingen, Personen, Begriffen mit regulären Strukturen finden. Diese Exemplare werden hier als Informationseinheiten bezeichnet. Eine Informationseinheit besteht aus mehreren Teilen (Informationselemente oder Attribute genannt). Die Informationseinheiten stehen oft miteinander in Beziehung.

Ein Beispiel für eine Informationseinheit ist eine Nachricht, die z.B. aus den Informationselementen Titel, Einleitung, Textkorpus und Erstellungsdatum besteht. Eine Gruppe von Nachrichten bildet oft ein Thema (dies wäre eine Beziehung unter den Nachrichten) oder die Nachrichten gehören einem bestimmten Bereich an (z.B. Fußball, Handball, Tennis).

Für die Entwicklung datenbankgestützter Sites ist die Identifikation der Struktur der Informationseinheiten der Site und deren Beziehungen ein wichtiger Schritt. Bei relationalen Datenbankmodellen werden aus der Struktur der Daten die notwendigen Tabellendefinitionen und deren Verknüpfungen abgeleitet. Auch für die Verarbeitung der Daten zur Darstellung und ebenso in Formularen sind diese Datenbeschreibungen die notwendige Grundlage.

Funktion

Innerhalb einer Site gibt es für Informationen verschiedene Aufgaben.

- **Inhalt** Die offensichtlichste Funktion für Information ist die des Inhaltes, also z.B. Nachrichten, Adressen, Produktabbildungen und Preise, Geschichten, etc.
- **Präsentation** Hiermit sind Elemente einer Site gemeint, die typischerweise die Inhalte unterstützen. Dem Nutzer der Site wird durch die Präsentation die Verarbeitung des Inhaltes erleichtert bzw. schmackhaft gemacht. Unter Präsentation fällt auch das Corporate Design als Teil des Corporate Images.
- **Navigationssystem** Das Auffinden von Inhalten zu ermöglichen, ist Aufgabe des Navigationssystems. Ein Navigationssystem auf einer Site besteht aus mehreren Teilen [Rosenfeld, Morville 1998], die jeweils eine Grundfrage für den Nutzer klären müssen.
    - *Navigation*, wohin kann ein Nutzer von der aktuellen Seite gelangen?
    - *Orientierung*, wo befindet sich der Nutzer gerade und wie ist er dorthin gelangt?

- *Bezeichnungen (Labeling)*, was befindet sich auf den dahinter liegenden Seiten?[7]

Wenngleich auch die Inhalte auf den meisten Sites die wesentliche Funktion für die Nutzer darstellen, so sind sie jedoch ohne eine gute Navigation unbrauchbar, da nicht auffindbar. Ein anderer wesentlicher Faktor für die Akzeptanz einer Site ist die Präsentation, denn die Präsentation der Inhalte muß der Intention der Nutzer entsprechend gestaltet sein und die Inhalte in ihrer Wirkung unterstützen. Ebenso kann die Präsentation den Zusammenhalt der Site stützen und die Außenwirkung einer Site — bzw. das Corporate Image des Betreibers — beeinflussen.

Die Gewichtung der drei Aspekte Inhalt, Präsentation und Navigation ist auf realen Sites sehr unterschiedlich. Das Spektrum reicht von sachlich nüchternen Darstellungen, in denen Informationen verdichtet dargestellt werden, wie die der Financial Times Deutschland (zu finden unter `http://www.financialtimes.de` [wwwExample:FinancialTimes]), bis hin zu nur auf Präsentation basierenden Sites, die als Schauplatz für Webtechniken gedacht sind `http://www.superbad.com` [wwwExample:SuperBad].

1.2.2 Wartung der Informationen und Sitestrukturen

Die Aufgabe der Wartung und Administration einer Website gliedert sich in drei Bereiche.

- **Inhaltspflege** Am häufigsten dürfte die Notwendigkeit zur Pflege des Inhaltes vorhanden sein. Es kommen neue Nachrichten hinzu, Linklisten verändern sich oder neue Personen werden einer Telefonliste hinzugefügt. All diese Änderungen sollten natürlich von den redaktionellen Betreibern einer Site durchgeführt werden, und das auf eine möglichst einfache Weise.
  In den Bereich der Inhaltspflege gehört bei Kommunikationsplattformen ebenfalls die Überprüfung der nutzergenerierten Inhalte auf Verstöße gegen die Etikette der Community bzw. auf Verletzungen geltenden Rechts.
- **Strukturpflege** Hierunter ist eine Veränderung der Bereiche einer Website zu verstehen. Die Veränderungen in der Struktur einer Site können zweierlei Natur sein.
  - **Lokal** Es kann zum Beispiel eine neue Kategorie in einer Linkliste eingeführt werden. Diese lokalen Änderungen sollten von spezi-

[7] Die Informationsarchitekten haben das Labelingsystem so zu gestalten, daß auf diese Frage von jedem — oder zumindest vom Großteil — der Nutzer die gleiche Antwort gegeben wird und diese Antwort zu den wirklichen Inhalten auf den dahinter liegenden Seiten passt.

ell dazu berechtigten Redakteuren[8] durchgeführt werden können. Zeitlich sind solche Änderungen meist semidynamisch.

- **Global** Das Erschaffen völlig neuer Bereiche einer Website oder das Einbringen komplett neuer Konzepte sind globale Änderungen. Diese sind nicht mehr von Redakteuren durchführbar, sondern bedürfen meist der Umgestaltung von Navigation und Orientierung auf der gesamten Site.

- **grafische Gestaltung/Layout** Diese Pflege taucht im Lebenszyklus einer Website typischerweise am seltensten auf und wird dann meist als ein Relaunch bezeichnet. Vorteilhaft ist es, wenn die der Site zu Grunde liegende Technik einen Relaunch ermöglicht[9].

1.3 Typische Bestandteile von Websites

In diesem Abschnitt wird eine beliebige Site in ihre typischen Bestandteile zerlegt. Hierbei werden nur die Bestandteile betrachtet, die in der Nutzersicht der Site auftreten, denn die Administrationsseiten sind, sofern vorhanden[10], der Öffentlichkeit ja nicht zugänglich.

Die Darstellung findet anhand der beiden Sites `http://www.mysql.com` [wwwProduct:MySQL] und `http://www.heise.de` [wwwExample:Heise] statt. Es werden einzelne Screenshots präsentiert, in denen die typischen Elemente markiert wurden.

Im Abschnitt 1.3.1 wird die gesamte Seite in ihre Teile zerlegt und betrachtet, und in den folgenden Abschnitten 1.3.2 bis 1.3.4 werden die Bestandteile des in 1.3.1 identifizierten Contentbereiches untersucht.

1.3.1 Kopf, Fuß, Navigation und Content

Fast jede Seite einer jeden Site läßt sich in vier Teile zerlegen. Dies sind der Kopf und der Fuß einer Seite, sowie die Bereiche der Navigation und des Contents (siehe auch Abbildung 1.5).

[8] Zu den Rollen der Mitwirkenden an einer Site siehe auch Abschnitt 1.4.

[9] Es sind gescheiterte Relaunches bekannt, weil es nicht möglich war, die vorhandenen Daten (ca. 40.000 Nachrichtenseiten) im neuen Erscheinungsbild ohne eine sehr hohe Investition in Arbeitszeit zu publizieren.

[10] Nur bei Verwaltungssystemen gibt es Administrationsseiten, bei von Hand als HTML-Seiten erstellten Sites natürlich nicht (siehe auch Abschnitt 1.6).

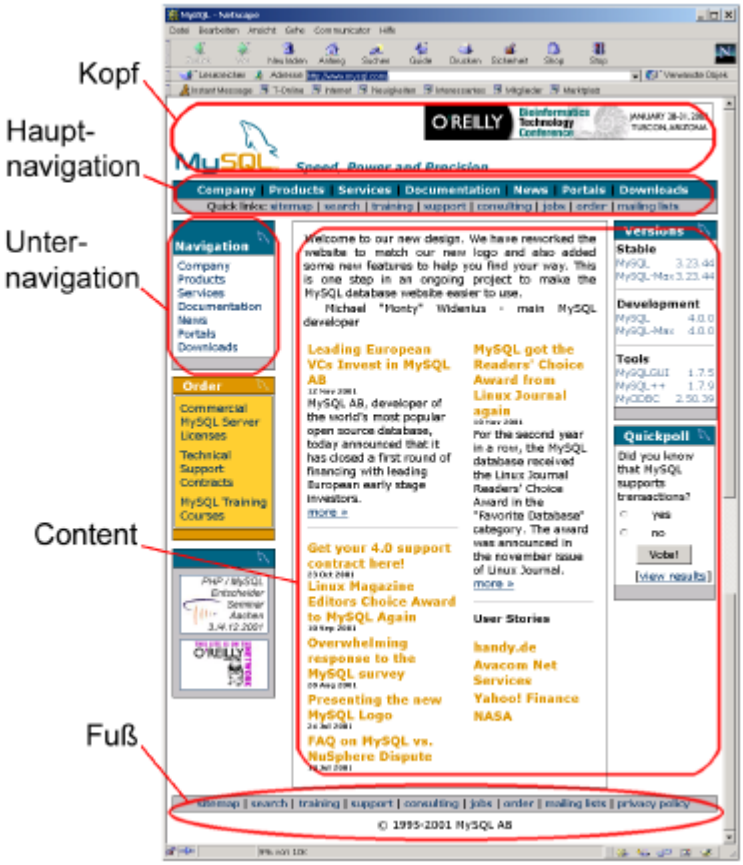

Abbildung 1.5: Bestandteile; Kopf-, Fuß-, Navigations- und Contentbereich

Im Kopf einer Seite findet sich meist ein Logo der Site oder des Betreibers, häufig auch Werbebanner[11]. Dieser Kopf ist meist wesentlicher Bestandteil des Corporate Images einer Site. In der Regel taucht er unverändert oder in farblichen Varianten auf jeder der Seiten einer Site auf. Hier ist auch ein guter Platz für das Motto der Site (im Beispiel: *„Speed, Power and Precision“*). Im HTML-Text des Kopfes sind meist Daten für Suchmaschinen, Indexierungen oder spezielle Zählmechanismen[12] eingebettet.

Der zweite große Bereich einer Seite umfaßt die Navigation, oft getrennt in eine Haupt- und eine oder mehrere Unternavigationen. Die Hauptnavigation erlaubt das schnelle Erfassen und Betreten der Hauptbereiche einer Site, sie ist auf allen Seiten vorhanden und sollte im wesentlichen, bis auf Hervorhebungen des aktuellen Bereiches, unverändert bleiben. Die Unternavigation wechselt von Bereich zu Bereich[13]. Zur Navigation wird meist auch

[11] Je nach zu Grunde liegendem Geschäftsmodell der Site.

[12] Wie z.B. sogenannte IVW-Tags zur Reichweitenermittlung von Werbeträgern; `http://www.ivw.de` [wwwCite:IVW].

[13] Vergleiche hierzu die Abbildungen 1.5, 1.6 und 1.8.

die Ortsbestimmung für die Nutzer gezählt[14], also der Hinweis darauf, in welchem Bereich sich der Nutzer aufhält.

Der dritte Teil einer Seite ist der Contentbereich. Hier finden sich die eigentlichen Inhalte einer Site wieder, also Nachrichten, Termine, Bilder, Geschichten, Personalien, Umfragen, Übersichten und so weiter. Die Inhalte für den Contentbereich werden von Redakteuren geliefert. Ein Verwaltungssystem für eine Website muß den Redakteuren einen einfachen Weg zur Änderung dieser Inhalte bereitstellen. Oft sind die einzelnen Inhalte der verschiedenen Seiten direkt voneinander abhängig und müssen konsistent[15] verändert werden. Auch dies ist eine Aufgabe für ein Verwaltungssystem.

Der Fuß einer Site bildet in der Regel einen abschliessenden Rahmen und beinhaltet oft eine weitere Navigation zu den Hauptbereichen der Site und Verweise auf ein Impressum, eine Sitemap, Suchfunktionen und Copyrightvermerke. Ebenso wie der Kopf ist der Fuß meist auf allen Seiten einer Site gleich.

1.3.2 Einzeldarstellungen

Die Inhalte einer Site lassen sich oft als einzelne Informationseinheiten auffassen (wie im Abschnitt 1.2.1 unter Struktur dargestellt).

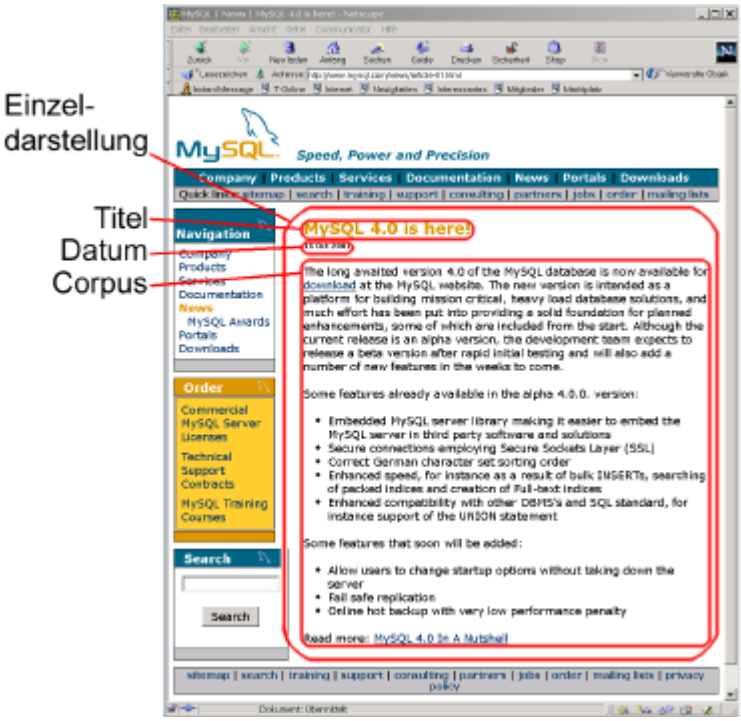

Abbildung 1.6: Bestandteile; Einzeldarstellung

[14] Siehe hierzu [Rosenfeld, Morville 1998]
[15] Siehe auch Abschnitt 1.5.

Umfangreiche Informationseinheiten, wie die Nachricht in der Abbildung 1.6, werden meist auf eigenen einzelnen Seiten präsentiert. Die Informationseinheiten besteht aus mehreren Informationselementen (im Beispiel aus Titel, Datum, Corpus). Die Elemente werden jeweils mit eigenen Formatierungen dargestellt. Wenn viele Informationseinheiten gleicher Struktur in einer Site auftauchen, dann sollten die Formatierungen der Elemente über alle jeweiligen Einzeldarstellungen hinweg natürlich gleich sein.

Einzeldarstellungen können auch wie in der Abbildung 1.7 in Kombination mit anderen Teilen auftreten. Hier werden dann meist verkürzte Darstellungen gewählt — im Falle einer Nachricht zum Beispiel nur der Titel und eine Zusammenfassung des Korpus, ein sogenannter Teaser — mit einem Verweis auf die vollständige Darstellung.

Um zu den Einzeldarstellungen zu gelangen, werden gern Auswahllisten und Tabellen genutzt.

1.3.3 Listen und Tabellen

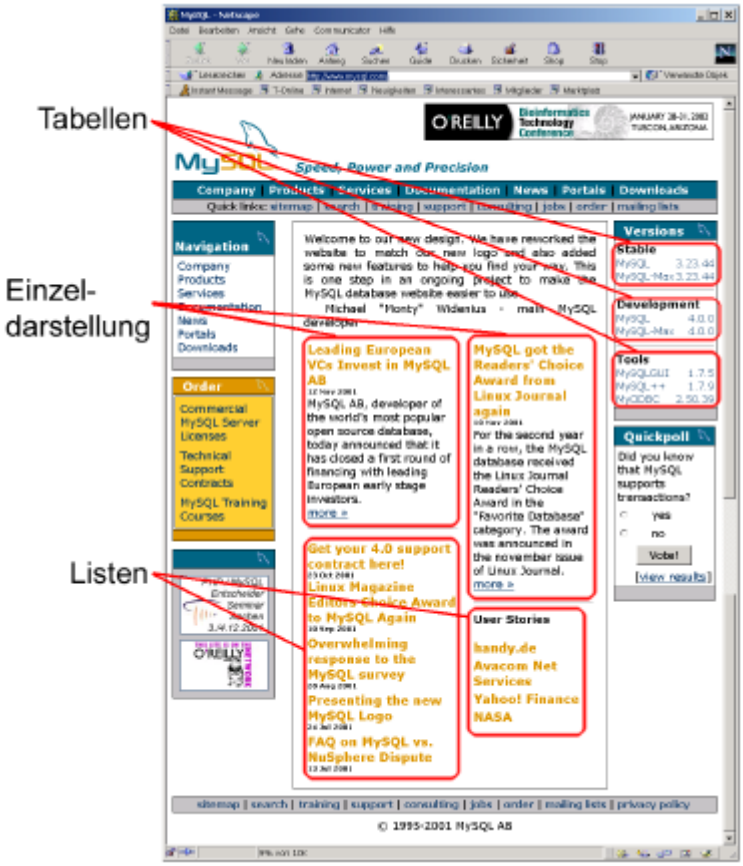

Abbildung 1.7: Bestandteile; Listen, Tabellen und Einzeldarstellungen

Unter Listen sind hier gleichartige, untereinanderstehende Darstellungen von Teilen von strukturgleichen Informationseinheiten zu verstehen (z.B. Nachrichten oder Termine). Es müssen keine Aufzählungs-, Definitions- oder nummerierten Listen im Sinne von HTML sein. Im Beispiel der Abbildung 1.7 sind zwei Listen zu finden.

Beide Listen sind Auswahllisten. In der einen Liste werden die aktuellen Nachrichten der Site mit ihrem Titel und dem Veröffentlichungsdatum aufgeführt. Der Titel ist jeweils ein Verweis auf eine Einzeldarstellung, wie sie Abbildung 1.6 zeigt. In der anderen Liste werden aktuelle Einsatzberichte der Software präsentiert, auch hier verweisen die Titel auf entsprechende Einzeldarstellungen.

Weniger häufig als Listen tauchen in Sites tabellarische Darstellungen von Informationseinheiten auf. Tabellen sind die dritte Variante der Darstellung von Informationseinheiten und haben eine Reihe eigener Formatierungsmöglichkeiten, wie z.B. das spalten- oder reihenweise wechselnde Einfärben des Hintergrundes und ähnliches.

Eine interessante Möglichkeit von Tabellen ist die Sortierung ihres Inhalts. Eine der Spalten dient hierbei als Sortierkriterium. Hierzu wird ein geeigneter Tabellenkopf benötigt, in dem die entsprechenden Links zur Sortierung der Spalten zu finden sind[16].

Auf der betrachteten Site kommen Tabellen noch in ganz anderer Form zum Einsatz. Alle Darstellungen als Kästen (In der Abbildung mit den Überschriften: ‚Navigation‘, ‚Order‘, ‚Version‘ und ‚Quickpoll‘) sind im HTML-Code Tabellen. Da die Gestaltung dieser Bestandteile einen wesentlichen Punkt des Corporate Images ausmacht, ist die gleichförmige Darstellung sehr wichtig. Das in dieser Arbeit entwickelte Verwaltungssystem ist in der Lage, die Konsistenz in der Darstellung solcher Elemente zu wahren.

1.3.4 Formulare

Bei Sites, in denen der Nutzer selbst Informationen hinterläßt oder anderweitig aktiv wird, bilden Formulare einen weiteren wichtigen Bestandteil. Bei browserbasierten Verwaltungssystemen sind Formulare existentiell.

Zu einem Formular gehört auch immer eine entsprechende Formularbehandlung. Was soll mit den Daten geschehen, die der Nutzer im Formular einträgt? Wie werden die Daten im Formular auf Gültigkeit geprüft? Wie wird auf fehlerhafte oder fehlende Daten reagiert? All diese Fragen werden in der Formularbehandlung geklärt.

Das Formular der Abbildung 1.8 stellt ein einfaches Weiterleitungsformular dar. Der Nutzer kann hier die oberhalb des Formulars in einer Einzelansicht präsentierte Meldung dritten Personen per E-Mail zukommen lassen.

[16] Diese Möglichkeit wird vor allem auf den Seiten der Administration der Site eingesetzt, siehe hierzu den exemplarischen Einsatz des Frameworks in Abschnitt 3.

Dazu gibt er die E-Mail-Adressen der Empfänger und seine eigene E-Mail-Adresse an und schickt das Formular ab. Zusätzlich kann er seinen eigenen Namen und einen Zusatztext mitverschicken. In diesem Formular müßte auf jeden Fall die Gültigkeit der E-Mail-Adressen überprüft und dem Nutzer eine entsprechende Rückmeldung erteilt werden. Nach dem Abschicken des Formulars sollte, wenn die Prüfung erfolgreich war, ebenfalls eine Zusammenfassung in Form einer Einzeldarstellung der verschickten E-Mail und der Empfänger- und Absenderadressen auf der Folgeseite erscheinen.

Abbildung 1.8: Bestandteile; Formular

1.3.5 Intros und Designerseiten

Ein ebenfalls anzutreffendes Element in Sites ist reines Design. Hiermit meine ich eigentlich inhaltslose Seiten, aus denen zum Beispiel sogenannte Introtunnel [Siegel 1996] bestehen. Diese Seiten sollen den Nutzer neugierig auf die folgende Site machen[17]. Ein anderer Einsatzbereich für stark designlastige Seiten[18] sind die in Abschnitt 1.1 vorgestellten Imagesites. Hier

[17] Bei Sites, die zum täglichen Gebrauch bestimmt sind, stören sich viele Nutzer an derartigen Designelementen, da sie Zeit stehlen, ohne einen direkten Nutzen zu erbringen.

[18] Solche Seiten werden häufig mit *Flash* der Firma *Macromedia, Inc.* (`http://www.macromedia.com` [wwwProduct:Macromedia]) oder aufwendigen HTML-Codierungen [Ibañez, Zee 1998] erstellt.

kommt es nicht auf Informationsvermittlung an, sondern auf die Vermittlung eines besonderen Corporate Images. Da solche Seiten meist außerhalb der eigentlichen Site stehen (Introbereich, Extrobereich, u.ä.), werden sie typischerweise nicht in Verwaltungssystemen behandelt, so auch nicht in dem System, das in dieser Arbeit entworfen wird (Kapitel 2).

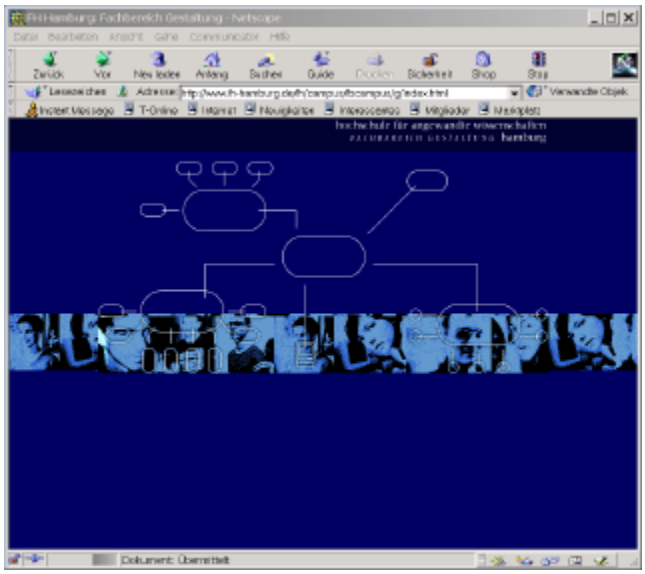

Abbildung 1.9: Bestandteile; reines Design

Als Beispiel einer reinen Designseite ist hier die Introseite des *Fachbereiches Gestaltung* der *Hochschule für Angewandte Wissenschaften Hamburg* aufgeführt. Die in der Abbildung 1.9 gezeigte Seite verschwindet — gesteuert über Javascript — nach 12 Sekunden automatisch, danach wird die eigentliche Site des Fachbereiches geladen.

1.4 Rollen bei großen Websites

Bei der Erstellung, Wartung und Benutzung einer Site sind verschiedene Rollen zu besetzen. Die Rollen sind verschiedenen Phasen im Leben einer Site zuzuordnen. Die Rolle des Betreibers ist allerdings keiner einzelnen Phase zuordbar.

Betreiber Die Erstellung und der Betrieb einer Site erfordern immer Investitionen, sowohl monetärer Art als auch in Form von Engagement. Die Betreiber stellen diese Investitionen bereit, um ein Ziel zu erreichen. Dieses Ziel ist die Mission einer Site aus der Sicht der Betreiber. Die Mission kann sehr unterschiedlicher Natur sein, vom kommerziellen Interesse bis hin zur politischen Meinungsbildung.

1.4.1 Rollen während der Erstellung

In der Anfangsphase einer Site, noch bevor die Site veröffentlicht wird, sind die folgenden Rollen die Hauptakteure. Während einer Strukturänderung oder eines Relaunches treten sie ebenfalls auf den Plan. Einige typische Fragestellungen dieser Akteure sind im Anhang C aufgeführt.

Informationsarchitekten Als erster Schritt kommt die Klärung der Mission und der Vision einer Site sowie der Abgleich der Interessen der Betreiber und der Nutzer der Site auf den Informationsarchitekten zu. Die Organisation der Inhalte einer Site, der Entwurf einer Navigation und des Bezeichnungssystems und die Festlegung notwendiger Funktionalitäten einer Site sind die dann folgenden Aufgaben.

Gestalter Die Präsentation der vom Informationsarchitekten gemachten Vorgaben ist die Aufgabe der Gestalter[19]. Die Gestalter müssen eine für den Nutzer geeignete, der Corporate Identity[20] und dem Corporate Image[21] des Betreibers entsprechende und dem Inhalt angemessene Form entwickeln, die von den Programmierern in eine produzierbare Site umgesetzt werden kann.

Programmierer Es gibt zwei Arten von Programmierern. *HTML-Programmierer* sind für die Umsetzung der von den Gestaltern erstellten Vorgaben verantwortlich. Wenn die Site per Hand gebaut wird, so muß der HTML-Programmierer alle von den Redakteuren erstellten Inhalte nach den Vorgaben der Gestalter umsetzen. Wenn aber eine Art von Redaktionssystem eingesetzt wird, so müssen nur einmalig Vorlagen erschaffen werden, die dann in einem automatisierten Prozeß mit den Inhalten der Redakteure verknüpft werden und so die Seiten für die Nutzer bilden.

Für die *Anwendungsprogrammierer* besteht die Aufgabe darin, einerseits Werkzeuge für den laufenden Betrieb bereitzustellen und andererseits interaktive Teile der Site für die Nutzer zu erstellen. Die Werkzeuge für den laufenden Betrieb sind für die Redakteure der Site gedacht. Sie ermöglichen die Pflege der Inhalte der Site. Diese Werkzeugerstellung umfaßt meist den Entwurf von Datenbanken und die Erstellung serverseitiger Programme, die die Daten mit den Vorlagen der HTML-Programmierer zum Versand an den Browser der Nutzer aufbereiten. Ebenso wird die Bereitstellung von Eingabemöglichkeiten zur Datenerfassung für die Redakteure, die Programmierung von Datenverarbeitungen sowie die Unterstützung des Arbeitsablaufes für die Redakteure durch Programme notwendig sein. Hierbei sollte auf eine

[19] Ich verwende hier nicht den Begriff Designer, da die Verwechslungsgefahr mit dem Softwaredesign zu groß ist.

[20] Selbstsicht einer Firma, Organisation oder Betreibers.

[21] Aussenwirkung einer Firma, Organisation oder Betreibers.

gute Umsetzung der Norm für *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten* [EN-ISO 9241] geachtet werden.

1.4.2 Rollen während des Betriebes

Wärend des laufenden Betriebes liegt die Hauptarbeit bei den Redakteuren.

Redakteur Die Redakteure erstellen Inhalte und geben sie entweder an die Programmierer weiter, oder pflegen sie in das von den Programmierern erstellte Redaktionssystem oder ein fertiges Content Management System[22] ein. Ebenfalls müssen die Redakteure gegebenenfalls die von Nutzern generierten Inhalte überprüfen und gemäß den Richtlinien der Betreiber der Site auf Verstöße gegen Regeln oder geltendes Recht reagieren. Je nach Organisation des Arbeitsablaufes (Workflow) gibt es *Chefredakteure*, die Themenvorgaben erstellen und an die Redakteure verteilen, bzw. *Chefs vom Dienst* (CvD), die die Inhalte vor der Publikation genehmigen müssen.

Administrator Die regelmäßige Sicherung der Inhalte einer Website (Datenbanksicherungen, Dateisystemsicherungen), die Kontrolle der Erreichbarkeit, die Pflege des Betriebssystems und der Software der eingesetzen Maschinen sowie das Auditieren von Sicherheitslogfiles und das Ergreifen von eventuell notwendigen Maßnahmen gegen Hackingversuche sind die Aufgaben eines Administrators. Die Gewährleistung der Kommunikation mit sogenannten Legacy-Systemen (siehe Abschnitt 4.5) fällt ebenfalls in diesen Aufgabenbereich.

1.4.3 Rollen während der Benutzung

Auf Seiten der Benutzung einer Site gibt es verschiedene Ausprägungen des Nutzers.

Nutzer Zunächst gibt es auf allen Sites *anonyme Nutzer*. Dies sind schlicht Personen, die mit einem Browser Kontakt zum Server der Site aufnehmen und via `HTTP` *(Hypertext Transfer Protokoll)* einzelne Seiten abfordern. Jeder Seitenaufruf via `HTTP` ist, da das Protokoll sitzungslos arbeitet[23], prinzipbedingt nicht auf einen vorhergehenden Abruf rückführbar. Erst durch den Einsatz von Sessionmechanismen innerhalb der Site (siehe Abschnitt 2.3.11) wird aus einem anomymen Nutzer ein *identifizierter Nutzer*. Jede Site, die individualisierte Dienste anbieten möchte, muß einen Identifikationsmechanismus für die Nutzer einsetzen. Sollen die individualisierten Dienste Persistenz

[22] Auch sogenannte fertige CMS benötigen meist eine individuelle Anpassung (Programmierung), die oft einen hohen Kostenfaktor darstellt.

[23] Zumindest in seiner ersten, immer noch im Einsatz befindlichen Version `HTTP/1.0` [wwwCite:RFC1945].

von einem Besuch der Site zum nächsten aufweisen, so ist eine Authentifikation des Nutzers erforderlich. Aus dem identifizierten Nutzer wird dann ein *authentifizierter Nutzer*, über den Daten im System der Site gespeichert und dem besondere Dienstleistungen angeboten werden können. Die Fähigkeit, Nutzer zu authentifizieren, ist die Voraussetzung für die Einführung von Prämiumdiensten für besondere Nutzer. Eine besondere Art des authentifizierten Nutzers ist der *Prämiumnutzer*, dieser besitzt weitergehende Rechte als ein normaler anonymer, identifizierter oder authentifizierter Nutzer. Diese Rechte können die Nutzung weiterer Anwendungen der Site sein, oder auch schlicht der Zugriff auf weitere, meist hochwertigere Inhalte.

Maschinelle Nutzer Hier sind Agentensysteme und Verfahren zur Content Syndication zu nennen. Dies sind Programme, die mit dem System kommunizieren, um Informationen auszutauschen. In dieser Arbeit stehen allerdings die menschlichen Nutzer im Vordergrund, obwohl mit der eingesetzten Technik auch automatisierte Kommunikation mit anderen Systemen implementiert wurde[24].

1.4.4 Rollen im Überblick

Zusammenfassend nochmal eine Darstellung der verschiedenen Rollen und deren Aufgaben. Auch das Zusammenspiel der Aufgaben, vor allem während der Erstellung, wird in der nachfolgenden Abbildung 1.10 dargestellt.

Einige der dargestellten Rollen werden oft von einem Menschen in Personalunion ausgefüllt. So ist der Gestalter oft auch gleichzeitig der HTML-Gestalter oder der Betreiber gleichzeitig Chefredakteur und Informationsarchitekt. Häufig füllen auch Anwendungsprogrammierer in Zusammenarbeit mit einem Chefredakteur und einem Gestalter die Rolle des Informationsarchitekten aus.

[24] Siehe hierzu Abschnitt 1.7.1 und 1.7.3

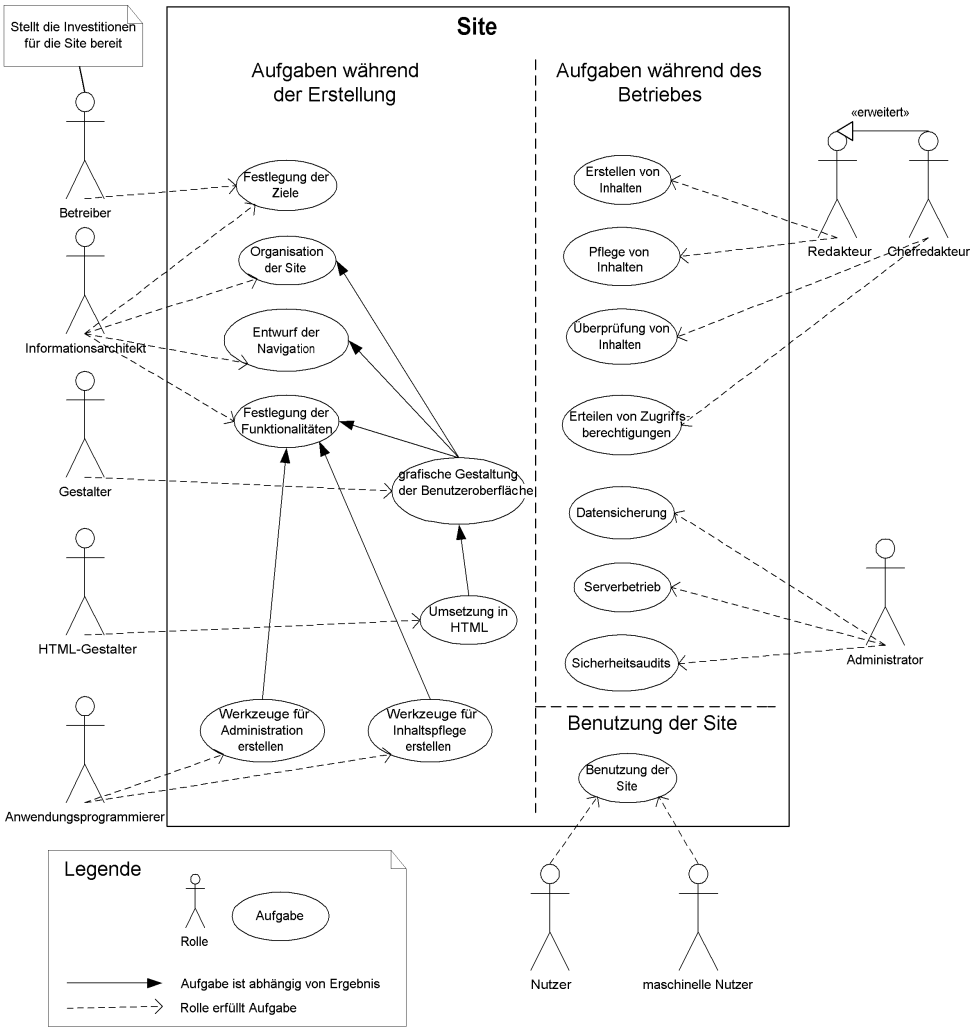

Abbildung 1.10: Rollen und deren Aufgaben

1.5 Qualitätsanforderungen an Sites

Die Qualitätsanforderungen an Sites sind sehr vielfältig. Hier seien deshalb nur die wichtigsten Forderungen für Sites genannt, deren Hauptaufgabe die Vermittlung von Informationen ist. In diese Kategorie fallen letztlich der Großteil der Archetypen aus dem Abschnitt 1.1, vor allem Informationsanbieter, Shoppingsysteme und Kommunikationsplattformen.

- **Usability** Die Benutzer der Site müssen die Informationen finden können. Die Navigation und die Orientierung innerhalb der Site müssen einfach und übersichtlich sein. Die Bezeichnungen der Bereiche und Elemente der Site müssen verständlich und eindeutig sein.

- **Konsistenz** Alle Verweise innerhalb der Site müssen korrekt sein, so dürfen Archivseiten beispielsweise nicht einfach gelöscht werden, ohne daß alle Verweise auf diese Seiten ebenfalls gelöscht werden. Auch Verweistexte müssen konsistent sein. Beispielsweise müssen die Titel von Nachrichten sowohl in der Auswahlliste als auch in der Einzeldarstellung der Nachricht gleich sein.

- **Aktualität** Die Informationen in der Site müssen aktuell sein. Ein Terminkalender, in dem vollmundig die Anmeldung zu einer längst vergangenen Aktion beworben wird, ist nicht aktuell, ebensowenig wie eine Linkliste in der sich Links auf nicht mehr existente Seiten befinden.

Die aufgeführten Forderungen sind eigentlich eine Selbstverständlichkeit. In der Realität jedoch leiden sehr viele Sites bzw. deren Benutzer an Problemen mit diesen essentiellen Forderungen.

Viele Sites enthalten sogenannte ‚broken links', das sind Querverweise innerhalb der Site, bei denen das Verweisziel nicht vorhanden ist. Der Querverweis zielt also ins Leere. Das ist ein typisches Konsistenzproblem.

Eine verbesserte Usability kann teilweise durch eine stärkere interne Verknüpfung der Site erreicht werden. Ein Beispiel hierfür sind Nachrichtenbereiche, bei denen in der Vollansicht einer Nachricht die anderen Nachrichten als Liste der Überschriften mit dargestellt werden[25]. Der Leser der Nachrichten muß somit nicht mehr erst zu einer Übersichtsliste zurückkehren, um eine andere Nachricht zu lesen, sondern kann die Nachricht seines Interesses direkt anspringen. Eine solche stärkere Verlinkung führt allerdings gleichzeitig zu einem erhöhten Konsistenzproblem.

Der dritte Punkt, die Aktualität einer Site, ist häufig eine Frage dessen, wie einfach es einem Redakteur gemacht wird, eine neue Information in die Site einzubringen. Bei einer Pflege der Site „von Hand" ist dies sicherlich deutlich aufwendiger als bei der Nutzung der Eingabemasken von datenbankgestützten Sites, insbesondere wenn durch eine hohe interne Verlinkung viele Seiten aus der Sicht des Nutzers geändert werden.

Die für die Erstellung und Pflege der Site benutzten Werkzeuge (siehe nächsten Abschnitt 1.6) haben somit einen entscheidenden Einfluß auf die Erfüllbarkeit der Forderungen. So ist die Konsistenzwahrung mit Hilfe von einfachen Editoren beispielsweise sehr schwierig und viele, die Usability verbessernde Ideen sind mit der Hilfe von einfachen Siteverwaltungstools sehr aufwendig. Erst die datenbankgestützten programmierten Websites erlauben es, bei mittleren und großen Projekten alle drei Forderungen mit vertretbarem Aufwand vollständig zu erfüllen.

[25] Siehe auch Abbildung 1.8 auf Seite 18.

1.6 Werkzeuge

Welche Werkzeuge stehen für den Betrieb einer Site zur Verfügung? In den folgenden Abschnitten werden verschiedene Werkzeuge vorgestellt und im Hinblick auf die Eignung für die Erstellung großer Sites überprüft. Zu diesem Abschnitt befinden sich im Anhang E die zugehörigen Internetquellen der Hersteller im Überblick.

1.6.1 Editoren

In Editoren werden einzelne Seiten einer Site erstellt. Ein Editor ‚weiß' nichts von der Existenz einer ganzen Site. Aus diesem Grunde ist der Einsatz von Editoren — zumindest für große Sites — meist nur auf die Phase der Erstellung (Programmierung) der Site beschränkt, für die Pflege der Inhalte müssen dann andere Werkzeuge bereitgestellt werden.

ACSCII-Texteditoren Die puristischsten Werkzeuge für die Erstellung von Websites sind Texteditoren[26]. Es gibt hier auch für die Erstellung von Webseiten spezialisierte Editoren mit einem erheblichen Vorrat an Werkzeugen zur Vereinfachung des Codierens. Exemplarisch sei hier *HomeSite* von *Macromedia, Inc.* (vormals von *Allaire Corporation*) [wwwProduct:Allaire] genannt.

Auch wenn bei der Erstellung von großen Websites sicherlich von den Programmierern Editoren eingesetzt werden, so ist doch zum Betrieb einer großen Site der reine Editoreinsatz keine praktikable Lösung. Dies gilt insbesondere, wenn ein Workflow zu berücksichtigen ist oder wenn eine Arbeitsteilung zwischen Gestaltern und Redakteuren vorliegt. Bei Einsatz von Arbeitsteilung sind erhöhte Wartezeiten und fehlerträchtige Übergaben von Arbeitsschritt zu Arbeitsschritt zu erwarten. Alternativ sind teure Allrounder[27] zu beschäftigen oder eine verminderte Qualität der Gestaltung oder der Inhalte zu erwarten. Ebenfalls ist die Komplexität der zu erstellenden Site stark eingeschränkt, da bei hoher Komplexität der Site beim Einfügen von nur einer Information oft sehr viele Seiten geändert werden müssen und dies einen sehr hohen Aufwand bedeutet.

WYSIWYG-Editoren Diese Art der Editoren behebt im Vergleich mit den Texteditoren nur das Problem der verminderten Gestaltung der einzelnen Seiten. Hierbei muß allerdings die teilweise mit erheblichen Mängeln behaftete HTML-Codeerzeugung der Editoren beachtet werden. *HoTMetaL* von *SoftQuad Software, Ltd.* [wwwProduct:Softquad] und *Dreamweaver* von

[26] Ich möchte hier nicht in den teilweise fast religösen Streit um den besten Editor einfallen. (Fast) jeder Editor hat seine Berechtigung und seine Nutzer.

[27] Personen, die sich sowohl mit der Erstellung redaktioneller Inhalte, als auch mit HTML und Programmierung auskennen.

Macromedia, Inc. [wwwProduct:Macromedia] sind Beispiele für WYSIWIG-Editoren, die kommerziellen Siteverwaltungstools sind ebenfalls in ihrer Funktionalität erweiterte WYSIWIG-Editoren.

Generell besteht beim Einsatz von Editoren für den Betrieb einer Site das Problem, daß die Wahrung eines Corporate Designs der Site in der Verantwortung der Ersteller einer jeden Seite liegt und hier eher Fehler auftreten können. Ebenso ist die Änderung der Struktur der Site oder gar ein Relaunch mit einem erheblichen Aufwand verbunden, da die Inhalte nicht von der Gestaltung getrennt vorliegen[28].

Einschub: Mit Hilfe von *Server Side Includes* [29] lassen sich bei mit Editoren handgebauten Seiten zumindest Teile der Pflege von seitenübergreifenden Elementen vereinfachen. So bietet es sich an, Seitenkopf, Seitenfuß und die Navigation in separaten Dateien zu pflegen, um so bei Veränderungen nicht alle Seiten einzeln anpassen zu müssen.

1.6.2 Siteverwaltungstools

In dieser Gruppe von Werkzeugen befinden sich meist in ihrer Funktionalität erweiterte WYSIWYG-Editoren. Typische Vertreter sind:

- *Frontpage* von *Microsoft Corporation* [wwwProduct:Microsoft],
- *Fusion* von *NetObjects, Inc.* [wwwProduct:NetObjects] und
- *GoLive* von *Adobe Systems Inc.* [wwwProduct:Adobe].

Allen Siteverwaltungstools gemeinsam ist die Möglichkeit, die Gestaltung von Basiselementen der einzelnen Seiten durch die Nutzung von Templates siteweit konsistent zu halten. Dies erleichtert die Durchsetzung eines Corporate Designs. Ebenfalls werden die automatische Verlinkung und Linknachführung bei Umstrukturierung der Site unterstützt. Was allerdings nicht unterstützt wird, ist die automatische Verteilung von Informationselementen auf andere Seiten der Site.

In der Diplomarbeit von Jan Pieper [Pieper 2000] wird ebenfalls ein Siteverwaltungstool vorgestellt, mit dem die Linkkonsistenz innerhalb der Site unterstützt wird.

1.6.3 Datenbankgestützte programmierte Websites

In dieser Klasse von Werkzeugen befinden sich die — für die Erstellung und den Betrieb von großen und komplexen Sites — einzig geeigneten Lösungen.

[28] Eventuell kann hier durch den Einsatz von `XML` und `XSLT` eine Trennung herbeigeführt werden, aber damit bewegt man sich im Bereich der programmierten Websites (siehe Abschnitt 1.6.3).

[29] Eine vom Apache-Webserver eingeführte Technik, bei der eine einzelne Seite aus unterschiedlichen Komponenten vom Webserver vor der Auslieferung zusammengefügt wird.

Es handelt sich hierbei um serverseitig laufende Programme, die alle Seiten einer Site aus Datenbankinhalten und Templates oder programmierten Objekten (siehe Abschnitt 2.3.5) zusammensetzen.

Erstmals ist es mit dieser Klasse von Werkzeugen möglich, individualisierte Sites zu erstellen oder auch echte Anwendungen den Nutzern einer Site zur Verfügung zu stellen.

Individuell programmierte Lösungen Hier wird im Prinzip für jeden Seitentypus einer Website ein kleines Programm erstellt, das die jeweiligen Datenbankinhalte abfragt und entsprechend aufbereitet an den Browser der Nutzer schickt. Oft wird auch für die Redakteure eine webbasierte Verwaltungsoberfläche für die Inhalte der Datenbank — und somit der Site — bereitgestellt.

Größter Nachteil einer solchen Lösung ist häufig die Durchmischung von Anwendungslogik und Gestaltung bzw. die Verteilung einzelner Komponenten der Gesamtgestaltung (Corporate Design) der Site auf viele Seiten. Hieraus resultiert eine stark eingeschränkte Wiederverwertbarkeit der Entwicklungen. Ebenso ist die Wartbarkeit bei Änderung der Datendefinition schwierig, da diese Informationen ebenfalls an verschiedenen Stellen redundant vorhanden sind.

Auf Frameworks basierte Lösungen Bei den auf Frameworks basierten Lösungen werden für die Teilprobleme bei der Entwicklung von großen Sites fertige Komponenten bereitgestellt, bei denen auf verschiedene Werte geachtet wurde. So sollte die Trennung von Inhalt, Gestaltung und Anwendungslogik vorgenommen worden sein. Damit das Rad nicht stets neu erfunden wird, sollten Komponenten wie Sessionmanagement, Rechteverwaltung, Useranmeldung, Formularhandling, etc. vom Framework bereitgestellt werden. Ebenso sollte es möglich sein, größere Anwendungen mit dem Framework zu entwickeln und diese in verschiedenen Sites einzusetzen.

Es gibt eine ganze Reihe von auf programmierten Lösungen basierenden kommerziellen Produkten. Hier seien exemplarisch die Produkte und Services der Firmen

- *Vignette Corporation* [wwwProduct:Vignette],
- *Gauss Interprise AG* [wwwProduct:Gauss] und
- *CoreMedia AG* [wwwProduct:Coremedia]

genannt. Auch die Applikationsserver bzw. -enabler der Firmen

- *IBM Corporation (WebSphere)* [wwwProduct:Websphere],
- *Macromedia, Inc. (ColdFusion)* [wwwProduct:Macromedia] und
- *BEA System, Inc. (Weblogic)* [wwwProduct:Bea]

sind hier zu nennen.

Unter `http://www.contentmanager.de` [wwwCite:ContentManager][30] findet sich ein Portal zum Thema mit vielen weiteren Produkten im Vergleich.

1.6.4 Zusammenfassung Werkzeuge

Während Editoren (Abschnitt 1.6.1) und Siteverwaltungstools (Abschnitt 1.6.2) nur für die Erstellung und den Betrieb von kleinen, nichtpersonalisierten und kaum interaktiven Sites mit geringer Komplexität geeignet sind, ist die Königsklasse der Werkzeuge, die datenbankgestützten programmierten Websites (Abschnitt 1.6.3), in der Lage, alle Wünsche nach Personalisierbarkeit, Interaktivität und Komplexität einer Site zu erfüllen. Hierbei ist der Einsatz von Frameworksystemen — wie überall in der Programmierung — von Vorteil, da damit zeitsparend auf bewährte Komponenten aufgesetzt werden kann. Allerdings begibt man sich mit dem Einsatz eines Frameworks auch in die vom Framework vorgezeichneten Grenzen und verliert etwas an Flexibilität.

1.7 Erfahrungen mit großen Webauftritten

„Scheitern, wieder scheitern, besser scheitern.“
Samuel Beckett

Nachdem in den Abschnitten 1.1 bis 1.6 zunächst allgemeine Begrifflichkeiten rund um Websites eingeführt und erläutert wurden, beginnt in diesem Abschnitt die konkrete Beschäftigung mit dem in dieser Arbeit zu erstellenden Verwaltungssystem für mittelgroße Websiten bzw. dem Framework für die Konstruktion eines solchen Verwaltungssystems.

Die in dieser Arbeit besprochenen Techniken und Technologien sind aus dem praktischen Einsatz heraus entstanden. Für das letztlich entstandene Framework waren einige Projekte[31] — und Erfahrungen daraus — von besonderer Bedeutung. Im Folgenden wird deshalb eine anonymisierte Projektgenealogie aufgezeigt, in der die jeweils gelernten Lektionen für das Framework angeführt werden. Ebenfalls werden Blockdiagramme der jeweiligen Systeme mit den — für das Framework wichtigen — Komponenten gezeigt. Die wichtigsten Ergebnisse der einzelnen Stationen waren:

- Abschnitt 1.7.1: Erprobung der Technologie, strukturierte Includes, Vereinheitlichung der Programmierweise
- Abschnitt 1.7.2: Erstes Framework mit Datenbeschreibungen

[30] Die Website ist aus der Diplomarbeit von Oliver Zschau [Zschau 1999] hervorgegangen.

[31] Die Projekte wurden unter Beteiligung des Autoren für unterschiedliche Auftraggeber realisiert. Alle Auftraggeber stammen aus der Branche der Medienunternehmen.

- Abschnitt 1.7.3: Session-, Nutzer- und Rechteverwaltung
- Abschnitt 1.7.4: Erste objektorientierte Fassung des Frameworks

1.7.1 Auftritt einer Sportzeitschrift

Zustand vorher Das Angebot gehörte zu der Klasse der datenbankgestützten programmierten Websites. Technologische Basis des Angebots war eine nichtrelationale Datenbank (im dBase Datenformat) mit der zugehörigen Skripting-Engine *HexBase* in der Version 1.0 der Firma *HEXMAC Software Systems AG* [wwwProduct:HEXMAC].

Der Umfang des Angebots war recht groß (je nach Saison ca. fünf bis acht Hauptbereiche) mit jeweils weiteren Unterbereichen, die wiederum teilweise weiter unterteilt waren. In allen Bereichen gab es Nachrichten, Termine, Spielberichte und Tabellen. Die Struktur der Site glich einem Baum, bei dem nur an den Blättern inhaltliche Informationen zu finden waren. Alle Verzweigungen (Seiten) unterhalb der Blätter waren reine Verteilseiten ohne redaktionelle Inhalte. Es gab keine Community- oder Personalisierungsfunktionen.

Die Site war von verschiedenen Programmierern im Laufe der Zeit aufgebaut worden, wobei häufig mittels Copy&Paste Programmteile übernommen und modifiziert wurden. Diese Evolution der Site führte zu einem gewissen Wildwuchs von verschiedenen Programmierstilen und -techniken, der nur noch schwer wartbar war. Das zu Grunde liegende Tabellenmodell litt ebenfalls unter evolutionären Altlasten und mangelnden Möglichkeiten der Datenbankengine.

Die Administration der einzelnen Bereiche wurde ebenso wie die Nutzerbereiche immer wieder neu programmiert, da die Funktionen nicht generalisiert waren. Selbst für eine neue Saison (beispielsweise in der Fußballbundesliga) mußten die Skripte kopiert und per Hand angepaßt werden.

Vor der Durchführung des Projektes stellte sich das System wie in der Abbildung 1.11 dar.

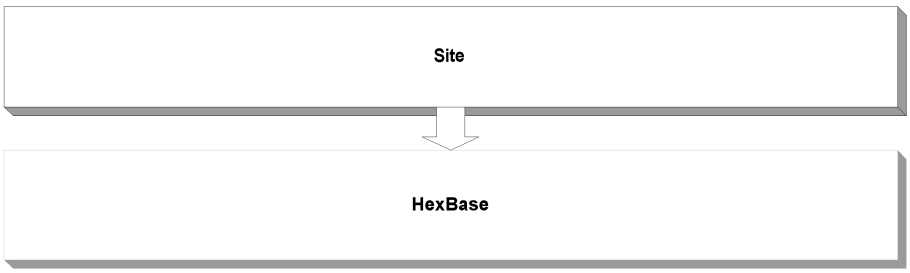

Abbildung 1.11: Genealogie; Es war keine modulare Struktur vorhanden

Gestellte Aufgabe Eine der Herausforderungen war, die Umsetzung der saisonal wechselnden Struktur der Site bezüglich des Wartungsaufwandes

einfacher zu gestalten. Hierbei war vor allem eine einfache Umsetzung zumindest auf der Administrationsseite gefordert. Auch größere fortlaufende Statistiken bzw. Historien verschiedener Sportarten sollten möglich werden. Desweiteren sollte eine höhere Verknüpfung innerhalb der Site erreicht werden, und zwar durch zentrale Funktionen wie einen Sportveranstaltungskalender und durch Ersetzen der Verteilseiten ohne Inhalt durch Zusammenfassungsseiten mit den redaktionellen Höhepunkten der jeweiligen Unterbereiche. Teilaufgabe war außerdem die Erstellung einer Schnittstelle zum automatischen Import und der automatischen Verteilung von Agenturnachrichten.

Die zweite große Aufgabe war die Entwicklung einer Community mit Messagingsystem, Foren, Newslettern und Chat. Im späteren Verlauf des Projektes wurde eine Communitysyndication mit einer befreundeten Sportsite durchgeführt, deshalb sollte die entwickelte Communitysoftware im Kern darstellungsneutral sein und nur die Darstellung auf der jeweiligen Site unterschiedlich programmiert werden.

Durchführung Da die bisher genutzte Technologie *HexBase* nicht leistungsfähig genug war, um die vielen Anforderungen zu erfüllen, mußte eine technologische Alternative gesucht werden. Allerdings mußte diese auch möglichst kostenneutral sein. Es wurden in einer kurzen Evaluationsphase verschiedene neue Technologien untersucht, und die Entscheidung fiel zugunsten der Kombination *Apache* [wwwProduct:Apache], *PHP* [wwwProduct:PHP] und *MySQL* [wwwProduct:MySQL].

Die Einführung vom relationalen Datenbankmodell für Mannschaften, Spiele, Turniere, Nachrichten, etc. und die Kategorisierung mittels relationaler Fremdschlüsselbeziehung ermöglichte eine wesentlich komplexere Informationsverarbeitung und eine viel stärkere interne Verknüpfung der Site. So konnte beispielsweise automatisch von den Spielergebnissen direkt zu den jeweiligen Mannschaftsdarstellungen verwiesen und umgekehrt alle Spiele einer Mannschaft mit ihren Terminen, Ergebnissen und Kommentaren dargestellt werden.

Es wurde ein strukturierter Ansatz für Kopf-, Fuß- und Navigationsdaten entworfen und eine Vereinheitlichung des Formularhandlings durchgeführt. Dies erleichterte die Wartung der Site erheblich. Ebenso wurde der Administrationsbereich der Site für die Redakteure vereinheitlicht, so daß beispielsweise neue Nachrichtenbereiche durch einfache Einträge in der Datenbank geschaffen werden konnten. Gleiches galt für bestimmte Turnierarten und Kategorien von Termindaten oder Linklisten.

Für die Community wurde ein System aus sogenannten Kernfiles und Files mit der Darstellungslogik entworfen. Damit sollte eine Trennung von Logik und Darstellung erreicht werden. Doch leider war in den Darstellungsfiles soviel Logik vonnöten, daß die Trennung nicht glückte. Die Community-

syndication scheiterte deshalb nach anfänglichen Erfolgen, denn die Pflege der verschiedenen Darstellungsfiles mit ihrer eingebetteten Logik für zwei verschiedene Sites war nicht zu koordinieren, so daß bei funktionalen Erweiterungen in der ersten Site auf den Seiten der zweiten Site zunehmend Fehler entstanden und deshalb die Syndikation letztlich eingestellt wurde.

Gelernte Lektionen Die Kombination *Apache*, *PHP* und *MySQL* erwies sich als tragfähig für kleinere und mittelgroße Projekte. Mittelgroß heißt hier, daß bis zu 2 Mio. Pageviews[32] pro Monat auf einfacher Hardware[33] bedient werden können[34] und das bei ca. 50.000 Seiten im Angebot und ca. 500 geänderten Seiten pro Tag. Hinzu kommt der Betrieb einer Community mit Chaträumen, in denen bis zu 30 Chatter gleichzeitig eingeloggt waren. Der datenbankgetriebene Chat war die größte Belastung für die Hardware. Damit ist die Erprobung der Technologie in der Praxis mit hoher Zuverlässigkeit und Leistungsfähigkeit nachgewiesen.

Vereinheitlichte Strukturen und Datei-Includes für Kopf-, Fuß- und Navigationsdateien ermöglichen es auf einfache Weise, die Struktur der Site zu ändern.

Die Generalisierung von Funktionalitäten erleichtert die Wartung der Site enorm. Gleichzeitig wird die laufende Entwicklungszeit stark verkürzt und die Qualität der Anwendung gesteigert.

Nach Durchführung des Projektes stellte sich das System wie in der Abbildung 1.12 dar.

---

[32] In der werbetreibenden Industrie gibt es eine *Informationsgemeinschaft zur Feststellung der Verbreitung von Werbeträgern e. V.* (unter `http://www.ivw.de` [wwwCite:IVW] zu erreichen). Die Auswertung von Websites (Onlineobjekten) ist eine Aufgabe des IVW. Hierzu wird zwischen Hits, Pageviews und Visits unterschieden. Ein Hit ist der Abruf eines Webobjektes mit `HTTP` [wwwCite:RFC1945] [wwwCite:RFC2616], ein Pageview ist der Abruf einer Inhaltsseite im Webbrowser (die aus vielen Hits bestehen kann, z.B. Grafikdateien) und ein Visit ist der Besuch einer Site mit einer Reihe von zusammengehörigen Pageviews (im Grunde eine Session eines Benutzers).

[33] Ein Pentium III Rechner mit 700 MHz und 256 MB RAM für Apache und PHP, sowie ein Pentium III Rechner mit 600 MHz und 128 MB RAM für die Datenbank MySQL.

[34] Hierbei ist zusätzlich noch zu berücksichtigen, daß die Verteilung der Zugriffe über den Tag sehr stark schwankend war, mit deutlichen Spitzenzeiten von ca. 2 Stunden Länge und kaum Zugriffen in der Nacht. Siehe hierzu auch die Darstellung im Anhang A.

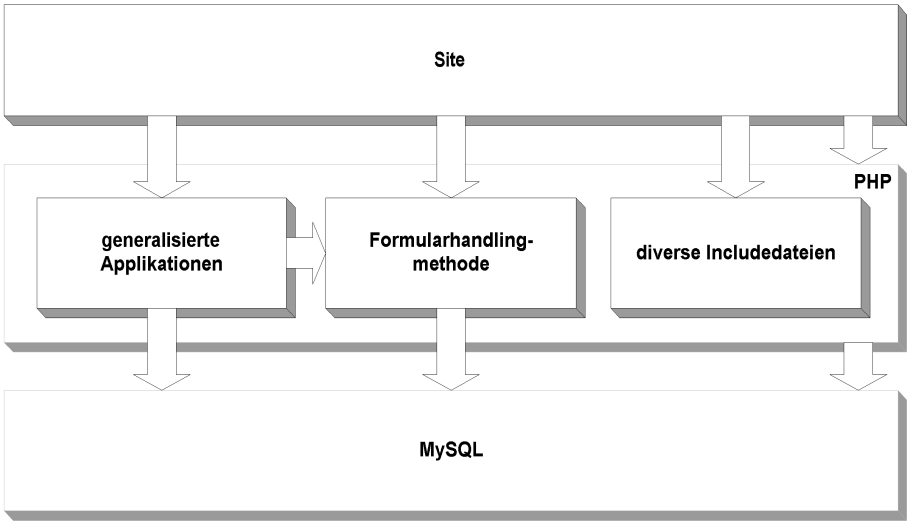

Abbildung 1.12: Genealogie; Neue Technologiebasis und erste Modularisierung

Aus dem Scheitern der Communitysyndication konnte gelernt werden, daß für die Syndikation von Applikationen ein erhöhter Aufwand in der Entwicklung zu treiben ist und daß einfache Includestrukturen mit prozeduralem Ansatz nicht ausreichend sind.

1.7.2 Auftritt einer Tageszeitung

Zustand vorher Ein existierendes Redaktionssystem auf Basis von tageweise publizierten HTML-Dateien war im Einsatz (programmiert in VisualBasic [wwwProduct:Microsoft]). Es war keine Datenhaltung über den Tag hinaus vorhanden und auch nicht nachrüstbar. Damit war das gesamte Nachrichtenarchiv der Site nur als eine Sammlung von — je nach Release des VisualBasic-Programms unterschiedlich strukturierten — HTML-Dateien vorhanden.

Da während des Tages die Ausgabe des nächsten Tages bearbeitet wurde, mußten die Daten des Vortages aus dem VisualBasic-Programm entfernt werden. Dadurch war es nicht möglich, während des Tages Aktualisierungen in der publizierten Site durchzuführen. Dies war insbesondere bei aktuellen Ereignissen während des Tages nicht tragbar. Ebenfalls war es wegen der fehlenden Daten nicht möglich, auf zurückliegende Nachrichten zuzugreifen und so Nachrichtenstränge zu bilden.

Durch die rein statischen Seiten war keine Interaktion mit dem Nutzer in Form einer Community möglich.

Gestellte Aufgabe Ein grafischer Relaunch unter Beibehaltung des Nachrichtenarchives und der sonstigen Artikel sollte durchgeführt werden. Gleich-

zeitig sollte die Funktionalität des Angebotes um während des Tages aktualisierbaren Inhalt erweitert werden.

In einer weiteren Ausbaustufe sollte die Errichtung einer eng mit dem Angebot verknüpften Community möglich sein, sowie eine Zusammenfassung von verschiedenen Nachrichten, Artikeln und externen Verweisen zu Themenblöcken, sogenannten Blickpunkten, eingerichtet werden. Desweiteren sollten die Inhalte der Site in einer Datenbank vorgehalten werden, um komplexere Verknüpfungen als bisher zu ermöglichen und auch einen späteren Relaunch unter Beibehaltung der Inhalte einfacher zu gestalten.

Durchführung Zunächst wurde eine Automatisierung des Imports der Archivdaten entwickelt. Nachdem die bisherigen und die laufenden Daten in der Datenbank abgelegt waren, wurde ein Verfahren für die Publikation der Tagesausgabe in statische Seiten erstellt (dies war gewünscht wegen der Indexierbarkeit durch Suchmaschinen). Zeitgleich fand die Entwicklung eines Communitykonzeptes und die Erstellung eines Redaktionssystems statt, in dem auch während des Tages Nachrichten nachgepflegt werden konnten. Eine Nachpflege der Inhalte während des Tages hatte einen zusätzlichen Publikationslauf von ca. 15 Minuten zur Folge, währenddessen sich die Site in einem teilweise inkonsistenten Zustand mit Broken-Links befand.

Zur Vereinheitlichung des Layouts und der Einhaltung der Designvorgaben bei gleichzeitiger einfacher Änderungsmöglichkeit wurde ein erstes Framework auf Basis von Datenbeschreibungen und Funktionsaufrufen erstellt[35].

Gelernte Lektionen Die Erzeugung von statischen Seiten (Caching) zum Zweck der Entlastung des Datenbankservers und der PHP-Maschine erwies sich als Hemmschuh bei der Entwicklung einer komplexeren internen Vernetzung der Site. So waren die entstandenen Themenblöcke nur als Übersichtsseiten zu realisieren oder als Rückverlinkung auf jeweils ältere Nachrichten. Die Aktualisierung aller Seiten eines Themenblockes hätte einen wesentlich komplizierteren (weil tagesübergreifenden) Publikationsalgorithmus erfordert, die Zahl der zu aktualisierenden Seiten stark nach oben getrieben und damit die Publikationszeit deutlich verlängert. Aus diesem Grunde wurde in folgenden Projekten auf den Einsatz von Caching-Methoden verzichtet (siehe auch Abschnitt 2.3.4).

Prinzipiell hat sich der Ansatz der Benutzung von Datenbeschreibungen für die verschiedenen Informationseinheiten der Website bewährt. Das erstellte Framework war aber wegen der mangelnden Datenkapselung und der Nichtvererbbarkeit auf die Verwendung in diesem speziellen Angebot

[35] Zu dieser Zeit war PHP 3.x aktuell und in dieser Version war noch kein brauchbares Objektmodell vorhanden.

beschränkt[36].

In der folgenden Grafik werden nur die Elemente dargestellt, die auch für die Entwicklung des Frameworks relevant sind. Insbesondere wird der Publishingmechanismus für die statischen Seiten außer acht gelassen.

Nach Durchführung des Projektes stellte sich das System wie in der Abbildung 1.13 dar.

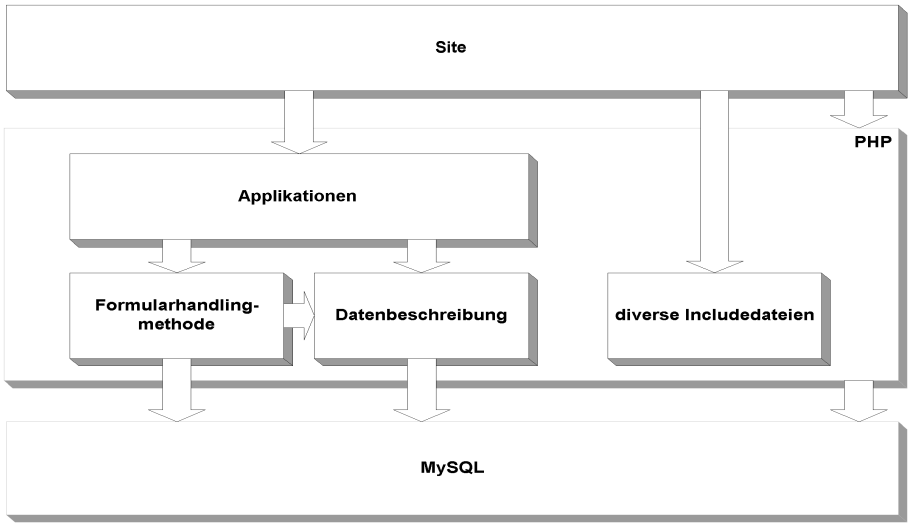

Abbildung 1.13: Genealogie; Erstes Benutzen von Datenbeschreibungen

1.7.3 Internes Redaktionswerkzeug

Zustand vorher Dieses Projekt entstand auf der grünen Wiese. Es war eine neue Redaktion mit Bedarf an einem zunächst internen Organisationswerkzeug.

Gestellte Aufgabe Die Entwicklung eines redaktionsinternen Werkzeuges zur Verwaltung von Aktienanalysen, deren Erfassung und Auswertung an mindestens drei verschiedenen Standorten zugleich und auf einer synchronen Datenbasis erfolgen mußte. Zusätzlich mußte eine Schnittstelle zu einem Webauftritt möglich sein (sie wurde später realisiert).

Um in dem Datenbestand eine hohe Qualität aufrecht zu erhalten, darf nicht beliebig an den Daten manipuliert werden. Es muß streng zwischen einer Reihe von verschiedenen Rollen mit verschiedenen Rechten unterschieden werden, z.B. erfassen Datatypisten die eintreffenden Analysen, und Redakteure werten die Daten aus. Ebenso gibt es Fachredakteure, die die zur

[36] Die Pflege des Frameworks überstieg bei der Übertragung auf das Projekt des Community Satelliten (Abschnitt 1.7.4) die Kapazitäten des Programmierteams und verwilderte dort zusehends (Entrophie im Programm [Hunt, Thomas 2000]).

Verfügung stehenden Aktien, Analysten, Märkte, etc. anlegen und betreuen. Es muß eine Historie über alle Änderungen am Datenbestand geführt werden und umfangreiche Recherchemöglichkeiten bereitstehen.

Es durften keine Lizenzkosten entstehen, da die Mittel für Lizenzen bereits erschöpft waren. Aus diesem Grund war der Einsatz von lizenzgebührfreien Open Source Komponenten obligatorisch.

Durchführung Aufbauend auf den Methoden der Sportzeitschrift wurde ein System entwickelt, in dem einzelne Bereiche mit Zugriffsrechten versehen waren. Diese Rechteverwaltung wurde in der Datenbank abgebildet und eine beliebige Anzahl von Nutzern konnte so am System mit individuell abgestimmten Rechten ausgestattet werden. Zur Wahrung und Gewährung der Rechte mußte ein Sessionmanagementsystem[37] mit zwangsweiser Nutzeranmeldung eingeführt werden, so daß der Nutzer jederzeit authentifiziert war.

Der automatische zeitnahe Abgleich der unterschiedlichen Redaktionsstandorte untereinander und mit dem Webauftritt (unter *WindowsNT* und mit *IIS/ASP/MSSQL* von *Microsoft* [wwwProduct:Microsoft] laufend) erforderte einen aufwendigen Synchronisationsmechanismus. Hierzu wurde ein dem Standard XML-RPC[38] ähnliches Verfahren eingesetzt.

Gelernte Lektionen Erstmals war ein umfangreiches System der User- und Rechteverwaltung nötig. Das entwickelte System wurde so flexibel gestaltet, daß es auf Änderungen im redaktionellen Alltag schnell angepaßt werden kann.

Nach Durchführung des Projektes stellte sich das System wie in der Abbildung 1.14 dar.

Die Durchführung eines Datenabgleiches mit Ausfallsicherheit und Resynchronisation war ein weiterer erfolgreich verlaufener Test für die Einsatzmöglichkeiten und die Flexibilität der Technologiebasis *Apache*, *PHP* und *MySQL*.

[37] In der damalig aktuellen PHP-Version war noch kein Sessionmanagement integriert.
[38] Siehe auch `http://www.xmlrpc.org/` [wwwCite:XML-RPC]

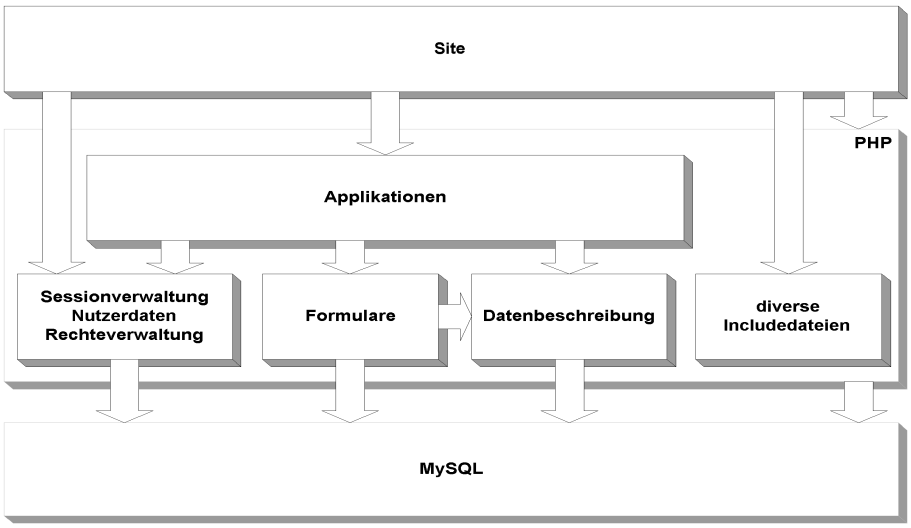

Abbildung 1.14: Genealogie; User- und Rechteverwaltung ist notwendig

1.7.4 Community Satellit

Zustand vorher Der Aufbau des Satelliten mit dem Framework aus dem Projekt Tageszeitung (Abschnitt 1.7.2) war zu 80% fertiggestellt, allerdings waren die gewünschten Funktionen des Satelliten nur schwer mit dem Framework realisierbar, so daß durch immer neue Anforderungen die Struktur des Frameworks stark verwilderte und kaum wartbar wurde. Die Errichtung der letzten 20% der Funktionalität der ersten Aufbaustufe des Projektes hätte weiterer großer Anstrengungen bedurft.

Gestellte Aufgabe In kürzester Zeit mußte die Fertigstellung und Aktivierung des Satelliten erfolgen (notfalls auch mit reduziertem Leistungsumfang).

Da das Framework aus Abschnitt 1.7.2 an seine Grenzen stieß, mußte ein neues, leistungsfähigeres Framework entwickelt werden und im laufenden Betrieb von der bisherigen Variante zur neuen migriert werden.

Durchführung Da die Grenzen des bisherigen Frameworks vor allem auf seiner prozeduralen Struktur beruhten, wurde für das neue Framework ein objektorientierter Ansatz aufgenommen. Zunächst wurde ein Userobjekt entworfen, als Basis für den Aufbau eines neuen Frameworks diente und zugleich den ersten Schritt zur Migration auf das noch zu erstellende Framework darstellte. Im Userobjekt war eine Sessionverwaltung und die Basis für eine Zugangsverwaltung und Authentifikation enthalten. Mit diesem Userobjekt wurden die auf dem alten Framework basierten Seiten zur Publikation fertiggestellt. Zeitgleich wurde am neuen Framework entwickelt. Nach Fertigstellung der ersten Teile wurde das bisherige Werk stückweise auf das neue,

jetzt objektorientierte[39] Framework umgestellt und in seiner Funktionalität erweitert.

Gelernte Lektionen Die Schwierigkeiten in der Nutzung des prozeduralen Frameworks führten zu der Erkenntnis, daß die prozedurale Struktur nicht genügend Flexibilität bot, um die gewünschte Site zu erstellen. Deshalb wurde ein erster Entwurf eines neuen Frameworks erstellt und eine Migration vom Paradigma der prozeduralen Programmierung auf das der Objektorientierung durchgeführt. Die erste objektorientierte Fassung des Frameworks erwies sich als tragfähig und der Ansatz wird somit weiterverfolgt.

1.8 Resultierende Anforderungen an das Framework

Aus den vier durchgeführten Projekten des Abschnittes 1.7 ergeben sich für das Framework einige Grundforderungen.

- Trennung von Logik, Inhalt und Gestaltung.
- Einfache Neudefinition von einfachen Anwendungen mittels Datenbeschreibungen. Einfache Erstellung der zugehörigen Administrationsbereiche.
- Bereitstellung einiger Standardausgabearten wie Liste, Tabelle, Formular und Einzeldarstellung
- Unterschiedliche standardisierte Formularhandlings.
- Zur Verbesserung der Wartbarkeit sind Redundanzen im Code streng zu vermeiden.
- Eine User-, Rechte- und Sessionverwaltung ist integraler Bestandteil des Frameworks.
- Die Möglichkeit, entstandene Applikationen eines Projektes in anderen Projekten auf einfache und geregelte Weise wiederzuverwenden.

Zur Realisierung der Forderungen ergeben sich fünf Funktionsblöcke, auf denen die Applikationen der zu erstellenden Site aufsetzen. Dies sind Formulare, Ausgaben, Datenbeschreibungen, Session-/Nutzer-/Rechteverwaltung und der Datenbanklayer. Die hauptsächlichen Beziehungen der angesprochenen Funktionsblöcke sind der folgenden Abbildung 1.15 zu entnehmen.

[39] In der zu diesem Zeitpunkt zur Verfügung stehenden PHP-Version 4.0 war erstmals ein brauchbares Objektmodell vorhanden.

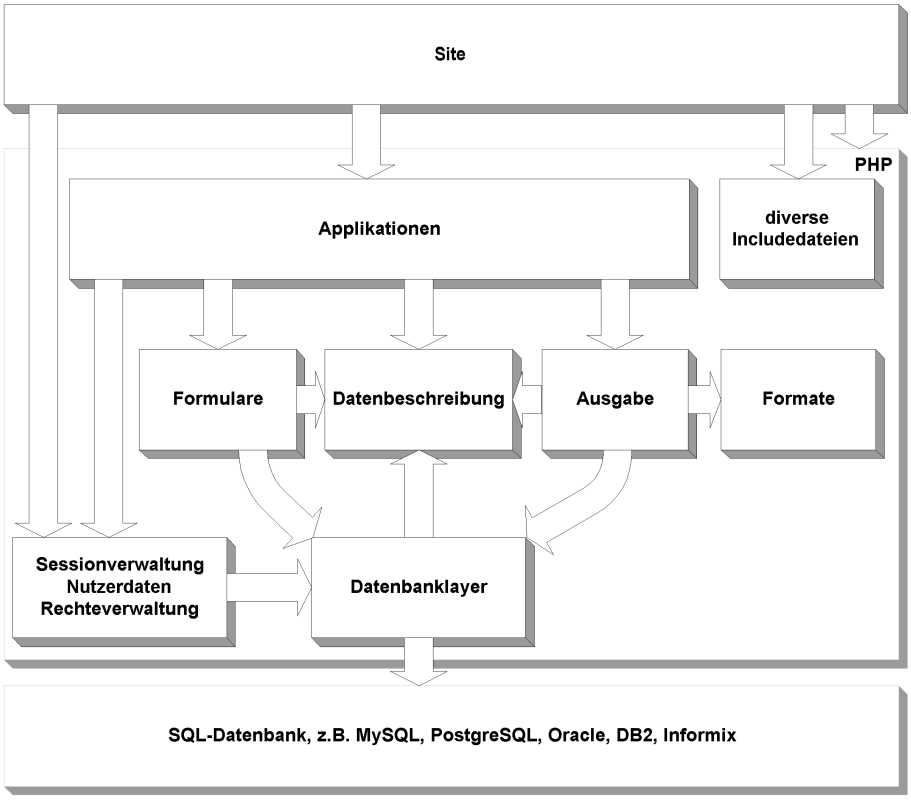

Abbildung 1.15: Funktionsblöcke des geforderten Frameworks

Eine Site besteht aus einer Anzahl von Applikationen (z.B. Nachrichten, Termine, Bildergalerie). Applikationen umfassen hierbei sowohl die für die Nutzer sichtbaren als auch die für die Pflege der Inhalte notwendigen Bereiche, die von den Redakteuren der Site genutzt werden. Ob ein Nutzer einen Bereich betreten darf und welche Elemente er dort zu Gesicht bekommt, hängt von den ihm erteilten Rechten ab. Hierzu ist eine Session- und Rechteverwaltung in Kopplung mit den Nutzerdaten gefordert.

Eine Applikation besteht aus Ausgaben und Formularen und beruht auf Datenbeschreibungen der — in der Applikation behandelten — Informationseinheiten (z.B. eine Nachricht, ein Termin, ein Bild). Die Daten der Informationseinheiten werden in einer Datenbank gelagert und sind über browserbasierte Formulare pflegbar. Die Datenbank soll möglichst austauschbar sein, deshalb greifen sowohl Eingaben mittels Formularen als auch Ausgaben als Einzeldarstellung, Listen- oder Tabellenform über eine wohldefinierte Schnittstelle mittels der Datenbeschreibungen auf die Datenbank zu.

Die Ausgabe der Daten und Formulare erfolgt über Ausgabemodule, die wiederum auf separate Formatierungsinformationen zurückgreifen.

Kapitel 2

Das Framework

„Es dünke mich etwas Herrliches, die Ursachen von allem zu wissen, wodurch jeglich entsteht, vergeht und wodurch es besteht.“
Sokrates

In diesem Kapitel wird das Framework entworfen und realisiert. Nach einer kurzen Darstellung des Software-Entwicklungsmodells folgt zunächst die Erfassung der Anforderungen (Abschnitt 2.2) und die Durchführung des Designs in einer Außensicht des Frameworks (Abschnitt 2.3). In der Umsetzung des Frameworks (Abschnitt 2.3.13) werden dann einige Implementierungsdetails in der Innensicht besprochen. Basis für die Überlegungen dieses Kapitels sind die Erfahrungen aus Abschnitt 1.7, insbesondere auch der dort entwickelte Rahmen aus Abbildung 1.15.

2.1 Zu Grunde liegendes Entwicklungsmodell

Die Basis des Frameworks wurde im Projekt des *Community Satelliten* aus Abschnitt 1.7.4 entwickelt. Dort wurde als Entwicklungsmodell eine Variante des *Rapid Application Developments*, das *Controlled Iteration Modell*[1] benutzt. Das Modell läßt sich durch folgenden Pseudocode beschreiben:

```
Analysiere die Basisanforderungen;
Entwickle ein initiales Design;
repeat{
  Entwickle ein vorfuehrbares Programm;
  Praesentiere es dem Kunden;
  Sammle Rueckmeldungen des Kunden;
  if(nicht fertig){
    Plane die naechste Iteration auf Basis der Rueckmeldungen;
  }
}until(fertig)
```

[1] siehe auch [Cantor 1998]

Die zwei wesentlichen Voraussetzungen für den Erfolg dieses Modells sind:

1. Es gibt einen effizienten Weg, um auf die Rückmeldungen des Kunden zu reagieren.
2. Die Entwickler und der Kunde einigen sich über das Abbruchkriterium.

Die Voraussetzungen waren erfüllt, da das Entwicklungsteam — bestehend aus 2 Personen: einem Festangestellten und einem freien Mitarbeiter — sehr klein war und gleichzeitig seinen eigenen Hauptkunden darstellt und eine enge Zusammenarbeit mit dem Auftraggeber (dem Vorgesetzten des Festangestellten) gegeben war. Zusätzlich war auf Grund der Erfahrungen aus den anderen Vorprojekten (Abschnitt 1.7) die Vision für das Framework in den Köpfen aller Beteiligten verankert.

Als Konsequenz des Entwicklungsmodells erscheinen die formulierten Anforderungen des nächsten Abschnittes 2.2 relativ vage, da sie die Basisanforderungen darstellen und erst im Laufe der Entwicklung die konkreten Anforderungen an das Framework formuliert und realisiert wurden.

Im Rahmen dieser Diplomarbeit wurde die Weiterentwicklung des Frameworks — vor allem in Bezug auf saubere informationstechnische, ingenieursmäßige Strukturen — und seine Dokumentation voran getrieben. Hierzu wurde die Klassenstruktur überarbeitet und die Verantwortlichkeiten der einzelnen Objekte durch Refaktorisierung [Fowler 2000] deutlicher hervorgehoben. Die Funktionalität des Frameworks wurde ebenfalls erweitert.

Die Ergebnisse der Entwicklung sind in der auf CD beigelegten API-Dokumentation des Frameworks festgehalten (siehe Anhang D) und werden im Abschnitt 2.3 *Design und Realisierung des Frameworks* zum Teil im Detail besprochen. Der exemplarische Einsatz des Frameworks in Kapitel 3 verdeutlicht ebenfalls einige Aspekte des Frameworks.

2.2 Anforderungen an das Framework

„Für den, der nicht weiß, welchen Hafen er ansteuert, ist jeder Wind der richtige Wind.“
Lucius Annaeus Seneca

Die Anforderungen des Abschnittes 1.8 bildeten die Basis für die Erarbeitung der Anforderungen an das Framework, die in diesem Abschnitt definiert werden.

Die im Folgenden aufgeführten Anforderungen lassen sich in drei Gruppen unterteilen:

- Abschnitt 2.2.1, nichtfunktionale Anforderungen
- Abschnitt 2.2.2 bis 2.2.7, Anforderungen allgemeinerer Natur
- Abschnitt 2.2.8 bis 2.2.11, konkrete Funktionalitäten

2.2.1 Nichtfunktionale Anforderungen

Preisgünstige Lösung

Um das Framework auf möglichst breiter Basis einsetzen zu können, muß eine preisgünstige Lösung gefunden werden. Dies gilt für alle drei Kostenbereiche: die Initialkosten, die Betriebskosten und die Wartungskosten.

Die Initialkosten setzen sich aus Lizenzkosten, Beratungsleistungen und initialem Programmieraufwand zusammen.

Die Betriebskosten werden durch Internet Service Provider und Servermieten bzw. den Kauf von Hardware sowie durch notwendige Administrationskräfte bestimmt. Ein weiterer Posten der Betriebskosten beinhaltet natürlich die Pflege der Inhalte des Auftrittes.

Die Wartungskosten decken eventuelle Erweiterungen und die Anpassung des Webauftrittes an geänderte Bedürfnisse ab. Auch Fehlerbeseitigungen können zu den Wartungskosten gerechnet werden[2].

Wiederverwendbarkeit von Strukturen

Die Forderung der Wiederverwendbarkeit erwächst direkt aus der Forderung nach Preisgünstigkeit. Jede Wiederverwendung erspart Programmierzeit und damit Kosten. Ebenso kann Wiederverwendung die Wartungskosten verringern, denn einmal gefundene Lösungen können erneut eingesetzt werden und Fehlerbeseitigungen greifen in mehreren Bereichen.

Generell sollte in den mit dem Framework erstellten Lösungen die notwendige Redundanz bei der Definition von Strukturen des Inhaltes oder des Layouts so gering wie möglich gehalten werden.

Auch Logiken bzw. komplette Applikationen — wie z.B. Communitymodule, Bildverwaltungen und ähnliches — müssen von einer Site auf eine andere unter Beibehaltung einer gemeinsamen Codebasis portiert werden können. Idealerweise können derartige Applikationen in den verschiedenen Sites individualisiert werden. Hierbei müssen vor allem die Erfahrungen aus dem Projekt der Sportzeitschrift (Abschnitt 1.7.1) und der gescheiterten Communitysyndication berücksichtigt werden.

Performance

Die Lösung muß im Hinblick auf den Performancebedarf so gestaltet sein, daß die Performance einer normalen[3] Rechnerausstattung für mittlere Auftritte[4] ausreichend ist. An folgenden Größen kann die Performance einer Website bzw. eines Webservers festgemacht werden:

[2] Dies hängt wesentlich vom Vertragsverhältnis zwischen dem Betreiber der Website und den Programmierern ab.

[3] Als normal wird in dieser Arbeit ein 700 MHz Pentium III mit 256 MB RAM Rechner betrachtet.

[4] 1 bis 20 Millionen Seitenabrufe pro Monat, je nach Komplexität des Seitenaufbaues.

- Geringe Latenzzeit für die Nutzer[5]
- Kurze Publikationszeit von neuen Inhalten
- Niedrige Serverlast bzw. große Anzahl von ausgelieferten Seiten pro Sekunde

Bemerkung: In dieser Arbeit können für viele Teilbereiche keine absoluten Werte angegeben werden, da die Bedürfnisse von Projekt zu Projekt doch sehr unterschiedlich sind. Wesentlich ist, daß auf Performance geachtet und dieses Thema bei der Lösung berücksichtigt wird. Es wäre z.B. eine mögliche Lösung, sämtliche Datenbeschreibungen in der Datenbank abzulegen und alle Ausgaben und Formulare vollständig generisch mit den Daten aus der Datenbank dynamisch zu erzeugen. Diese Lösung wäre sehr elegant und flexibel, aber sicher nicht sehr performant, da Datenbankzugriffe eine erhebliche Belastung für den Server darstellen.

Skalierbarkeit

Da viele Webprojekte zunächst klein anfangen und dann mit steigender Bekanntheit stetig größer werden, muß das System mit wenig Aufwand skalierbar sein.

Skalierbarkeit bezieht sich zum ersten auf Benutzerzahlen, also auf die Anzahl der angemeldeten Benutzer und der abgerufenen Seiten. Zum zweiten auf den Siteumfang, also der prinzipiell unterschiedlichen Seiten und den Informationsumfang, die Anzahl der in der Site präsentierten Informationen, wie z.B. Nachrichten.

2.2.2 Browserbasierte Administration

Da in dem Entstehungsumfeld des Frameworks eine sehr heterogene Rechnerstruktur vorhanden war[6], ist es notwendig, daß die Anforderungen an die Hard- und Software der Redakteursarbeitsplätze so klein wie möglich zu halten ist. Eine browserbasierte Administration ist gefordert. Weitere Vorteile der browserbasierten Lösung sind

- der einfache Zugang zur Administration über das Internet,
- keine zusätzlichen Kosten für die Distribution und Wartung der Redakteursarbeitsplätze,
- Systemunabhängigkeit der Redakteursarbeitsplätze

[5] Die Zeit, die ein Nutzer von der Anfrage auf Auslieferung einer Webseite bis zum Erscheinen der ersten Teile der Webseite warten muß, sollte 3 Sekunden nicht überschreiten. Hierzu zählen neben der Bereitstellungszeit durch den Webserver auch die Transferzeiten durch das Internet und die Darstellungszeiten durch den Webbrowser.

[6] Apple/Macintosh und Microsoft/Windows unterschiedlichsten Alters an verschiedenen Standorten und Heim- bzw. Außenarbeitsplätze mit unterschiedlichen Anbindungen an das Internet.

2.2.3 Trennung von Inhalten, Layout und Logik

Im Framework müssen Inhalte, Layout und Logik so strikt wie möglich voneinander getrennt sein. So muß beispielsweise die Darstellung einer Liste durch die Änderung einer Formatierungsanweisung modifizierbar sein. Diese Formatierungsanweisung darf aber nicht in den Logikcode einer Applikation eingemischt werden, sondern ist in separaten Dateien bzw. Objekten zu halten.

2.2.4 Zu bearbeitende Inhalte

Das Framework erhebt nicht den Anspruch, für alle möglichen Inhalte (siehe auch Abschnitt 1.2) Lösungen bereitzustellen, sondern beschränkt sich primär auf die im Folgenden aufgeführten Bereiche.

Pflege strukturierter Daten

Die Inhaltspflege und die Darstellung strukturierten Inhalts muß direkt unterstützt werden. Hierbei soll die Pflege und Darstellung auf die gleiche Datenbeschreibung der Struktur zugreifen. Diese Datenbeschreibung muß auch für die unterschiedlichen Darstellungsarten und Pflegeformulare der Informationen geeignet sein (Liste, Tabelle, Einzeldarstellung in der Administrations- und Nutzersicht, sowie Formulare für Administratoren und Nutzer). Die Pflege der Datenbeschreibungen muß nicht für die Redakteure möglich sein. Es reicht aus, wenn die Programmierer auf einfache Weise die Datenbeschreibungen anlegen können.

Pflege unstrukturierter Daten

Für unstrukturierte Daten gibt es in der ersten Ausbaustufe des Frameworks keinen direkten Bedarf. Allerdings sollte es möglich sein, in einer späteren Fassung des Frameworks auch unstrukturierte Daten umfassender bearbeiten zu können (siehe hierzu auch Abschnitt 4.7, WikiWeb).

Für den Einstieg sollte in der ersten Ausbaustufe des Frameworks das Editieren eines HTML-Teilbereiches mittels Formular als HTML-Quelltext möglich sein. In folgenden Ausbaustufen können dann Eingabefelder mit weitergehenden Formatierungs- bzw. Strukturierungsmöglichkeiten entworfen und in das Framework integriert werden.

Pflege der Sitestruktur und Navigation

Die globale Struktur der Site muß ebenfalls nicht von den Redakteuren veränderbar sein, allerdings darf eine Änderung in der Struktur nicht zur Folge haben, daß hunderte von Dateien angepaßt werden müssen. Die Navigation und Orientierung sind als eigene Komponenten zu definieren, die

dann im Kontext der jeweiligen Seite auf Grund von Parametern die korrekten Darstellungen liefern. Dieser Punkt ist aber nicht integraler Bestandteil des Frameworks, sondern eher allgemeiner Natur und mit Hilfe von Auslagerungen der entsprechenden Elemente in zentrale Dateien zu realisieren.

2.2.5 Erstellung einfacher Applikationen

Einfache Applikationen — wie eine Nachrichtenliste oder eine Linkliste — sind im wesentlichen mit Hilfe von einfachen Datenbeschreibungen zu erstellen. Die zur Verwaltung der Daten notwendigen Auswahllisten und Eingabemasken sind durch die Datenbeschreibungen zu steuern, d.h. für derartige Aufgaben ist keine oder nur wenig individuelle Programmierung notwendig.

2.2.6 Navigation durch größere Datenmengen

Für längere Auswahllisten oder -tabellen sind auf einfache Weise Navigationen wie seitenweises Vor- und Zurückblättern oder alphabetische Selektion einer Teilmenge zu realisieren. Diese Navigationen sollen siteweit einheitlich dargestellt werden (siehe auch Abschnitt 2.2.7).

2.2.7 Konsistenz des Layouts

Die grafische Gestaltung der Site muß ein einheitliches Bild ergeben (Corporate Identity). Hierzu sollten die einzelnen grafischen Elemente benannt und einheitlich gepflegt werden können. Diese Pflege muß ebenfalls nicht von den Redakteuren erfolgen, sondern sollte durch Programmierer auf einfache Weise siteweit möglich sein. Tabellen- und Listendarstellungen zur Auswahl von Inhalten sollten siteweit gleich bzw. ähnlich aussehen und auch an zentraler Stelle (mit wenigen Formatierungsanweisungen) gesteuert werden.

2.2.8 Datenbankzugriff

Der Datenbankzugriff ist unabhängig von der konkreten Datenbank zu halten.

Da im Web häufig Datenbankschlüssel als Bestandteil der URL auftauchen, sind stumme Schlüssel wie z.B. Datensatznummern von Vorteil. Ebenso ist die Verwendung von stummen Schlüsseln im Hinblick auf Modifikationen der Attribute der Datenbankentitäten (und damit der mit ihnen assoziierten Informationseinheiten) von Vorteil, da mit ihnen eine erhöhte Flexibilität erreicht wird.

Das Framework muß die Generierung und Verwendung von stummen Schlüsseln unterstützen.

2.2.9 User- und Rechteverwaltung

Bei jedem größeren Webprojekt existieren — wie in Abschnitt 1.4 dargestellt — unterschiedliche Rollen für Mitarbeiter und Nutzer der Site. Eines der am häufigsten wiederverwendbaren Teile des Frameworks stellt die User- und Rechteverwaltung dar. Bei jeder Website, in der von unterschiedlichen Autoren Beiträge geliefert werden, taucht schnell das Problem der Userverwaltung auf. Zuerst auf der Seite der Administration der Website. Auf der Nutzerseite wird eine User- und Rechteverwaltung in dem Moment relevant, wenn Nutzer sich aktiv an der Entwicklung der Site beteiligen können: sei es durch Communitytools, Personalisierung, oder weil für die Nutzer unterschiedliche Applikationen auf der Site angeboten werden.

Aus diesem Grund ist es sinnvoll, im Framework eine generalisierte User- und Rechteverwaltung als Basisapplikation vorzuhalten und ebenso entsprechende Methoden der Rechteabfrage zu standardisieren.

2.2.10 Sessionverwaltung

Aus dem Abschnitt 2.2.9 der User- und Rechteverwaltung entsteht notwendigerweise die Anforderung der Sessionverwaltung. Das System muß in der Lage sein, die durch die zustandslose Natur des HTTP [wwwCite:RFC1945] zwangsweise unabhängigen Seitenabrufe einer Serie von Abrufen durch einen Nutzer zuordnen zu können.

Die Sessionverwaltung des Systems sollte unterschiedliche Methoden der Sessionidentifikation ermöglichen. Als sinnvolle Methoden stehen Cookies und URL-Modifikationen sowie die Übertragung von SessionID's via HTTP-`get` oder -`post` zur Verfügung. Kombinationen aus Client-IP-Adresse und eventueller Client-Browser-Identifikation sind aus verschiedenen Gründen (z.B. Existenz von Proxyservern) nicht zuverlässig[7].

2.2.11 Formularhandling

Da Formulare das Hauptmittel sind, um auf Websiten interaktive Applikationen zu erstellen, müssen standardisierte Methoden des Formularhandlings Bestandteil des Frameworks sein. Hierbei müssen die Formulardaten auf Korrektheit und Konsistenz geprüft werden können und bei Fehlern dem Nutzer zur Bearbeitung wiedervorgelegt werden. Die Konsistenzprüfung muß hierbei für einzelne Felder (z.B. Plausibilitätsprüfung eines Kalenderdatums oder einer E-Mail-Adresse) und auch für ein Ensemble von Feldern (ob z.B. ein Startdatum vor einem Enddatum liegt) erfolgen können. Es sollten unterschiedliche Arbeitsabläufe mit Formularen realisiert werden können.

[7] Siehe hierzu auch den Working Draft zur *Session Identification URI* unter `http://www.w3.org/TR/WD-session-id.html` [wwwCite:W3C Session]

Die Standardvariante des Ablaufes ist in der Abbildung 2.1 dargestellt. Dieser Ablauf ist für die Neuanlage von Datensätzen ebenso geeignet wie für die Bearbeitung von existierenden Datensätzen.

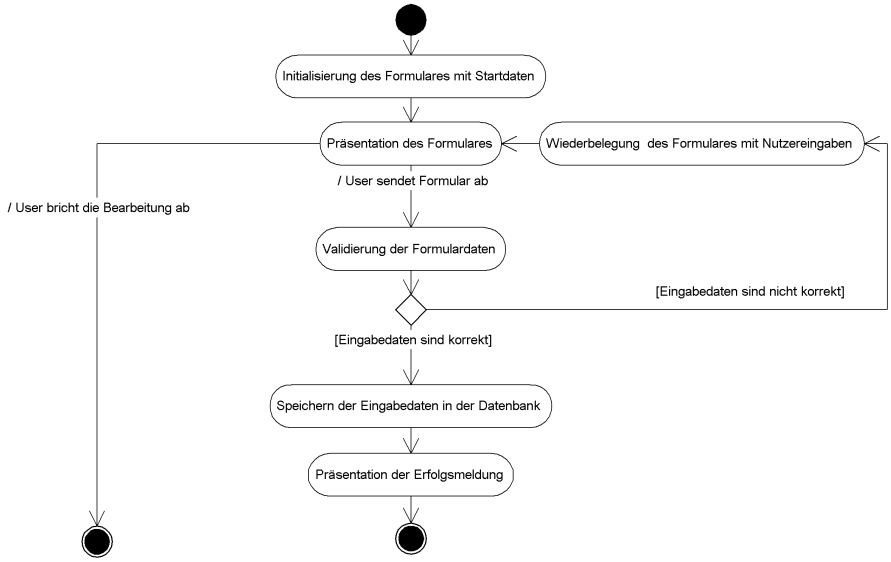

Abbildung 2.1: Ablaufdiagramm einer Standardformularbearbeitung

Validierungsfunktionen Um das Rad nicht jedesmal neu erfinden zu müssen, sind die Validierungsfunktionen für die einzelnen Felder eines Formulares so zu gestalten, daß sie wiederverwendbar sind. Auf diese Weise soll mit der Zeit ein Sortiment von Validierungsfunktionen für verschiedene Zwecke entstehen: E-Mail-Adressen, URL's, Kalenderdaten, Uhrzeiten, Währungsbeträge, Länderkennungen und weiteres.

2.3 Design und Realisierung des Frameworks

„Wenn es nur eine Wahrheit gäbe, könnte man nicht hundert Bilder über dasselbe Thema malen.“
Pablo Picasso

In diesem Abschnitt wird das Framework in seiner Außenansicht entworfen und beschrieben. Es werden ebenfalls einige Detailfragen der Implementierung angesprochen.

Es wird zunächst die dem Framework zu Grunde liegende Philosophie eingeführt (Abschnitt 2.3.1), dann folgen einige Grundentscheidungen (Abschnitte 2.3.2 bis 2.3.5). Die Vorstellung[8] der einzelnen Komponenten des Frameworks (Abschnitt 2.3.6 bis 2.3.11) erfolgt *bottom-up* nach dem Schema der Abbildung 1.15 auf Seite 38. Nach der Vorstellung aller Komponenten folgt ein Überblick über die Klassen des Frameworks (Abschnitt 2.3.12). Den Abschluß des Kapitels bildet der Abschnitt 2.3.14 mit einer Darstellung der möglichen Arten der Vererbung von Datenbeschreibungen.

2.3.1 Weltsicht des Frameworks

Die prinzipielle Weltsicht des Frameworks ist eine datenzentrierte Sicht. Dies ist deshalb so, weil das Framework serverseitig ausgeführt wird und im Zusammenspiel von HTTP-Server und HTTP-Browser immer vollständige HTML-Seiten (Datenpakete) ausgetauscht werden.

Auch die Vorgänge zur Erzeugung von HTML-Seiten beinhalten vor allem die Manipulation von Daten. Es werden Informationen (Daten) aus einer Datenbank gelesen, diese Informationen werden mit Layouts (Daten) verknüpft und so zu einer HTML-Seite zusammengestellt. Im Zusammenhang mit HTML-Formularen werden ebenso Daten manipuliert: das Abschicken eines Formulares vom Browser (gleich ob mit der HTTP-Methode `post` oder `get`) sorgt dafür, daß die Inhalte der Formularfelder (ein Datenpaket) an den Server geschickt werden. Der Server muß nun das Datenpaket in Empfang nehmen und auswerten. Gegebenenfalls speichert er Teile der Formularfelder (Daten) in einer Datenbank und generiert eine HTML-Seite als Antwort (wieder ein Datenpaket).

[8] Eine ausführliche Dokumentation der Schnittstellen der Komponenten bzw. der Objekte ist in der API-Dokumentation auf der beiliegenden CD vorhanden.

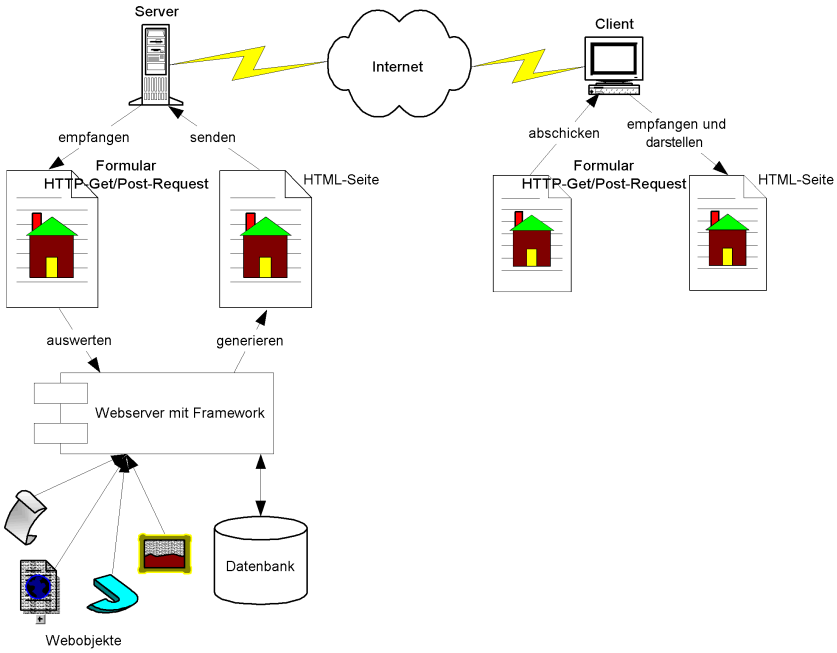

Abbildung 2.2: Datenzentrierte Weltsicht des Frameworks

Das Framework ist als Sammlung von lose datenstrukturgekoppelten Objekten konzipiert. Die Datenstrukturen, die auf oberster Ebene zwischen den Objekten ausgetauscht werden, sind so einfach und universell wie möglich gestaltet. Als Beispiel sei hier das Zusammenspiel zwischen Datenbanklayer und der Ausgabe in Tabellen- oder Listenform genannt (Abschnitt 2.3.6 bzw. 2.3.9). Aus der Datenbank wird mit Hilfe einer Datenbeschreibung ein Datensatz oder eine Gruppe von Datensätzen ausgelesen. Diese Daten werden in einem zweidimensionalem Array gespeichert: pro Reihe ein Datensatz, pro Spalte ein Feld des Datensatzes. Das Array wird nun einem Printerobjekt zur Ausgabe übergeben. Hierbei ist es egal, ob die Ausgabe durch ein Tabellenausgabeobjekt oder ein Listenausgabeobjekt erfolgen soll, da die jeweiligen Signaturen der aufzurufenden Methoden der Objekte identisch sind und mit der Datenstruktur des zweidimensionalen Arrays als Parameter arbeiten. Der real verwandte Printer ist somit für die Datengewinnung aus der Datenbank irrelevant. Auch ist umgekehrt die Datenquelle für den Printer irrelevant. Die Daten können ebensogut aus beispielsweise einem einzulesenden Textfile oder einem fest codierten Programmtext stammen. Die Kopplung von Ausgabe und Datenquelle erfolgt über die einfache Datenstruktur des zweidimensionalen Arrays.

2.3.2 Grundentscheidungen

Im Rahmen des Designs von Redaktionssystemen bzw. Content Management Systemen gibt es einige zum Teil orthogonale Grundentscheidungen zu fällen, dazu gehören die folgenden Fragestellungen:

- Welche Inhalte sollen verwaltet werden? Diese Entscheidung wurde durch das Einsatzfeld definiert und in der Anforderung 2.2.4 festgehalten.
- Wie und wo werden die Inhalte gespeichert? Dieser Punkt ist beeinflußt durch die Art der Inhalte und er beeinflußt wiederum die Wahl der Technologie. In diesem Framework wurde die Entscheidung auf Grund der Erfahrungen in den Vorprojekten und der zu verwaltenden Daten zu Gunsten relationaler Datenbanksysteme gefällt.
- Welche Technologie soll eingesetzt werden? (siehe Abschnitt 2.3.3)
- Was für ein Publikationsmechanismus soll eingesetzt werden? Caching oder dynamisch? (siehe Abschnitt 2.3.4)
- Wie wird der Inhalt vom Layout getrennt? (siehe Abschnitt 2.3.5)
- Wie wird die Logik einer Applikation vom Layout getrennt? Im Framework findet die Trennung durch Ausgabeobjekte, Datenbeschreibungen, Informationselemente und die lose datenzentrierte Kopplung der Objekte statt (siehe Abschnitte 2.3.9, 2.3.7, 2.3.8 und 2.3.1), damit können die Daten in den Applikationen weitgehend unabhängig von der Darstellung erzeugt und dann den Ausgabeobjekten übergeben werden.

2.3.3 Grundentscheidung: Eingesetzte Technologie

Es gibt eine ganze Reihe von möglichen Technologien für die Realisierung des Frameworks: angefangen bei serverseitigen Skriptspracheninterpretern wie Coldfusion, ASP und Java-Servlet-Containern bis hin zu Applikationsservern wie IBM WebSphere oder WebLogic von BEA. Auch existieren eine Reihe fertiger Frameworks wie CoreMedia, Vignette oder Gaus VIP[9]. Ebenso kommen eine Reihe von Datenbanken für die Datenhaltung in Betracht. Die Kosten für Lizenzen von Applikationsserver und Datenbank sind allerdings beachtlich und liegen zwischen EUR 5.000 und EUR 250.000 (und darüber hinaus) und überschreiten häufig das Gesamtbudget eines Webprojektes.

[9] Für eine Übersicht über Informationsquellen zu den hier genannten Produkten siehe auch Anhang E.

Um der Forderung nach einer preisgünstigen Lösung (Abschnitt 2.2.1) Rechnung zu tragen, wird für das Framework PHP[10] als Applikationssprache und MySQL als Datenbank genutzt.

PHP ist eine Open Source Software, für die keine Lizenzgebühr anfällt. Es gibt aber im Bedarfsfall auch kommerziellen und professionellen Support z.B. durch die Firma *Zend Technologies Ltd.*[11], oder auch durch eine Reihe von in Deutschland ansässigen Firmen und Entwicklern. PHP wurde durch einen Entwicklerzusammenschluß[12] weiterentwickelt und gewartet. Mittlerweile ist PHP ein Projekt der *Apache Software Foundation*, damit kann die Zukunft von PHP als gut gesichert angesehen werden.

MySQL ist ebenfalls eine Open Source Software, die von der Firma *MySQL AB*[13] weiterentwickelt und gewartet wird. Für den Einsatz mit dem Framework ist keine Lizenzgebühr zu entrichten. Als Datenbank wird MySQL zwar empfohlen, aber das Framework ist auch mit anderen Datenbanken einsetzbar, wie im Abschnitt 2.3.6 ausgeführt wird.

Die laufenden Kosten für den Betrieb einer auf der Plattform PHP/ MySQL basierten Site sind auf Grund der hohen Verbreitung der Plattform bei Internet Service Providern ebenfalls sehr günstig (ab einer Monatsmiete von ca. EUR 10 bis EUR 40). In diesem Entgelt sind dann auch die Kosten für den administrativen Teil — also für Sicherungen der Daten, Wartung und Pflege der Software sowie Sicherheitsaudits der Maschinen — enthalten.

2.3.4 Grundentscheidung: Caching vs. dynamisches Publishing

Publikationslösungen für den allgemeinen Zugriff via Internet müssen (zumindest im geplanten Einsatzbereich des Frameworks) durchaus mit einer größeren Anzahl von zeitgleichen Zugriffen zurechtkommen, ohne in Performanceprobleme zu geraten (siehe Forderung des Abschnitts 2.2.1). Hierzu muß entweder der Publikations- bzw. Produktionsmechanismus einer HTML-Seite schnell genug sein oder ein Cachingmechanismus[14] eingesetzt werden.

Bei Cachinglösungen (siehe Abbildung 2.3) geschieht im Prinzip folgendes: die auszuliefernden Seiten werden von einem Publikationsserver zunächst vorproduziert (bzw. bei der ersten Anforderung der Seite generiert) und als

[10] Die Version 1.0 wurde am 8.6.1995 von Rasmus Lerdorf als *Personal Home Page Tools* in der Newsgroup news://comp.infosystems.www.authoring.cgi angekündigt. Mittlerweile liegt die Version 4.1.1 vor.

[11] unter `http://www.zend.com` [wwwProduct:Zend] zu erreichen.

[12] unter `http://www.php.net` [wwwProduct:PHP] zu erreichen

[13] unter `http://www.mysql.com` [wwwProduct:MySQL] zu erreichen

[14] Hiermit ist *nicht* ein herkömmlicher Proxy gemeint, wie er bei vielen Internet Service Providern zu finden ist. Derartige Proxies sollen nur die Transfermenge verringern, um für die ISP geringere Leitungskosten zu erreichen. Bei heutigen dynamischen Websites werden durch solche Proxy typisch nur noch Grafikdateien effektiv zwischengespeichert.

fertige HTML-Seiten für die (erneute) Auslieferung an den Client zwischengespeichert. Den Vorgang des Zwischenspeicherns für die spätere Nutzung nennt man *Caching*.

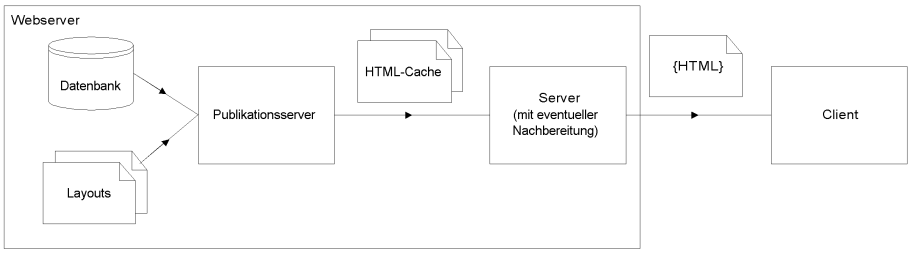

Abbildung 2.3: Publizieren mit Caching

Wird auf Caching verzichtet und die Seite erst bei einer Anfrage stets neu produziert, dann spricht man vom On-Demand-Publishing oder auch von dynamischem Publizieren (siehe Bild 2.4).

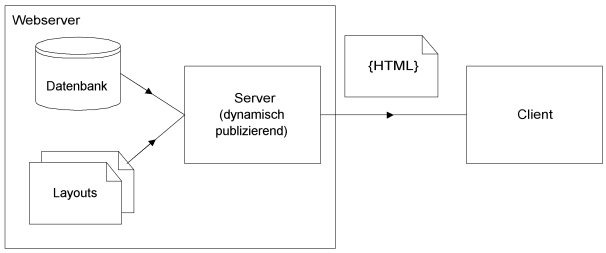

Abbildung 2.4: Dynamisches Publizieren

Es gibt im Markt auch Mischsysteme mit aufwendigen Algorithmen, die einige Teile der Seiten cachen, andere Teile On-Demand produzieren und dann die zusammengefügten Seiten ausliefern.

Beide Varianten haben Vor- und Nachteile. In der Tabelle 2.1 werden Caching und On-Demand-Publishing gegenübergestellt und bewertet.

Tabelle 2.1: Vergleich, Caching vs. On-Demand-Publishing

	Caching		**On-Demand-Publishing**
+	kleinere Latenzzeit in der Auslieferung	−	eventuell größere Latenzzeit in der Auslieferung
+	Blätterfunktionen einfacher möglich	−	Blätterfunktionen erfordern komplexere Datenstrukturen
−	verzögerte Publikation	+	sofortige Publikation
−	schwierige Verwaltung der Gültigkeitsdauer	+	die Daten sind immer gültig

wird fortgesetzt

Fortsetzung

	Caching		On-Demand-Publishing
−	Teilcaching erhöht die Komplexität des Seitenaufbaus (Programmierung)	+	einfache Seitenstruktur (Programmierung)
−	niedrige Interaktivität (besonders schwerwiegend in Bezug auf die Entwicklung von Applikationen)	+	Interaktivität fordert keinen Wechsel des Paradigmas
−	schlechte Personalisierbarkeit	+	gute Personalisierbarkeit
−	Session nur mit Cookies oder unsicher auf Basis von IP und Clientdaten (Proxyproblem) oder mit Nachbereitung der Seiten (also teilweisem dynamischen Publizieren) möglich	+	Session auch via GET und POST

Der Vergleich fällt hier deutlich zu Gunsten des On-Demand-Publishings aus. Vor allem unter der Berücksichtigung der Vision einer stark interaktiven und individualisierbaren Site werden die Algorithmen für eine Caching-Lösung sehr aufwendig und kompliziert, und die entstehenden Kosten sind nicht durch den Nutzen zu rechtfertigen. Das Framework wird somit nicht mit einer Cachinglösung ausgestattet.

Diese Entscheidung hat keinerlei Auswirkung auf das Zusammenspiel mit sogenannten Proxies bei Internet Service Providern (AOL, T-Online, ect.). Derartige Proxies stellen zwar auch eine Cachingsoftware dar, haben aber einen ganz anderen Einsatzweck. Sie speichern Webobjekte völlig transparent für den Nutzer zwischen, um den Internet Service Providern hohe Transferkosten für häufig zugegriffene Webobjekte zu ersparen.

2.3.5 Grundentscheidung: Template vs. Objekt

Häufig wird im Zusammenhang der Trennung von Darstellung und Inhalt im Webumfeld von templatebasierten Ansätzen gesprochen. Hierbei wird eine fast normale HTML-Seite benutzt, die angereichert mit Platzhaltern die Gestaltung darstellt. Der Inhalt wird dann zu einem Zeitpunkt durch einen Parserlauf an Stelle der Platzhalter in die Seite eingebracht und das Ergebnis als fertige Seite zum Browser oder in einen Zwischenspeicher (Cache, siehe Abschnitt 2.3.4) geliefert.

Eine andere Weise der Trennung von Darstellung und Inhalt ist die Programmierung innerhalb einer Seite mit Hilfe von Daten- und Darstellungsobjekten. Im Folgenden wird ein Vergleich der beiden Ansätze durchgeführt.

- Die Zielrichtung von Templates ist, daß sie auch von HTML-Designern bzw. Grafikern benutzt werden können, im besten Fall unter Einsatz eines beliebigen HTML-WYSIWYG-Editors (siehe Abschnitt 1.6.1). Im objektorientierten Ansatz können die Grafiker nur die Vorgaben liefern, die dann von Programmierern umgesetzt werden müssen.

- Negativ ist die erhöhte Belastung des Servers durch Templates, denn die Templates müssen ja — zusätzlich zur Abarbeitung einer Applikationslogik — zunächst geparst werden und dann die Platzhalter in den Templates mit den ermittelten Werten ersetzt werden.

- Templates sind nicht vererbbar. Die siteweit gleiche Darstellung z.B. von Tabellen muß in jedem einzelnen Template gepflegt werden. Dies hat bei der Änderung z.B. der Rahmenfarbe das manuelle Ändern von vielleicht 20 Templates zur Folge. Beim objektorientierten Ansatz kann das Vaterobjekt für die Tabellendarstellung angepaßt werden und damit sind alle Tabellen der Site angepasst. Alternativ wird bei Templates ein Includemechanismus für Subtemplates eingebaut, dies führt aber dazu, das keine beliebigen HTML-WYSIWYG-Editoren mehr eingesetzt werden können. So wird ein Vorteil der Templates verspielt. Desweiteren rückt die Templateerstellung stärker in die Nähe der Programmierung.

- Die Erweiterung von Datensätzen muß bei Templates in den einzelnen Templates vorgenommen werden. Beim programmierten Ansatz erfolgt die Änderungen in den Metadaten und setzt sich damit automatisch in den verschiedenen Darstellungen durch.

- Die Namensräume für die Platzhalter in den Templates erfordern eine gute Konvention, damit die Organisation übersichtlich bleibt. Beim objektorientierten Ansatz ist die Ordnung in Namensräume prinzipiell gegeben.

- Das Formularhandling erfordert entweder eine größere Anzahl von Templates, oder aber eine aufwendigere (programmierbare) Art von Templates (und damit rückt der Templateansatz in die Nähe des programmierten ojektorientierten Ansatzes).

- Beim objektorientierten Ansatz läuft man Gefahr, die Trennung der Applikationslogik von der Darstellung zu verwischen.

Für das Framework wurde unter Berücksichtigung obiger Punkte die Entscheidung zu Gunsten des objektorientierten Ansatzes gefällt. Dies auch

deshalb, weil das Framework primär von Programmieren benutzt werden wird und der gewählte Ansatz stärker deren Denkweise entspricht. Ein letzter ausschlaggebender Punkt waren durchgeführte Lasttests mit einer ersten Version des Frameworks, die gegen eine in der Open Source zur Verfügung stehenden Templatelösung[15] gefahren wurden. Hierbei wurde die Performance einer programmierten Lösung (in ähnlicher Struktur wie das realisierte Framework) als gleichwertig und ausreichend überprüft. Die Ergebnisse sind im Anhang A zusammengefasst. Dort wird auch das letztlich realisierte Framework auf seine Belastbarkeit getestet.

Prinzipiell steht trotz der Entscheidung für den objektorientierten Ansatz dem parallelen Nutzen einer zusätzlichen Templatelösung nichts im Weg. Die lose Koppelung der Frameworkkomponenten erlaubt es, statt einer Ausgabe über die Printerklassen (Abschnitt 2.3.9) auch eine Templatelösung für die Ausgabe zu erstellen und zu nutzen. Dieser Ansatz wird aber in der Arbeit nicht weiter verfolgt.

2.3.6 Datenbankzugriff

Erster Baustein des Frameworks ist das Design des Datenbankzugriffes. Alle Inhalte einer Site, die mit dem Framework aufgesetzt wird, werden in einer Datenbank gespeichert. Auf diese Weise wird die einfache Pflege der Inhalte über Formulareingaben ermöglicht. Auch laufende Informationen im Betrieb der Site wie die aktuellen Sessions werden in einer Datenbank gehalten. Der Datenbankzugriff ist somit die Grundlage für das gesamte Framework.

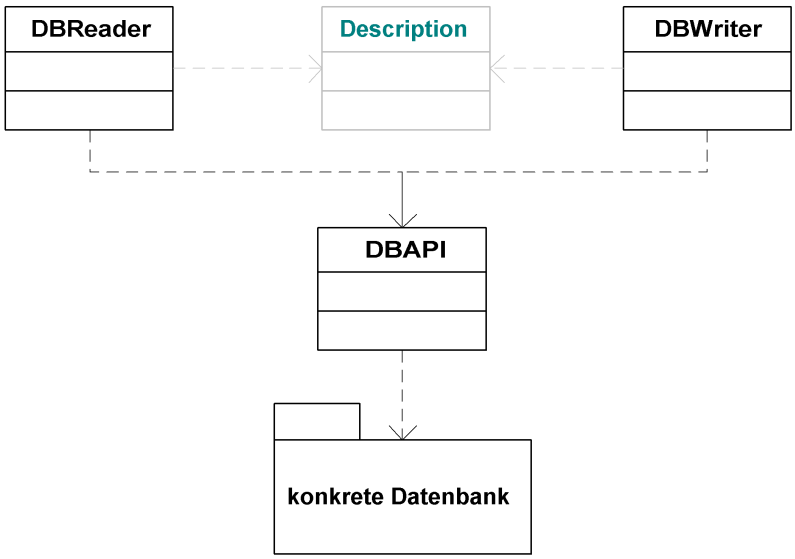

Abbildung 2.5: Klassendiagramm zum Datenbankzugriff

[15] *PHPLib*; unter `http://phplib.sourceforge.net` zu finden [wwwProduct:PHPLib].

Der Datenbankzugriff wird über drei Klassen realisiert. Die erste Klasse `WCK_DBAPI` sorgt für eine Abstraktion von der konkret eingesetzten Datenbank (Anforderung 2.2.8), `WCK_DBReader` liest die Datenbank mittels Datenbeschreibungsobjekten (siehe Abschnitt 2.3.7) aus und `WCK_DBWriter` beschreibt oder ändert die Datenbank mittels Datenbeschreibungsobjekten.

Referenzimplementierungen der Klasse `WCK_DBAPI` sind auf der Basis von PostgreSQL und MySQL durchgeführt worden. Die Implementierung für Oracle war aufgrund von Schwierigkeiten in der Beschaffung einer laufenden Oracle-Entwicklungsumgebung leider nicht möglich. Da alle Datenbankspezifika in der Klasse `WCK_DBAPI` gekapselt werden, sollte aber eine Implementierung für Oracle ebenfalls keine Schwierigkeiten bereiten.

WCK_DBAPI

SQL ist nicht gleich SQL. Diese Aussage ist in Bezug auf die konkreten Implementierungen von DBMS leider zutreffend. Obwohl sich die DBMS fast alle am SQL92-Standard orientieren, implementieren sie diesen oft nicht vollständig oder sie fügen dem Standard nützliche, aber proprietäre Erweiterungen hinzu. Da das Framework jedoch so unabhängig wie möglich vom eingesetzten DBMS sein soll, müssen für besondere Fähigkeiten des — bei bisherigen Projekten eingesetzten — MySQL generalisierte Methoden benutzt werden, die vom konkret eingesetzten DBMS abstrahieren. Hier sind im speziellen die MySQL-proprietären Fähigkeiten zur

- **automatischen Generierung von Primärschlüsseln** mit Hilfe der Funktionalität des Feldattributes `auto_increment` und
- **die Beschränkung der Anzahl der gelieferten Ergebnisreihen** mit Hilfe der Ergänzung des `select`-Statements durch den Zusatz `limit n,m`

zu nennen.

Im Framework werden diese Funktionalitäten durch das Objekt zum Datenbankzugriff `WCK_DBAPI` isoliert und somit austauschbar gemacht.

`WCK_DBAPI` ist für die direkte Kommunikation mit dem eingesetzten DBMS verantwortlich. Hierzu gehören folgende Aufgaben (in Klammern stehen die Methoden, die die Aufgabe erfüllen[16]):

- Herstellung einer Verbindung zum DBMS und Anmeldung des entsprechenden Datenbanknutzers. (`connect`)
- Abbau der Verbindung zum DBMS. (`close`)
- Ausführen einer SQL-Abfrage. (`query` und `querySelect`)

[16] Hier und im Folgenden gilt, daß eine ausführliche Dokumentation der Klassen in der API-Dokumentation und im Sourcecode auf der beigelegten CD zu finden ist.

- Lieferung des Ergebnisses als zweidimensionales Array (siehe unten). (`fetch`)
- Lieferung der Anzahl der Ergebnisreihen der letzten Abfrage. (`numRows`)
- Funktionen zur automatischen Primärschlüsselgenerierung. (`lastInsertID`, `sqlInsertIDName` und `sqlInsertIDValue`)
- Abfrage des Fehlerzustandes der Datenbank bzw. ausgeführter SQL-Anfragen. (`error` und `errorMsg`)

Die Daten aus der Datenbank werden als zweidimensionales Array geliefert. Das Format dieses Arrays ist schlicht: zeilenweise werden die Datensätze gespeichert und in den Spalten finden sich die jeweiligen Felder der Datensätze wieder. Der Grund für dieses simple Datenformat findet sich im Design der Ausgabe wieder (siehe Abschnitt 2.3.9). Je einfacher und unspezifischer dieses Datenformat ist, desto stärker werden die davon abhängigen Klassen entkoppelt und damit flexibler einsetzbar.

WCK_DBReader

Dieses Objekt stellt den lesenden Zugriff mittels Datenbeschreibungen auf die Datenbank zur Verfügung. Es hat folgende Verantwortlichkeiten[17]:

- Auswertung der globalen Variablen zur Steuerung der Blätter- und Sortierfunktionen bei Listen und Tabellen. (`$wck_DBrestrict`, `$wck_DBskip`, `$wck_DBorder`, `$wck_DBrows`, die in `$constraints[]` zusammengefasst werden). (`fetchGlobals`)
- Abfrage der Datenbank unter Berücksichtigung der Datenbeschreibung und der `$constraints[]`. Lieferung der Daten in einem zweidimensionalen Array. Hierbei werden die Daten der Datenbank gegebenenfalls durch die Datenbeschreibung zu direkt darstellbaren Daten transformiert. (`getArray` und `getValueArray`)

WCK_DBWriter

Dieses Objekt stellt den schreibenden Zugriff mittels Datenbeschreibungen auf die Datenbank zur Verfügung. Es hat folgende Verantwortlichkeiten[18]:

- Abspeichern der Daten einer Datenbeschreibung. Entweder als neuen Datensatz oder als Update eines existierenden Datensatzes. (`saveData`)
- Liefern des letzten generierten Primärschlüssels, wenn ein neuer Datensatz gespeichert wurde. (`lastInsertID`)

[17] In Klammern stehen die Methoden, die die Aufgabe erfüllen. Eine ausführliche Dokumentation der Objekte und ihrer Methoden in der API-Dokumentation und im Sourcecode ist auf der beigelegten CD zu finden.

[18] s.o.

Benutzung

Der Nutzer des Frameworks greift normalerweise nur über Objekte von Typ `WCK_DBReader` oder `WCK_DBWriter` auf die Datenbank zu. Allerdings sollte auf jeder Seite eine globale Variable namens `$wck_DBapi` existieren, diese Variable ist ein fertig konnektiertes Objekt der Klasse `WCK_DBAPI`. Sie ist deshalb notwendig, weil viele Objekte des Frameworks (auch `WCK_DBReader` und `WCK_DBWriter`) auf eine Resource zur Datenbankanbindung zurückgreifen müssen und mit der Variablen diese Rescource auf einfache Weise bereitsteht.

Die Entscheidung, eine globale Variable vom Typ `WCK_DBAPI` anzulegen, ist aus Performancegründen gefallen. Es ist nicht sinnvoll, für jeden einzelnen Datenbankzugriff auf einer Seite jeweils eine eigene Verbindung zur Datenbank aufzubauen, zu nutzen und wieder abzubauen. Dies wäre eine viel zu performancelastige Vorgehensweise. Typischerweise wird die globale Variable `$wck_DBapi` in einer allgemeinen Konfigurationsdatei (z.B. `config.prt)` angelegt.

2.3.7 Datenbeschreibungen

Die Datenbeschreibungen sind einer der zentralen Bestandteile des Frameworks. In ihnen werden jeweils einzelne Informationeinheiten regulärer Struktur (siehe auch Abschnitt 1.2) beschrieben. Datenbeschreibungen stellen eine Zusammenfassung von Informationselementen (siehe Abschnitt 2.3.8) zu einer Informationseinheit bereit. Ein Beispiel: die Datenbeschreibung der Informationseinheit *Nachricht* faßt die Informationselemente *Schlagzeile*, *Intro*, *Text*, *Erstellungsdatum* und *Autor* zusammen.

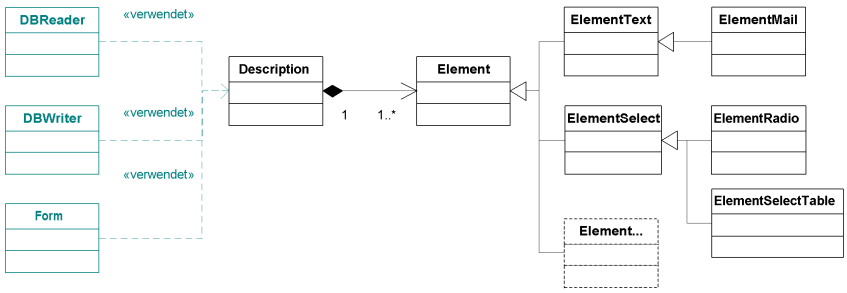

Abbildung 2.6: Klassendiagramm zur Datenbeschreibung

Über die Datenbeschreibungen werden im Framework einfache Applikationen schnell und in einer standardisierten Art aufgebaut (siehe hierzu die Realisierung von z.B. der Linklisten im Kapitel über den Einsatz des Frameworks, Kapitel 3). Mit Hilfe der Datenbeschreibungen und dem Datenbankleser `WCK_DBReader` können die verschiedenen Darstellungsformen wie Einzeldarstellungen (Abschnitt 1.3.2), Listen und Tabellen (Abschnitt

1.3.3) und Formulare (Abschnitt 1.3.4) mit Daten versorgt bzw. die Daten aus Formularen mit dem Datenbankschreiber `WCK_DBWriter` gespeichert werden.

Die Basisklasse für Datenbeschreibungen ist `WCK_Description`, diese Klasse hat folgende Verantwortlichkeiten[19].

- Zur Verfügung stellen der Informationselemente einer Informationseinheit. (Inhalt des Arrays `$fields[]`)
- Auflistung der Informationselemente, die dargestellt werden sollen. (Inhalt des Arrays `$order[]`)
- Bereitstellung der SQL-Statements dieser Informationseinheit für die Operationen *SELECT*, *INSERT* und *UPDATE*. (`createSQLSelect`, `createSQLInsert` und `createSQLUpdate`)
- Bereitstellung der Standardwerte der einzelnen Informationselemente. (`getDefaultValues`)
- Abfrage der aktuellen Daten, mit denen die einzelnen Informationselemente gefüllt sind. (`getDisplayAndValues`)
- Überprüfen der Konsistenz innerhalb der Gesamtheit der Informationselemente. (`verifyDependencies`)

Die Klasse `WCK_Description` ist eine abstrakte Basisklasse, denn in ihr sind noch keinerlei Informationselemente vorhanden. Erst durch Vererbung und Überladung des Konstruktors werden die beiden Arrays `$fields[]` und `$order[]` mit Inhalten — genauer Informationselementen, das sind Objekte des Typs `WCK_Element` — gefüllt (siehe dazu auch Abschnitt 2.3.14).

2.3.8 Informationselemente

Informationselemente — Objekte der Klasse `WCK_Element` oder einer abgeleiteten Klasse — tauchen nur innerhalb der Klasse `WCK_Description` auf (siehe auch Abbildung 2.6). Sie sind in diesem Sinne Bestandteil der Datenbeschreibungen. Die Informationselemente sind gewissermaßen die atomaren Einheiten von Informationen im Framework. Die Aufgaben eines Objektes des Klassenbaumes von `WCK_Element` sind folgende[20].

- Bereithalten eines Labels zur Benennung des Informationselementes. (`$display`)

[19] In Klammern stehen die Methoden, die die Aufgabe erfüllen. Eine ausführliche Dokumentation der Objekte und ihrer Methoden in der API-Dokumentation und im Sourcecode ist auf der beigelegten CD zu finden.

[20] s.o.

- Bereithalten von Informationen zur Speicherung des Informationselementes in der Datenbank. (`$name`)
- Bereithalten des Vorgabewertes des Informationselementes. (`$default`)
- Bereithalten des Speicherplatzes für einen aktuellen Wert des Informationselementes. (`$value`)
- Ausgabe eines Formularelementes zur Bearbeitung des Informationselementes. (`giveFormCharacter`)
- Ausgabe einer Kopfzeile mit eventueller Sortierfunktion für Tabellenspalten des Informationselementes. (`giveHeading`)
- Bereithalten von Metainformationen des Informationselementes. (`$sortable`, `$changeable`, `$viewable`, `$persistent`, `$mustNotEmpty`)
- Einlesen einer entsprechenden globalen Variable zur Verarbeitung von Rückgabewerten von Formularen. (`collectValue`)
- Validierung des aktuell gespeicherten Wertes. (`verify`)

Von der Klasse `WCK_Element` werden eine ganze Reihe von weiteren Klassen abgeleitet. Diese bilden jeweils ein besonderes Element zur Dateneingabe ab. Das Spektrum reicht hier von einfachen Textfeldern mit einer besonderen Validierungsroutine (z.B. `WCK_ElementMail` zur Eingabe von E-Mail-Adressen), über Textareas bis hin zu Checkboxen und Radiobuttons. Auch Selektionsfelder, deren Optionen aus einer Datenbanktabelle gelesen werden, gehören zu dem Klassenbaum von `WCK_Element`. Dieser Pool von Eingabemöglichkeiten wird mit der Entwicklung des Frameworks ständig erweitert und stellt letztlich einen Grundstock für eine schnelle Applikationsentwicklung dar.

2.3.9 Ausgaben

Nachdem nun mit dem Datenbanklayer und den Datenbeschreibungen Informationseinheiten gelesen werden können, müssen diese auch bis zum Nutzer weitergereicht und als HTML-Text ausgegeben werden. Diese Aufgabe wird von den Printern (zunächst `WCK_TablePrinter` und `WCK_ListPrinter`) übernommen. Um eine Trennung von Logik und Formatierung zu erreichen, nutzen die Printer Objekte von Formatierungsklassen als Hilfsobjekte (je nach Printer `WCK_TableFormat` oder `WCK_ListFormat` bzw. davon abgeleitete Klassen).

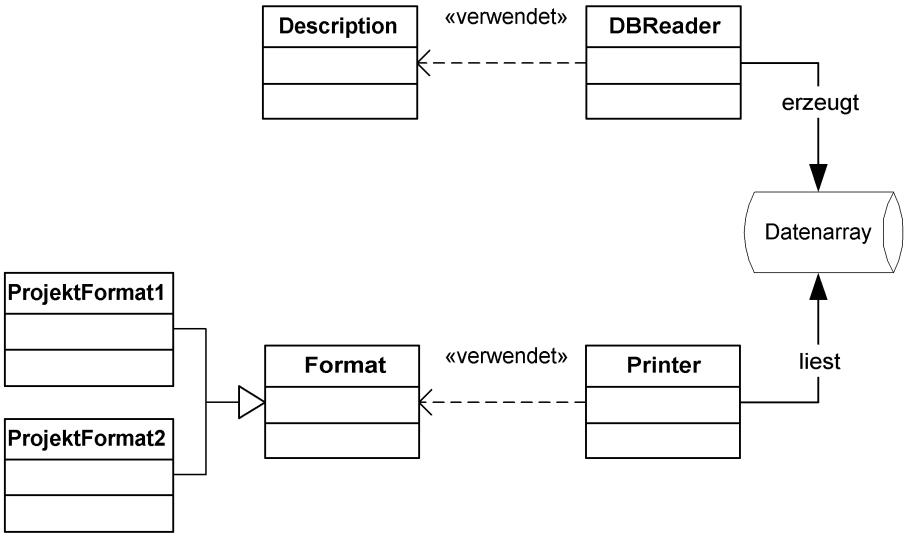

Abbildung 2.7: Klassendiagramm zur Ausgabe

Printer

Es stellt sich die Frage, wieviel Intelligenz die Printer besitzen sollen? Sollen sie etwas von den auszugebenden Daten verstehen, also Zugriff auf die dahinter liegenden Datenbeschreibungen bekommen? Generell wurde entschieden, die Printer in Bezug auf die Daten so „dumm“ wie möglich zu machen. Auf diese Weise werden die Printer so universell einsetzbar wie möglich. Die Tabellen- und Listenprinter arbeiten nur auf zweidimensionalen Arrays, in denen die auszugebenden Daten stecken[21]. Erst in Bezug auf Formatierungen der Eingabedaten werden die Printer intelligent. So kann zum Beispiel der Tabellenprinter zeilen- und/oder spaltenweise Einfärbungen vornehmen oder einzelne Zeilen bzw. Spalten hervorheben. Alle Möglichkeiten der Formatierungen darzulegen würde hier den Rahmen sprengen, dazu existieren im Framework Beispieldaten auf der beigelegten CD.

Die Verantwortlichkeit der Printer ist entsprechend dem verarbeiteten Datenformat einfach[22].

- Ausgabe der übergebenen Daten nach Maßgabe des ebenfalls übergebenen Formatierungsobjektes. (`output`)

Es gibt aber auch Spezialprinter wie den Formularprinter. Dieser Printer bekommt eine umfangreichere Datenstruktur, da Formulare wesentlich

[21] Diese Arrays können auch sehr leicht aus anderen Quellen als der Datenbank generiert werden. Auf diese Weise können auch z.B. festcodierte Einzelseiten von den Formatierungsfähigkeiten der Printer profitieren.

[22] In Klammern stehen die Methoden, die die Aufgabe erfüllen. Eine ausführliche Dokumentation der Objekte und ihrer Methoden in der API-Dokumentation und im Sourcecode ist auf der beigelegten CD zu finden.

komplexer aufgebaut sind als Tabellen oder Listen (siehe Abschnitt 2.3.10).

Formatierungen

Die Formatierungsklassen `WCK_TableFormat` und `WCK_ListFormat` sind letztlich nur externe Datensammlungen für die Printer. Der Grund für die Ausgliederung aus den Printern ist die bessere Trennung von Logik (den Printern) und Design (der Formatierung) und die getrennte Vererbbarkeit.

Klassenhierarchie der Printer

Die eigentlich logisch Klassenhierarchie für die Printer wurde aus Performancegründen aufgelöst. Die Printer stellen nun jeweils eigene Klassenbäume für Tabellen und Listen dar, denn die Gemeinsamkeiten sind so klein, daß eine gemeinsame Basisklasse nur bedingt nützlich wäre. Bei der Aufspaltung der Klassenbäume wurde aber das *Liskovsche Ersetzungsprinzip* für die öffentlichen Schnittstellen der Printer gewahrt, so daß Listen- und Tabellenprinter in einer Applikation leicht ausgetauscht werden können.

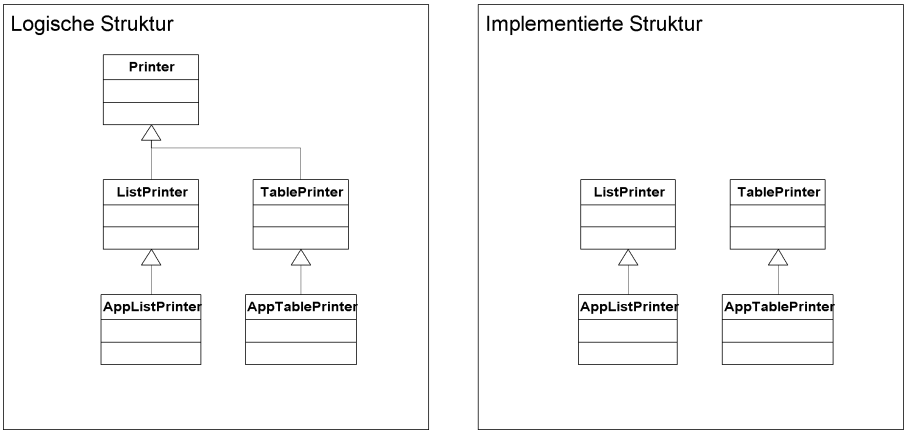

Abbildung 2.8: Printer Klassenbaum

2.3.10 Formulare

Das Formularhandling und die Formularausgabe haben eine etwas andere Aufteilung der Verantwortlichkeiten als die bisherigen Ausgaben. Hier liegt für die Gesamtausgabe eines Formulars die Verantwortlichkeit bei der Klasse `WCK_Form`. Der FormPrinter in der Klasse `WCK_FormPrinter` gibt nur einzelne Formularelemente und deren Belabelung bzw. zugehörige Fehlermeldungen aus.

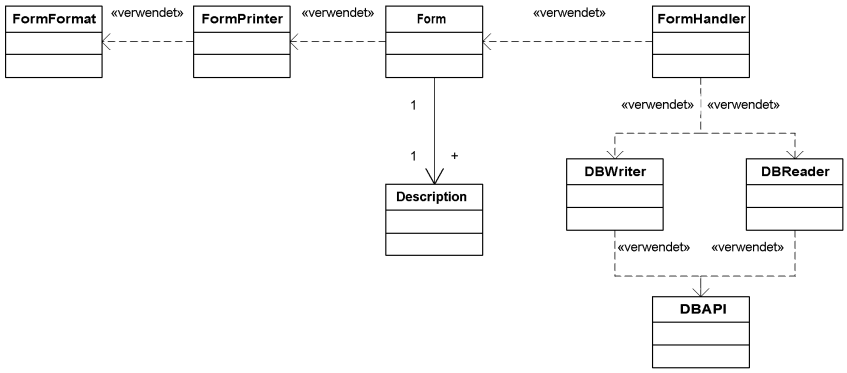

Abbildung 2.9: Klassendiagramm zum Formularhandling

In der Abbildung 2.9 sind die beteiligten Klassen und deren Verwendungsbeziehungen für das Formularhandling aufgezeigt. Die einzelnen Klassen haben dabei folgende Verantwortlichkeiten[23].

FormHandling `WCK_FormHandling` (diese Klasse stellt eine Formularverarbeitung nach dem Ablaufplan aus Abbildung 2.1 zur Verfügung)

- Abwicklung der gesamten Formularverarbeitung. (`doit`)
- Kommunikation mit der Datenbank
  - Speichern von neuen Datensätzen.
  - Auslesen von zu editierenden Daten aus der Datenbank.
  - Aktualisieren von editierten Daten in der Datenbank.

  zur Erledigung dieser Aufgabe greift die Klasse über `WCK_Form` auf die Datenbeschreibung zurück.
- Wiedervorlage bei fehlerhaften Eingaben.
- Ausgabe von Bestätigungen bei erfolgreicher Bearbeitung.

Form `WCK_Form`

- Ausgabe des gesamten Formulars (mit `<form>`-Tag). (`output`)
- Validierung von in der Description gespeicherten Daten und deren Abhängigkeiten. (`verify`)
- Einlesen von empfangenen Formulardaten. (`collectValues`)

[23] In Klammern stehen die Methoden, die die Aufgabe erfüllen. Eine ausführliche Dokumentation der Objekte und ihrer Methoden in der API-Dokumentation und im Sourcecode ist auf der beigelegten CD zu finden.

- Liefern von SQL-Statements für *INSERT* und *UPDATE*. (`createSQLInsert` und `createSQLUpdate`)

FormPrinter `WCK_FormPrinter`

- Ausgabe des sichtbaren Formularanfanges und -endes. (`start` und `stop`)
- Ausgabe der einzelnen Formularelemente. (`elementCharacterX`)

FormFormat `WCK_FormFormat`

- Bereitstellung einiger Formatierungselemente wie den sichtbaren Formularanfang und -ende
- Bereitstellung der Formatierungen für die Elemente

2.3.11 Nutzer-, Rechte- und Sessionverwaltung

Da die Nutzer-, Rechte- und Sessionverwaltungen eng miteinander verknüpft sind, werden alle drei Aspekte des Frameworks an dieser Stelle gemeinsam besprochen. Im Folgenden wird daher statt Nutzer-, Rechte- und Sessionverwaltung nur Userhandling gesagt, wenn alle drei Aspekte gemeinsam gemeint sind.

Dreh- und Angelpunkt des Userhandlings ist die Klasse `User`. Diese Klasse hat folgende Verantwortlichkeiten[24].

- Identifikation einer Folge von Seitenabrufen und damit eines Nutzers. Sessionidentifikation. (Konstruktor `User`, `giveGet`, `giveForm`, `quit`)
- Authentifizierung eines Nutzers; z.Z. durch Überprüfung von angegebenen Namens- und Passwortkombinationen. (`checkUser`)
- Erteilen von Auskünften über die Rechte eines Nutzers. (`permission`)
- Verwaltung von Sessionvariablen. (`close`, `setVar`, `getVar`, `existsVar`, `deleteVar`)

Voraussetzungen für das Userobjekt

Um eine Skalierbarkeit nach Abschnitt 4.4 im Einsatz des Frameworks zu ermöglichen, arbeitet die Klasse `WCK_User` nicht mit dem allgemeinen Datenbankschnittstellenobjekt `$wck_DBapi`, sondern mit `$wck_userDBapi` (ebenfalls ein Objekt der Klasse `WCK_DBAPI`). Über `$wck_userDBapi` wird auf die Nutzerdaten und -rechte zugegriffen und die Sessiondaten geschrieben

[24] In Klammern stehen die Methoden, die die Aufgabe erfüllen. Eine ausführliche Dokumentation der Objekte und ihrer Methoden in der API-Dokumentation und im Sourcecode ist auf der beigelegten CD zu finden.

und gelesen. Hierzu müssen in der angesprochenen Datenbank die folgenden Tabellen ad minimum definiert sein. Die Namen der Tabellen sind als Variablen in der Klasse `WCK_User` definiert und können bei Bedarf überschrieben werden.

Für die Nutzerdaten:

```
CREATE TABLE wck_user (
  // die ID des Nutzers; Primaerschluessel der Tabelle
  id bigint(20) unsigned NOT NULL auto_increment,
  // der Nutzername
  scrnname varchar(15) NOT NULL default '',
  // das Passwort
  password varchar(20) NOT NULL default '',
  // der Status des Nutzers; aktiv, geloescht, gesperrt
  status tinyint(4) NOT NULL default '0',
  PRIMARY KEY  (id),
  UNIQUE KEY scrnname (scrnname)
)
```

Für die Nutzerrechte:

```
CREATE TABLE wck_userperm (
  // ID des Datensatzes
  id bigint(20) unsigned NOT NULL auto_increment,
  // Nutzername dem das Recht gewaehrt wird
  scrnname varchar(15) NOT NULL default '',
  // Aufgabe fuer die das Recht gilt
  task varchar(10) NOT NULL default '',
  // Bereich in dem das Recht gilt
  area varchar(10) NOT NULL default '',
  // Sonderdaten zum Beispiel ein Forumsthema
  special varchar(15) NOT NULL default '',
  // Status des Rechtes; aktiv, geloescht
  status tinyint(4) NOT NULL default '0',
  PRIMARY KEY  (id),
  UNIQUE KEY user (scrnname,area,task)
)
```

Für die Sessiondaten:

```
CREATE TABLE wck_session (
  // ID des Datensatzes
  sid_id bigint(20) unsigned NOT NULL auto_increment,
  // Schluessel fuer den Datensatz
  sid_key varchar(32) NOT NULL default '',
```

```
  // IP-Adresse des Nutzers dieser Session
  sid_client_ip varchar(15) NOT NULL default '',
  // ID des mit der Session assozierten Nutzers
  sid_userid bigint(11) NOT NULL default '0',
  // Speicher der Sessionvariablen
  sid_sessionvar text NOT NULL,
  // letzter Zugriff auf die Session
  sid_last datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (sid_id),
  KEY last (sid_last)
)
```

Für die Rechteverwaltung (siehe weiter unten) kommen noch weitere Tabellen hinzu, aus denen die Daten für die Tabelle `wck_userperm` generiert werden.

Für die Bereiche:

```
CREATE TABLE wck_area (
  // ID des Datensatzes
  id bigint(20) unsigned NOT NULL auto_increment,
  // Name des Bereiches
  area varchar(10) NOT NULL default '',
  PRIMARY KEY  (id)
)
```

Für die Aufgaben:

```
CREATE TABLE wck_task (
  // ID des Datensatzes
  id bigint(20) unsigned NOT NULL auto_increment,
  // Name der Aufgabe
  task varchar(10) NOT NULL default '',
  PRIMARY KEY  (id)
)
```

Für die möglichen Kombinationen von Bereich und Aufgabe:

```
CREATE TABLE wck_areatask (
  // ID des Datensatzes
  id bigint(20) unsigned NOT NULL auto_increment,
  // Name des Bereiches und der Aufgabe
  area varchar(10) NOT NULL default '',
  task varchar(10) NOT NULL default '',
  special varchar(20) NOT NULL default '',
  PRIMARY KEY  (id)
)
```

Lebenszyklus eines Userobjektes

Das Userobjekt kann nach seiner Initiierung und Initialisierung mit der SessionID mehrere Zustände annehmen. Die „stabilen“ Zustände sind im folgenden Diagramm hervorgehoben. Die ebenfalls möglichen Übergänge zwischen den Zuständen *User ist Anonym* und *Sessionvariable gespeichert* via Aufruf von `close()` und zwischen dem Zustand *Sessionvariable gespeichert* und jeweils *User ist Anonym* und *User ist bekannt* via Aufruf von `setVar()` wurden zwecks Übersichtlichkeit weggelassen.

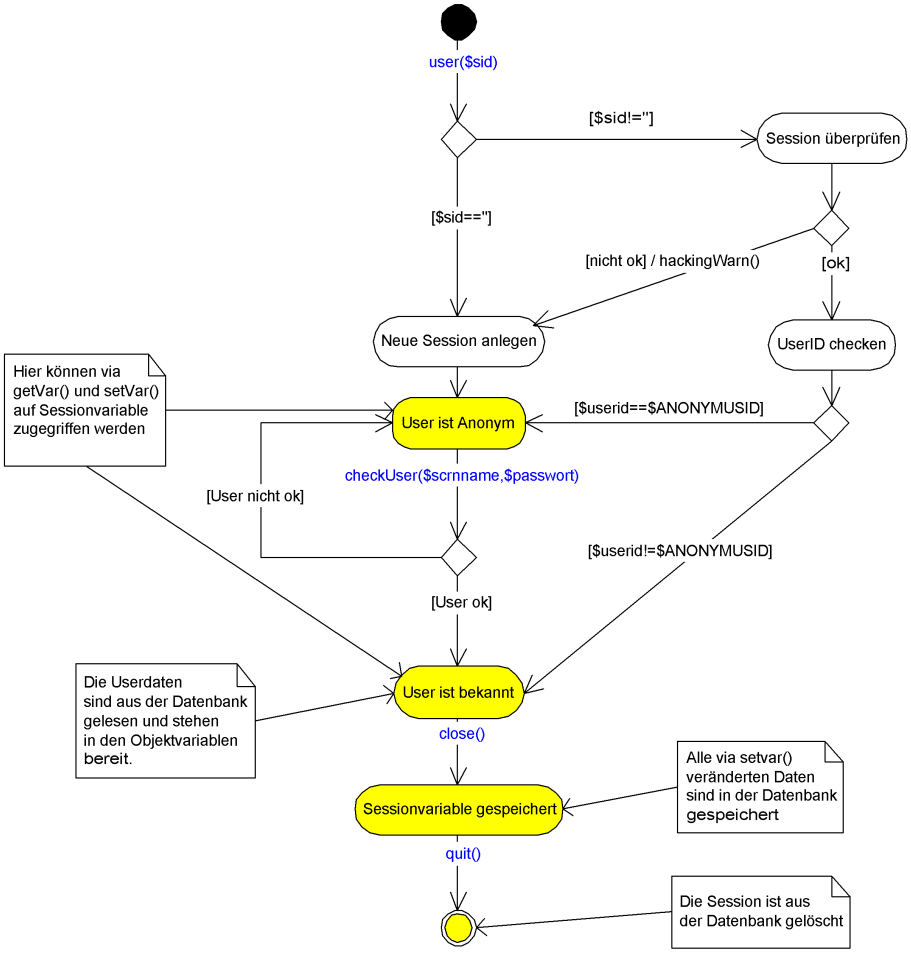

Abbildung 2.10: Lebenszyklus des Userobjektes

Sessionverwaltung

Um diesen Lebenszyklus zu ermöglichen, ist es notwendig, eine Sessionverwaltung einzuführen.

Einzusetzende Technik Da HTTP (zumindest in der Version 1.0) ein verbindungsloses Protokoll ist, müssen Sessions auf einer anderen Ebene als dem HTTP behandelt werden.

Es stehen hierzu verschiedene Ansätze zur Identifikation eines speziellen Clients (einer Session) zur Verfügung.

- **Identifikation über Eigenarten des Browsers (IP, Client)**
  Diese Identifikation ist unsicher, da durch Proxies im Übertragungsweg die Informationen verändert werden können und so keine eindeutige Zuordnung zu einem Client möglich ist[25].

- **Cookies**
  Hiermit stünde eine sichere Methode zur Identifikation von Clients (allerdings nicht von Personen) zur Verfügung. Alle modernen Clients unterstützen Cookies, allerdings ist der Einsatz von Cookies mittlerweile aus Datenschutzgründen in der Regel unerwünscht.

- `mod_rewrite`
  Mittels des Moduls `mod_rewrite` des Apache-Webservers ist es möglich, eine Session-ID in die URL einer Webseite einzubauen, innerhalb des Servers diese wieder zu extrahieren und als Parameter bereitzustellen. Dies ist aber eine proprietäre Methode des Apacheservers.

- **`GET`- und `POST`-Variable**
  Dies ist neben dem aufwendigeren `mod_rewrite` der einzige sichere und allgemein gangbare Weg, eine Session-ID über aufeinanderfolgende Seitenabrufe zu transportieren.

Sicherheit der Session Wenn man Sessions dazu einsetzt, einen Nutzer einmal zu authentifizieren und dann als diesen Nutzer wiederzuerkennen, dann ist die Sicherheit der Session eine gewichtige Forderung. Sicherheit bedeutet hier, daß niemand eine solche Session entführen kann und sich so als jemand ausgibt, der er nicht ist und beispielsweise Daten ausspioniert oder manipuliert. Es gibt unterschiedliche Angriffsmöglichkeiten auf eine Session-ID, deren Abwehr im Rahmen des Frameworks im Folgenden diskutiert werden.

Erraten einer Session-ID Als Roh-Session-ID kann jede Zufallszahl genutzt werden. Mit Zufallszahlen entstehen Session-ID's variabler Länge und damit kann eventuell eine aktive Session mit einer kurzen Session-ID

[25] Siehe hierzu auch den Working Draft zur *Session Identification URI* unter `http://www.w3.org/TR/WD-session-id.html` `[wwwCite:W3C Session]`

entstehen, die erraten und so entführt werden kann. Um dies zu vermeiden, wird die erzeugte Session-ID mittels MD5-Algorithmus[26] auf eine feste Länge gebracht (128 Bit). Die ID's abgelaufener Sessions werden aus der Datenbank gelöscht. Die Chance, eine aktive Session-ID zu erraten, wird damit selbst bei vielen gleichzeitig aktiven Nutzern (bzw. aktiver in der Datenbank vorhandener Sessions) verschwindend gering.

$\frac{n_{Session}}{2^{128}}$, bei 10000 aktiven Sessions also: $\frac{10000}{2^{128}} \approx 2.94 * 10^{-35}$

Alte Seiten im Browser-Cache Da die Session-ID beim korrekten Ausloggen aus der Site aus der Datenbank gelöscht werden, ist auch eine Wiederaufnahme einer Session durch Benutzung des Back-Buttons der Browser nicht möglich. Zusätzlich wird eine Überprüfung auf Aktivität einer Session durchgeführt und Sessions, die eine einstellbare Zeit lang nicht aktiv waren, automatisch gelöscht.

Ausspionieren der Session-ID durch Mithörer Gegen diese Form der Spionage ist das System nicht gesichert, da die Session-ID normaler Bestandteil der via HTTP übertragenen Daten ist. Allerdings läßt sich die Verbindung vom Client zum Server mittels der Standard-Lösungen SSL, TLS oder SHTTP sichern, dies bereitzustellen ist jedoch Aufgabe des HTTP-Servers (beim Apache-Webserver z.B. mit dem Modul `mod_ssl`) und unabhängig vom hier betrachteten Framework.

Als zusätzlicher Schutz wird im Framework die IP-Adresse des Clients mitprotokolliert, der die Session initiiert hat. Bei jedem Zugriff wird überprüft, ob immer noch die gleiche IP-Adresse benutzt wird. Sollte sich die Adresse geändert haben, so wird die Session als ungültig angesehen und neu initiiert. Damit muß ein potenzieller Angreifer also gleichzeitig noch ein IP-Spoofing[27] vornehmen, was durch die Routingeinträge in den Servern des Internets dazu führt, daß der Angreifer meist gar keine Anwort erhält und so der Angriff ins Leere läuft.

Session-ID generieren In der Implementierung der Session-ID wird die Session-ID aus einer Zufallszahl via MD5-Algorithmus auf eine feste Länge gebracht. Da MD5 jedoch ein Hash-Algorithmus ist und somit keine eineindeutige Abbildung $\aleph \rightarrow \aleph$, sondern nur eine Abbildung $\aleph \rightarrow \mathbf{M}$, die zwar surjektiv, aber mit $\mathbf{M} \subset \aleph$ logischerweise nicht injektiv ist, können durch den Algorithmus prinzipiell zwei gleiche Session-ID's entstehen. Dies

[26] Die im Sommer 1996 aufgedeckten Sicherheitslücken [Grafinkel, Spafford 1997] des MD5-Algorithmus sind an dieser Stelle nicht relevant, da es nicht darum geht, eine Nachricht auf Authentizität zu prüfen, sondern nur darum, das Datum mittels eines Hashes auf eine feste Länge zu expandieren und zu randomisieren.

[27] Das ist das Vorspiegeln einer gefälschten IP-Adresse.

darf nicht der Fall sein, da sonst die Sessions nicht eindeutig identifiziert werden können. Um die Session-ID's eindeutig zu machen, wird dem generierten MD5-Hash noch der eindeutige Primärschlüssel des entsprechenden Datenbankeintrages angehängt. Das folgende Code-Fragment aus dem Konstruktor der Klasse `WCK_User` zeigt die Generierung der Session-ID.

```
// SessionId generieren
// zuerst den Session-Key erzeugen
$this->sidKey= md5(uniqid(rand()));
// und eintragen
$query="insert into $this->ac_session".
 "(sid_key, sid_client_ip, sid_userid)".
 "values ('$this->sidKey','$this->sidClientIP','$this->id')";
$userDBapi->query($query);
$this->sidID=$userDBapi->lastInsertID();
// SessionString zusammensetzen
$this->sid=$this->sidKey.'-'.$this->sidID;
```

Rechteverwaltung

Die Rechteverwaltung des Frameworks beruht auf dem Prinzip, daß man typischerweise verschiedene Aufgaben (Tasks wie *editieren*, *löschen* und *anlegen*) in unterschiedlichen Bereichen (Areas wie *Nachrichten*, *Termine* und *Linklisten*) durchführen kann. Nicht alle Kombinationen von Aufgaben und Bereichen sind sinnvoll. Der Programmierer einer Site entscheidet über sinnvolle Kombinationen (AreaTask) und legt diese in der Datenbank ab. Das Identifizieren von Aufgaben, Bereichen und deren Kombinationen wird als Zugriffsverwaltung bezeichnet.

Ein Beispiel: in einer Site gibt es einen Nachrichtenbereich. In der zur Site zugehörigen Administration gibt es einen Bereich (namens `news`), um die Nachrichten zu verwalten. Ebensfalls existiert dort ein Bereich (namens `sysinfo`), um auf Systeminfomationen wie die Logfiles zuzugreifen. Es gibt die folgenden Aufgaben: Nachrichten anschauen (`view`), neue Nachricht anlegen (`create`), Nachrichten editieren (`edit`), Nachrichten löschen (`delete`) und Systeminformationen anschauen (wieder `view`). Diese Aufgaben können bestimmten Gruppen (oder Rollen) zugewiesen werden. Alle auf Nachrichten bezogenen Aufgaben werden den Newsredakteuren zugesprochen und die Systeminformationen dem Systemverwalter. Als Aufgaben/Bereichskombinationen mit ihren zugehörigen Rollen gibt es also:

Area	Task	Gruppe/Rolle
news	view	Newsredakteur
news	create	„
news	edit	„
news	delete	„
sysinfo	view	Systemverwalter

Diese Rollen können nun einzelnen Nutzern der Site (bzw. des Administrationsbereiches) zugewiesen werden. Die Definition von Gruppen und die Zuweisung einzelner Aufgaben/Bereichskombinationen an eine Gruppe sowie das Zuweisen und Entziehen von Gruppenzugrhörigkeiten an Nutzer (bzw. von Rollen an Nutzer) wird Rechteverwaltung genannt.

Die Zugriffs- und Rechteverwaltung beruht in der Datenbank auf dem in Abbildung 2.11 gezeigten Entity-Relationship-Diagramm.

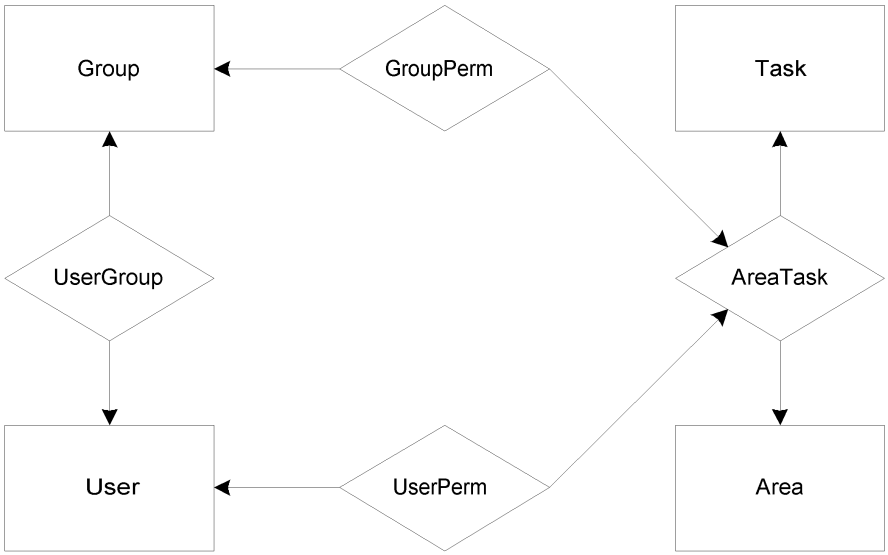

Abbildung 2.11: Vollständige Userverwaltung

Bei der Zuweisung eines Nutzers zu einer Gruppe geschieht folgendes: alle in der Tabelle `GroupPerm` gespeicherten Rechte der Gruppe werden für den Nutzer in der Tabelle `UserPerm` in Kopie angelegt. Wenn die Zugehörigkeit eines Nutzers zu einer Gruppe aufgelöst wird, dann müssen alle Rechte des Nutzers in `UserPerm` gelöscht werden und die verbleibenden Rechte der Gruppen, in denen der Nutzer noch Mitglied ist, wieder zugewiesen werden. Dies ist notwendig, weil die Gruppen nicht zwangsweise disjunkte Rechte haben müssen.

In der Standardrechteverwaltung des Frameworks können Nutzer, die die Berechtigung besitzen, Rechte zu vergeben, nur die Rechte vergeben, die sie selbst besitzen. So kann man die Verantwortung für einzelne Bereiche komplett — inklusive der Nutzer- und Rechteverwaltung — an mehrere

Personen weiterreichen und den Hauptverantwortlichen für die Site von der Administration entlasten.

Das Interface zur Rechteabfrage ist der Methodenaufruf `$wck_user->permission('area','task')` mit dem überprüft wird, ob der mit dem Objekt `$wck_user` der Klasse `WCK_User` identifizierte Nutzer das entsprechende Recht zur Ausführung der Aufgabe (`task`) im gewünschten Bereich (`area`) besitzt. Der Nutzer kann hier auch ein anonymer Nutzer sein, so können auch der Gesamtheit der anonymen Nutzer einfach Rechte zu- und aberkannt werden. Zum Beispiel kann so je nach Auslastung der Site der Zugriff für anonyme Nutzer zu bestimmten Bereichen verwehrt werden, um die angemeldeten Nutzer besser bedienen zu können.

Insgesamt erlaubt diese Art der Rechteverwaltung es, sehr fein abgestufte Rechte zu vergeben, ohne ein nicht mehr beherrschbares Chaos an Rechten vorliegen zu haben.

Sessionvariable

Letzter Teilaspekt der Userverwaltung sind die Sessionvariablen. Dies ist ein Mechanismus, um zusätzliche Informationen an eine Session zu hängen. Diese Informationen können zum Beispiel in einem Shoppingsystem ein virtueller Einkaufwagen sein. Eine andere Anwendung könnte in einem WikiWeb (siehe auch 4.7) der bei einem Besuch bisher zurückgelegte Weg durch das WikiWeb sein oder auch eine gewählte Darstellungsvariante. Sessionvariablen sind schlicht Informationen, die an den Besuch einer Site angehängt werden und so auch einem noch nicht authentifizierten Nutzer (also einem anonymen Nutzer) zugeschrieben und gespeichert werden können. Die Sessionvariablen können also auch für anonyme Nutzer eingesetzt werden.

Bei der Nutzung von Sessionvariablen ist darauf zu achten, daß zumindest immer dann, wenn die Sessionvariablen geändert wurden, die Methode `close` aufgerufen wird, da nur dadurch die geänderten Sessionvariablen auch in die Datenbank übernommen werden (dies wäre eigentlich eine typische Aufgabe für einen Destruktor der Klasse `WCK_User`, ein solches Sprachelement wird aber von PHP in der jetzigen Version leider nicht unterstützt[28]).

2.3.12 Zusammenspiel der Objekte

Abschließend wird nochmals eine Übersicht über die Klassen und ihr Zusammenspiel präsentiert. In der Abbildung 2.12 wird der Präfix `WCK_` vor den Klassennamen weggelassen.

[28] In der entsprechenden Mailingliste der Entwickler ist aber für die nächsten PHP-Versionen ein überarbeitetes Objektmodell mit Destruktoren angekündigt.

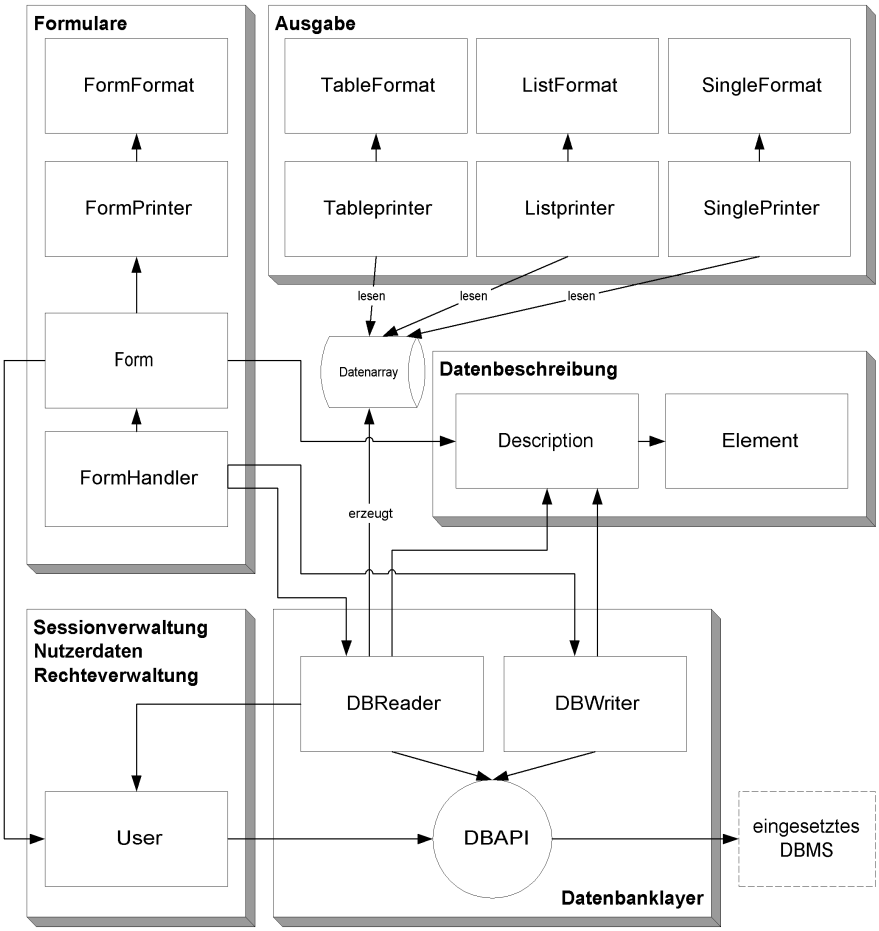

Abbildung 2.12: Zusammenspiel der Objekte

2.3.13 Umsetzung des Frameworks

„Pragmatics must take precedence over elegance, for Nature cannot be impressed.“
Coggins' Gesetz des Software-Engineerings

Bei der Umsetzung des Frameworks wurden einige Richtlinien zum Codieren und zur Namensvergabe in einem Style Guide festgeschrieben. Diese Richtlinien sind im Anhang B wiedergegeben.

An einigen Stellen wurden im Framework — zu Gunsten einer besseren Performance oder einer einfacheren Schnittstelle der Klassen — Kompromisse zu der reinen Lehre der Eleganz gemacht, hierzu zählt vor allem der Einsatz globaler Variablen. Die Natur der PHP-Programmierung, seitenorientiert zu sein, mildert diesen Stilbruch aber wieder ab, denn die Anzahl der in einem Interpreterlauf auftretenden globalen Variablen übersteigt durch die Partitionierung in abrufbare Seiten selten die Anzahl drei. Ebenso ist

die Zahl der Programmzeilen, die in einer Seite tatsächlich inkludiert werden, hinreichend gering, um noch den Überblick über das Geschehen zu behalten.

Im Folgenden werden einige Aspekte des Frameworks besprochen, die außerhalb des eigentlichen Kerns liegen.

2.3.14 Datenbeschreibungen

Zur Zeit werden die Informationselemente (Objekte aus dem Klassenbaum `WCK_Element`) der Datenbeschreibungen `WCK_Description` durch Vererbung und Überladung des Konstruktors in die Datenbeschreibungen hineingebracht.

Vererbung von Beschreibungen Von einer Datenbeschreibung gibt es im Einsatz typischerweise eine Reihe von Versionen. In einer Version sind zum Beispiel nur Teile der Informationen sichtbar (für eine Listendarstellung), während in einer anderen Version deutlich umfangreichere Darstellungen gefordert sind (Einzeldarstellung), oder eventuell zusätzliche Verwaltungsdaten angegeben werden (Formulardarstellungen im Administrationsbereich). Es stellt sich die Frage, wie diese Vererbung organisiert werden sollte, insbesondere gilt dies im Hinblick auf die Wiederverwendbarkeit von Applikationen in unterschiedlichen Sites (Projekten).

Es gibt drei unterschiedliche Varianten, die Beschreibungen zu vererben, die jeweils eigene Vor- und Nachteile mit sich bringen. Im Folgenden werden Vererbungsbäume vorgestellt, in denen für eine Applikation zwei Sichten (`Ansicht1` und `Ansicht2`) gefordert sind und die in zwei unterschiedlichen Projekten (`Projekt1` und `Projekt2`) eingesetzt werden soll.

Variante 1 Das Vorgehen der Abbildung 2.13 ist einfach, wenn keine Änderungen in der Elementanzahl, den Elementeigenschaften und der Elementanordnung zu erwarten sind. Gut ist hier, daß auf der Ebene der wiederverwendbaren Applikation alle Sichten vorhanden sind und damit eine funktionsfähige Basis zur Modifikation bereitsteht oder einfach übernommen werden kann. Daß projektspezifische Änderungen der vorhandenen Elemente oder Elementeigenschaften an vielen Stellen (in jeder Sicht) eingefügt werden müssen, ist hierbei ein Nachteil. Damit erfordert eine Individualisierung der Applikationen in den einzelnen Projekten einen hohen Aufwand.

Variante 2 Die Variante der Abbildung 2.14 hat nur den Vorteil, daß hier der Klassenbaum übersichtlicher wird, da auf die Sichten in der Applikationsebene verzichtet wurde. Allerdings ist damit noch keine Verbesserung in Bezug auf den Aufwand für individualisierte Applikationen in den Projekten erreicht. Der Nachteil, daß in der Applikationsebene nun Sichten fehlen,

kann durch eine Bereitstellung von Vorlagen für diese Sichten ausgeglichen werden.

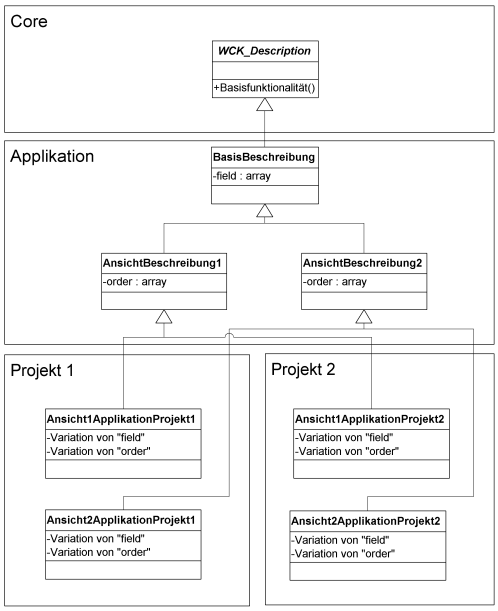

Abbildung 2.13: Datenbeschreibungsvererbung, Variante 1

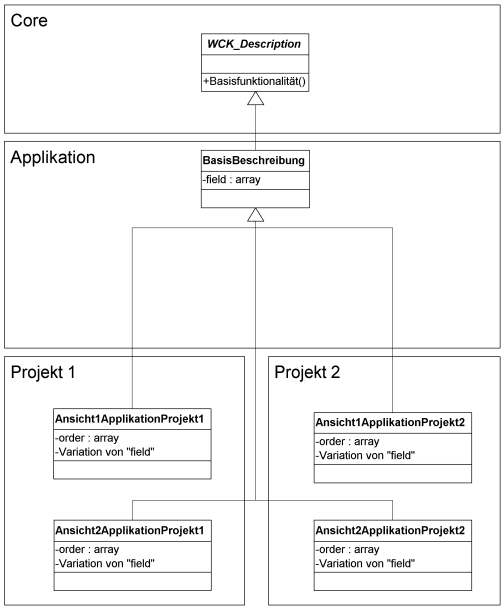

Abbildung 2.14: Datenbeschreibungsvererbung, Variante 2

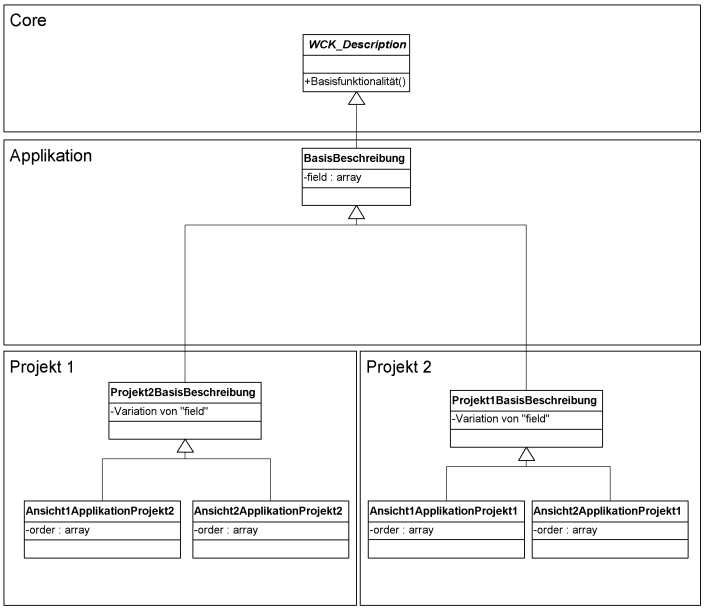

Abbildung 2.15: Datenbeschreibungsvererbung, Variante 3

Variante 3 Die letzte Variante (Abbildung 2.15) ist aus der Sicht der Individualisierung die flexibelste. Projektspezifische Änderungen der verfügbaren Elemente oder der Elementeigenschaften müssen nur an einer Stelle durchgeführt werden, in den jeweiligen `ProjektXBasisBeschreibung`. Damit bleiben die Applikationen in den Projekten robuster gegen spätere Anpassungen. Die in der Applikationsebene fehlenden Sichten können leicht als Vorlagen bereitgestellt werden.

Abschließend erscheint die Variante 3 insgesamt als am flexibelsten und robustesten. Sie ist damit die bevorzugte Variante für die Entwicklung von wiederverwendbaren Applikationen.

Alternative zur Vererbung Zur Zeit werden aus Gründen der Performance die Informationselemente (Objekte aus dem Klassenbaum der Klasse `WCK_Element`) der Datenbeschreibungen `WCK_Description` durch Vererbung und Überladung des Konstruktors in die Datenbeschreibungen hineingebracht. Hierzu muß also immer PHP-Quellcode geschrieben werden. Prinzipiell ist es aber auch möglich, eine Art DescriptionFactory zu programmieren, in der die Konfiguration der Informationselemente zum Beispiel aus einer Datenbank gelesen wird. Damit wäre es dann möglich, einfache Applikationen von Nichtprogrammierern über eine einfache Bedienoberfläche „zusammenklicken“ zu lassen und so die Applikationserstellung von den Programmierern auf die Redakteure zu verlegen. Allerdings dürfte ein solches Vorgehen deutliche Performanceeinbußen mit sich bringen. Die Erstellung

eines solchen Applikationsgenerators liegt außerhalb des Fokus dieser Arbeit.

Kapitel 3

Exemplarischer Einsatz des Frameworks

„Grau, teurer Freund, ist alle Theorie, und grün des Lebens goldner Baum.“
Goethe, Faust I

Da man ein Framework am besten im praktischen Einsatz kennenlernt, wird im Folgenden die Realisation einer exemplarischen Website mit Hilfe des in Kapitel 2 entworfenen Frameworks durchgeführt. Als Beispiel dient der Entwurf einer alternativen Lösung für die existierenden Seiten des Projektes zur *Integration Kognitiver Systeme* (IKS) der *Hochschule für Angewandte Wissenschaften Hamburg*[1].

3.1 Aufgabenstellung

Der Umfang der gewachsenen Struktur und die Anzahl der Seiten der Site des IKS-Projektes nimmt langsam eine Größe an, bei der es nicht mehr sinnvoll erscheint, die Site wie bisher mit den Mitteln der klassischen statischen Site, die als eine Sammlung einzelner HTML-Dateien gepflegt wird, zu betreiben. Die Veränderung der heutigen Weblandschaft legt auch eine Neuorientierung der inhaltlichen Ausrichtung der Site des IKS-Projektes nah. Nutzer erwarten immer stärker aktuelle Informationen. Ebenso steigt das Bedürfnis der Nutzer an lebendigen Inhalten und aktiver Beteiligung.

Aus diesen Gründen wird im Rahmen dieser Diplomarbeit — als exemplarische Einsatzmöglichkeit des entwickelten Frameworks — der Entwurf einer Site für das IKS-Projekt durchgeführt. Dieser Entwurf ist als eine Alternative zum bisherigen Bestand zu sehen, ersetzt ihn aber nicht.

[1] Leider ist die alternative Lösung zum Zeitpunkt der Fertigstellung dieser Diplomarbeit auf Grund von internen administrativen Schwierigkeiten an der *Hochschule für Angewandte Wissenschaften Hamburg* noch nicht im realen Einsatz.

Es folgt zunächst eine Bestandsaufnahme und eine Analyse der bisher bestehenden Site des Projektes. Anschließend wird eine Vision für die neue Site entwickelt und diskutiert, hierfür werden auch die Zielgruppen der Nutzer identifiziert. Abschließend wird eine Restrukturierung der Site vorgestellt und die einzurichtenden Anwendungen für die Administration der Site angerissen. Dort werden einige Bildschirmfotos der neuen Site und ihres Administrationsbereiches präsentiert. Der gesamte Quellcode und die notwendigen Datenbankskripte für MySQL sind auf der beigelegten CD vorhanden.

3.2 Bestandsaufnahme

Die Bestandsaufnahme umfaßt die Seiten der folgenden URL's des IKS-Projekts.

- `http://www.informatik.fh-hamburg.de/~robots` bzw. `http://www.informatik.fh-hamburg.de/~kvl`
- `http://www.informatik.fh-hamburg.de/~pioneer`
- `http://www.informatik.fh-hamburg.de/~krabat`
- `http://www.informatik.fh-hamburg.de/~lego`

Die URL `http://www.informatik.fh-hamburg.de/~robots` [wwwCite:IKS] stellt den Einstiegspunkt in die — prinzipiell als Einheit zu sehende — Website dar und dient im wesentlichen nur der Verteilung auf die Unterseiten bzw. Subsites der einzelnen Plattformen (Lego, Pioneer und Krabat).

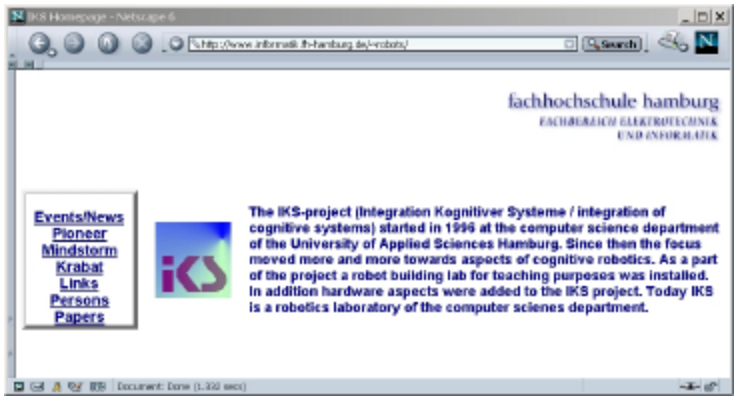

Abbildung 3.1: Bisherige Startseite des IKS-Projektes

3.2.1 Vorhandene Bereiche

Es folgt eine grafische Übersicht über die Site. Die einzelnen Elemente werden dann kurz in ihrer Funktion und inhaltlichen Ausprägung besprochen.

Übersicht über das Angebot

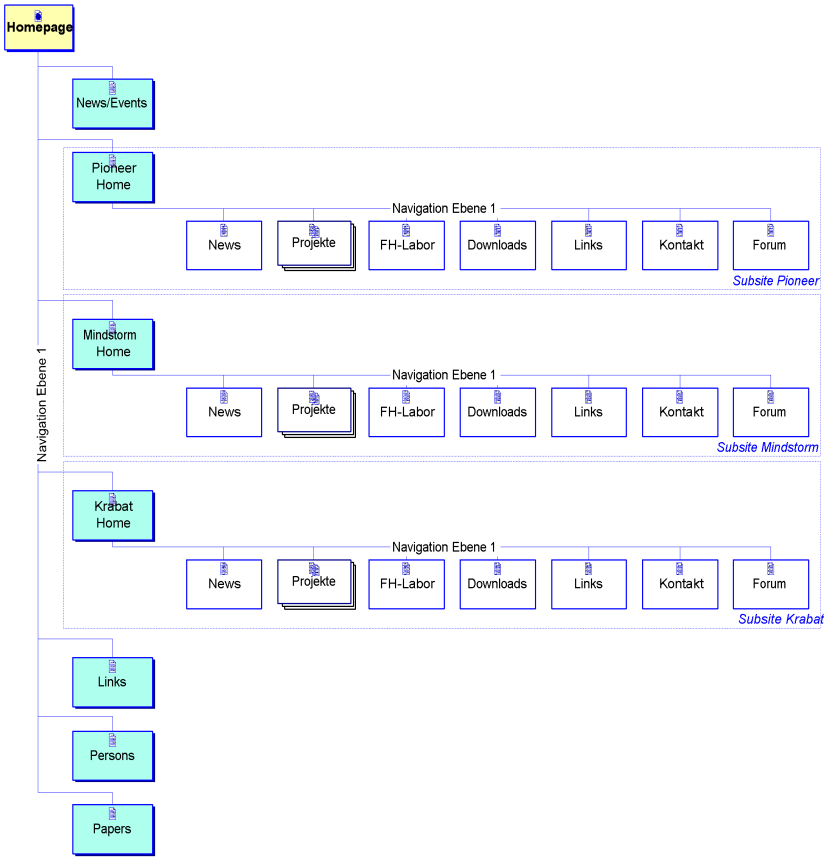

Abbildung 3.2: Grafische Darstellung der vorhandenen IKS-Site

Hauptseiten des Angebotes

Homepage Hier befindet sich ein Absatz mit kurzer englischsprachiger Beschreibung des Projekts (siehe Abbildung 3.1). Ansonsten ist die Homepage nur die Verteilseite auf die Unterseiten (News/Events, Links, Persons und Papers) und die Subsites der einzelnen Systeme (Pioneer, Mindstorm und Krabat).

Events/News Hier befinden sich Bilder von verschiedenen Auftritten des Robotlabores und von einzelnen Abschlußwettbewerben. Die Bildersamm-

lungen sind von der Eingangsseite, aus einer Liste mit den verlinkten Überschriften heraus zu erreichen (siehe Abbildung 3.3).

Die Bildersammlungen sind wie folgt ausgestattet: Übersichtsseite mit verlinkten Thumbnails (verkleinerten Abbildungen der Fotos); die Thumbnails linken nur direkt auf die Bilddatei; keine weitere Navigation; keine Beschreibung der Bildersammlung; keine Einzelbeschreibung der Bilder.

Es sind entgegen der benutzten Bezeichnung *Events/News* keine Nachrichten vorhanden (z.B. Ankündigungen von Wettbewerben oder neuen Projekten in den Subsites).

Abbildung 3.3: Bisheriger Event/News-Bereich mit einer Bildersammlung

Pioneer, Mindstorm, Krabat Für die einzelnen Plattformen, die im IKS-Projekt eingesetzt werden, existieren in ihrer Struktur gleiche Subsites zum jeweiligen System. Die Subsites werden weiter unten beschrieben.

Die folgenden Elemente (Links, Person, Papers) sind listenförmige Sammlungen von Materialien. Ihnen gemeinsam ist eine Grundstruktur im Layout und die nur sehr knappe Darstellung des Inhaltes (siehe Abbildung 3.4).

Links Auf einer Seite befindet sich eine kategorisierte Linkliste mit verlinkten Kurztiteln.

Person Eine Liste der beteiligten Mitarbeiter, mit Verweisen auf die jeweiligen Homepages, die in Eigenverantwortung gepflegt und auf sehr unterschiedlichem Niveau gewartet werden.

Papers Eine kategorisierte Liste der im Zusammenhang mit dem IKS-Projekt erstellten Veröffentlichungen mit Autorenangaben, Titeln, Publikationsangaben und — soweit elektronische Versionen verfügbar — Links zum Download der Publikation.

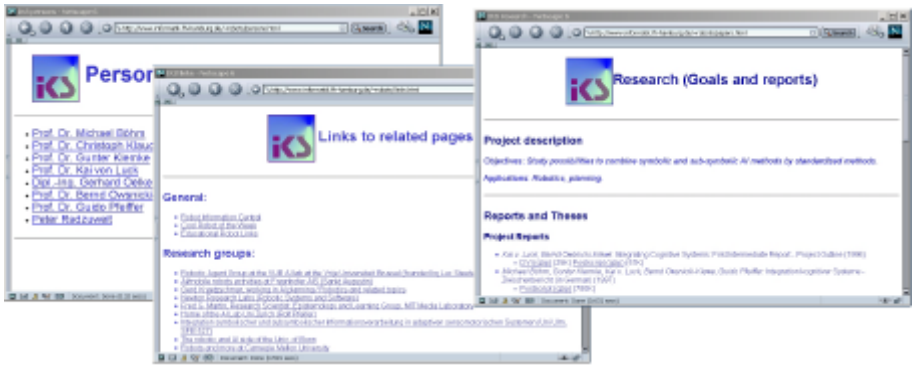

Abbildung 3.4: Bisherige Seiten des IKS-Bereiches: Personen, Links und Papers

Elemente der Subsites zu den einzelnen Plattformen

Im IKS-Projekt werden drei verschiedene Hardwareplattformen für den Roboterbau eingesetzt. Zu jeder Plattform existiert eine eigenen Subsite. Die Strukturen der Subsites sind identisch.

SubHome Die Einstiegsseiten stellen sich mittlerweile unterschiedlich dar. Während beim Pioneer und Mindstorm keine Informationen geliefert, sondern nur große Bilder der Plattformen präsentiert werden, ist auf der Krabatsite ein historischer Abriss über die Entstehungsgeschichte des Boards zu finden.

Anmerkung: In der Abbildung ist ebenfalls ein Symptom der schwierigen Wartung und Pflege der Site zu erkennen. Das Logo der Krabatsite ist fehlerhafterweise das Pioneerlogo.

Abbildung 3.5: Bisherige Einstiegsseiten: Krabat, Pioneer, Mindstorm

News Auflistungen von verschiedenen abgeschlossenen Fakten mit Datum. Chronologische Ordnung mit ältestem Eintrag oben.

Projekte Auflistung der (abgeschlossenen) Projekte. Verlinkung auf Material der Projekte bzw. eigene Projektseiten. Verlinkung zur E-Mail-Adresse der Projektmitarbeiter/-autoren (wobei bei den Studien- und Diplomarbeiten typischerweise E-Mail-Adressen der Fachhochschule angegeben wurden, diese aber kurz nach dem Studium ihre Gültigkeit verlieren).

FH-Labor Hier werden Materialien zum eingesetzten Robot bzw. Originaldokumentationen der Entwicklungsumgebungen zum Download angeboten (Krabat) oder die Ausstattung des Labors aufgezählt (Pioneer). Bei der Mindstormsite ist dieser Bereich nach einem Fehler bei der Wartung der Site leer.

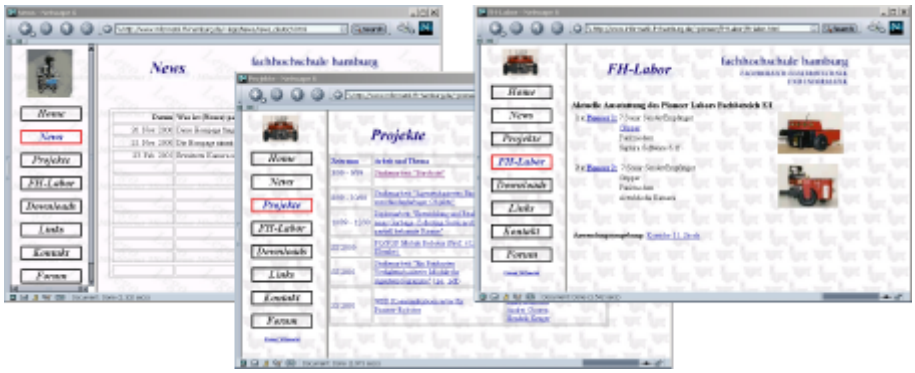

Abbildung 3.6: Bisherige News-, Projekte- und Laborseiten

Downloads Weitere Datenblätter bzw. Diplom- und Studienarbeiten werden zum Download angeboten. Die Präsentation erfolgt mittels simpler Tabellenform mit Datum, Datei, Kurzbeschreibung und teilweise Größenangabe.

Linklisten Verweise auf andere Webseiten. Mit Kurzbeschreibung und verlinktem Titel. Teilweise Angabe der URL. Ein Problem der Linklisten ist, daß keinerlei Verbindung untereinander bzw. mit der IKS-Linkliste besteht, so tauchen Links doppelt auf oder werden wegen ihrer Zuordnung nicht wiedergefunden. Die Wartung der doppelten Einträge ist schwierig, da Aktualisierungen häufig nur teilweise durchgeführt werden.

Kontakt Liste mit Personen, ihren Funktionen und ihren E-Mail-Adressen.

Forum Ein Verweis auf ein Forum zum Thema ist zwar in der Navigation angelegt, die Foren selbst sind aber nicht vorhanden (nur Baustellenschilder).

Abbildung 3.7: Bisherige Link-, Kontakt- und Forenseiten

3.2.2 Art der Site

Die bisherige Struktur der Site ist in Art und Umfang eher eine Broschureware, also eine kurze Beschreibung der Tätigkeiten des IKS-Projektes. Sie stellt im wesentlichen 'historische' Daten dar und liefert kaum aktuelle Informationen über die Arbeit des IKS-Projektes oder laufender Unterprojekte. Es gibt außer den angegebenen E-Mail-Kontakten keine Interaktionsmöglichkeiten.

Inhaltlich hat die Site in hohem Maße dokumentarischen Charakter. Die gebotenen Informationen sind daher von ihrem zeitlichen Charakter eher statische Informationen (im Sinne des Abschnittes 1.2.1). Änderungen der einzelnen Seiten betreffen vornehmlich die Ergänzung von Listen um weitere Einträge von neu hinzugekommenen Informationen.

Als Zielpublikum der bisherigen Site lassen sich vorrangig Personen identifizieren, die einen ersten Eindruck von der Arbeit des IKS-Projekts gewinnen wollen. Für eine weitergehende Beschäftigung mit dem Thema ist die angebotene Information nur bedingt geeignet und auch nicht sehr zeitnah an den laufenden Projekten.

Für die Organisation oder die Durchführung kollaborativen Arbeitens innerhalb des IKS-Projektes werden keine Werkzeuge angeboten.

3.2.3 Probleme der Site

Im Folgenden werden einige Probleme der Site betrachtet, die unabhängig von einer inhaltlichen Neuorientierung existieren.

Probleme für den Besucher/Nutzer

Besucherseitig zeigt die jetzige Site einige Schwachstellen, die sich im wesentlichen in drei Punkten zusammenfassen lassen.

Informationsarchitektur Das gesamte Erscheinungsbild der Site und ihrer Subsites ist uneinheitlich. Dadurch wird der Zusammenhang der Themen

mit dem IKS-Projekt verschleiert und die Site zerfällt in einzelne Teile. Hierzu paßt auch, daß es keine Möglichkeit zur Navigation zwischen den einzelnen Bereichen gibt. Die Subsites sind gegeneinander so stark isoliert, das die Navigation zwischen den Subsites immer über die Einstiegsseite erfolgen muß. Die Isolation der Subsites geht sogar so weit, daß es noch nicht einmal einen Rücklink aus den Subsites zur Einstiegsseite des IKS-Projektes gibt.

Es gibt keinen Ort, an dem übergeordnete bzw. von den einzelnen Plattformen unabhängige Themen präsentiert werden können, obwohl auch solche Themen vorhanden sind.

Es gibt in der Site einige reine Verteilseiten, deren Informationsgehalt gegen null geht. Der Einsatz von reinen Verteilseiten ist nicht mehr „State of the Art“, da die Geduld der Nutzer bei jeder Verteilebene auf die Probe gestellt und dadurch die Abbruchquote vor Erreichen der gewünschten Information erhöht wird[2].

Bei den Problemen auf konzeptioneller Ebene ist vor allem die uneinheitliche Ablage von Informationen zu nennen. So liegen Dokumentationen zum Teil unter der Rubrik *Downloads* oder unter *FH-Labor*. Auch die Ablage von Arbeiten allgemeinerer Natur ist schwierig. So findet sich z.B. eine Arbeit über pädagogische Aspekte im Mindstorm-Bereich wieder, obwohl die Arbeit den generellen Einsatz von Robotern im Informatikunterricht behandelt. Ähnliches gilt auch für die Linksammlungen. Eine Lösung dieses Problems wäre die mehrfache Nennung von Informationen. So könnten Arbeiten beispielsweise in den Subsites aufgeführt werden und gleichzeitig in einer Gesamtliste aller Arbeiten unter IKS/Papers wieder aufgeführt werden. Auf diese Weise ist es leichter, einen Überblick über die Tätigkeit des IKS-Projektes zu bekommen, und auch schwierig einzuordnende Arbeiten können besser gefunden werden.

Technische Probleme Durch die Pflege der Site mittels einfacher HTML-Seiten bzw. einfacher Serverside-Includes für Kopf- und Fußteile entstehen sehr leicht Fehler, die zu nicht erreichbaren Seiten führen oder zu fehlerhaften Verlinkungen der Subsites (so waren z.B. eine Zeitlang unterhalb der Krabat-Subsite die Informationen zum Pioneer abgelegt).

Fehlende/komplizierte Interaktivität Die vorhandenen Verweise auf (leider nicht angelegte Foren) lassen ebenso wie die mehrfach auftretenden Hinweise auf Mitarbeit und Ergänzung darauf schließen, daß in der ursprünglichen Konzeption der Site das Thema *Interaktivität* bzw. *Beteiligung der Besucher* durchaus eine Rolle gespielt hat.

Leider wird die Interaktivität nicht unterstützt, wodurch aus Besuchern der Site keine Nutzer der Site werden. Es wird viel nutzbares Potential an

[2] In kommerziellen Websites wird mit Abbruchquoten pro Verteilebene zwischen 30 und 50 Prozent gerechnet.

Kommunikation und Kollaboration (vor allem unter den Studenten des IKS-Projektes) verschenkt.

Probleme der Administration

Es gibt keine Möglichkeit, auf einfache Weise der Site neue Daten hinzuzufügen. Dies erstreckt sich auf alle Bereiche, angefangen bei den Terminen und News bis hin zu den Linklisten oder Projektseiten.

Die einzige Administration besteht in der manuellen Veränderung der HTML-Seiten. Hierbei ist insbesondere die Gefahr der gleichzeitigen überlappenden Veränderung von zwei unterschiedlichen Autoren gegeben, weil die Seiten von den Autoren zum Teil über Tage hinweg in Kopie auf heimischen Rechnern bearbeitet werden. Ebenso ist das Hinzufügen beispielsweise eines Links so aufwendig, daß die sinnvolle Ergänzung der Informationen der Site oft aus Zeitmangel unterbleibt. Dadurch wirkt die Site nicht lebendig und ein Besucher wird nicht zum wiederholten Besuch animiert.

3.3 Vision

Als Leitmotiv der alternativen Struktur und inhaltlichen Ausrichtung der Site des IKS-Projektes steht in der ersten Stufe die Vereinfachung der Administration der Site sowie die Schaffung eines einheitlichen Erscheinungsbildes des IKS-Projektes und seiner Teilgebiete im Vordergrund.

Die Nutzer (und Administratoren bzw. Autoren) der Site sollen sich wohlfühlen, Informationen leicht erreichen und einen positiven Eindruck vom IKS-Projekt bekommen.

Die aktive Mitarbeit der Nutzer ist das Ziel einer weiteren Ausbaustufe, deren gewünschtes Ergebnis es ist, die Verstärkung synergetischer Effekte aus dem Wissen der Nutzer und dem der Teilnehmer des IKS-Projektes zu provozieren. Die zweite Ausbaustufe wird nicht innerhalb dieser Arbeit realisiert. Allerdings werden die Grundlagen für die zweite Ausbaustufe gelegt. Dies sind insbesondere die Einrichtung einer Session- und Nutzerverwaltung.

Neudefinition der Zielgruppen Als Zielgruppen für die Site stehen vier Hauptgruppen an. Studenten des IKS-Projektes, Forschungsgruppen anderer Hochschulen (auch international), an Robotik Interessierte und nicht zuletzt IKS-Förderer und Berichterstatter. Diese Gruppen besitzen sicherlich unterschiedliche Fokusse, die im Folgenden kurz umrissen werden.

Aus studentischer Sicht soll die Site eine aktive Zusammenarbeit und gegenseitige Inspiration fördern. So sollte es beispielsweise möglich sein, eine Idee zu einem Thema grob zu umreißen und anderen die Ausarbeitung der Idee anzubieten oder bei auftretenden Problemen inter pares neue Lösungs-

ansätze zu erarbeiten[3]. Ebenso ist ein Fundus an Ideen und oder Arbeitsergebnissen sehr interessant, denn dann muß das sprichwörtliche Rad nicht stets neu erfunden werden und man kann auf den Ergebnissen anderer weiterführende Entwicklungen aufbauen. Ein weiterer interessanter Aspekt für die Studenten ist die Information über mögliche Studien- und Diplomarbeiten im IKS-Projekt.

Für Forschungsgruppen anderer Hochschulen steht die Information über aktuelle Arbeiten im Vordergrund. Aus dieser Kommunikation können sehr gut Kooperationen entstehen, die gegenseitig befruchtend sind.

Ein an Robotik im allgemeinen Interessierter (möglicherweise auch ein Laie) findet auf der Site den Kontakt zu Gleichgesinnten und einen guten Einstieg in das Netzwerk der Robotik. Im Idealfall entwickelt sich die Site zu einem regen Treffpunkt zum Thema[4]. Hiervon können alle Nutzer der Site partizipieren. *Anmerkung:* Zudem ist dem Autoren keine wirklich aktive allgemeine Community zum Thema Robotik im Web bekannt (weder deutsch- noch englischsprachig).

Die Gruppe der IKS-Förderer und der Berichterstatter über das Projekt hat eine politische Komponente, die nicht aus den Augen verloren werden darf. Für jedes Projekt ist eine gute Öffentlichkeitsarbeit notwendig. Hier sind vor allem Erfolgsmeldungen und Termine für Vorstellungen der Arbeit des Projektes interessant. Auch ein Pressespiegel ist in diesem Zusammenhang relevant.

Der Weg zum Ziel Die zur Verfügung stehende Zeit zur Administration ist natürlicherweise begrenzt. Ebenso wie die zur Verfügung stehenden Ressourcen zur Erstellung neuen Inhalts. Deshalb muß zum einen die Administration stark vereinfacht werden, dies wird mit einem formularbasierten Adminstrationsbereich erfolgen. Zum anderen kann — in der zweiten Ausbaustufe — die Quelle des nutzergenerierten Inhalts (User Contributed Content) zur Erstellung weiteren Inhalts herangezogen werden. Dies hat zudem noch den Vorteil, daß die Nutzer selbst die Richtung von für sie spannenden Themen beeinflussen können und so die Site interessanter wird.

Um die angesprochenen hohen Ziele zu erreichen, muß zuerst die Technik der Site runderneuert werden. Die Inhaltsdaten der Site müssen in einer Datenbankstruktur abgelegt und dynamisch publiziert werden. Desweiteren müssen Interaktionen und eigene Aktivitäten der Nutzer in die Site eingebracht werden, dies geschieht mit dem Community-Ansatz. Als technologische Basis kommt hierbei das in dieser Arbeit entwickelte Framework zum Einsatz.

[3]Dies wäre Bestandteil der zweiten Ausbaustufe mit Einführung von interaktiven Anwendungen.

[4]Dies wäre ebenfalls Bestandteil der zweiten Ausbaustufe.

3.4 Restrukturierung

Nachfolgend einige Maximen zur Restrukturierung der Site.

Beseitigung navigatorischer Probleme Um die navigatorischen Probleme zu beseitigen, wird eine durchgehend zweistufige Navigationsstruktur eingeführt. Dies bedeutet, daß über die erste Ebene der Navigation von jeder Seite der Site die Startseiten der Hauptbereiche (IKS, LegoLab, Pioneer und Krabat) der Site erreichbar sind. Die Navigation innerhalb eines Hauptbereiches wird über eine zweite Navigationsebene durchgeführt.

Vereinheitlichtes Design Um den Umfang des IKS-Projektes und seiner Unterprojekte deutlicher zu machen, muß auf eine in sich geschlossene und konsistente Gestaltung der Site in allen ihren Elementen und Bereichen geachtet werden. (In der Realisierung wird dies durch den objektorientierten Ansatz auch für die Gestaltung erreicht.)

Beseitigung von Zero-Content-Pages Die reinen Verteilseiten ohne eigentliche Inhalte werden durch Zusammenfassungen der betreffenden Inhalte eines Bereiches und eine Einführung in den Bereich ersetzt. Die Zusammenfassungen können z.B. die aktuellen Termine, die neuesten Subprojekte oder auch frisch hinzugekommenes Material der Linksammlung des Bereiches sein. Wichtig ist hier, daß gezeigt wird, was in dem Bereich geschieht. Die Lebendigkeit der Forschung muß hier dokumentiert werden. (Die Erstellung der Zusammenfassungsseiten erfolgt automatisiert durch die Ausnutzung des datenbankgestützten Publizierens.)

Größere Aktualität Durch die vereinfachte Administration der Site kann viel zeitnaher und leichter die Arbeit des IKS-Projektes dokumentiert werden. Weiterhin gewinnt die Site in der zweiten Ausbaustufe durch die nutzergenerierten Inhalte an Aktualität und Umfang.

Stärkere Interaktivität Die Mitarbeit von Nutzern der Site wird durch ein formularbasiertes Vorschlagswesen für z.B. Links oder Buchtipps oder Besprechungen von Links und Büchern nicht nur eingefordert, sondern auch gefördert. In der zweiten Ausbaustufe kann der Kern der Interaktivität durch die Community für Studenten vor Ort und für Robotinteressierte gebildet werden. Diese Community kann zunächst aus der identitätsbildenden Maßnahme der Screennames und Web-Visitenkarten bestehen, mit deren Hilfe in der Forensammlung zu den Themen des IKS-Projektes die Beiträge namentlich gekennzeichnet werden, wobei die Screennames der Nutzer gegen Fremdbenutzung geschützt sind. Ebenso wird so die Kontaktaufnahme unter den

Nutzern der Site gefördert. Dies kann beispielsweise durch ein Messagingsystem erfolgen, über das die Nutzer der Site kommunizieren können. Ein weitergehendes Konzept ist das des WikiWebs [Leuf, Cunningham 2001], in dem die Nutzer der Site die Seiten selbst bearbeiten können und so zu Autoren der Site oder eines definierten Teilbereiches werden.

3.5 Realisierung

Für die exemplarische Realisierung werden nicht alle in der Vision angesprochenen Teile umgesetzt. Es werden nur einige grundlegende Bereiche definiert und implementiert (siehe Abschnitt 3.5.1). Die ausgewählten Bereiche erlauben den Betrieb der Site mit fast allen bisherigen Möglichkeiten, aber zusätzlich unter Berücksichtigung aller in Abschnitt 3.4 festgelegten Maximen. Im Abschnitt 3.5.1 wird die Notwendigkeit einiger Hilfsanwendungen deutlich, diese werden im Abschnitt 3.5.2 besprochen. Aus den Anwendungen ergeben sich auch die im Projekt notwendigen Rollen, die in Abschnitt 3.5.3 dargelegt werden. Als Abschluß wird in Abschnitt 3.5.4 die Trennung von Layout und Inhalt durch das Framework demonstriert.

3.5.1 Definition der Anwendungen

Die vier wichtigsten Anwendungen auf der alternativen IKS-Site werden in diesem Abschnitt dargestellt: Nachrichten, Links, Projektvorstellungen und Einzelseiten. Die Definitionen der Anwendungen folgen dem Muster:

- **Beschreibung der Anwendung** Hier wird der Sinn der Anwendung und ihre Besonderheiten dargestellt.
- **Definition der Datenstruktur** Hier wird die zu Grunde liegende Datenbeschreibung mit ihren wichtigsten Elementen dargestellt. Die vollständigen Datenbeschreibungen finden sich auf der beigelegten CD wieder.
- **Aussehen der Anwendung** Hier werden Bildschirmaufnahmen der wichtigsten Administrationsseiten und der Darstellung auf Seiten der Nutzer gezeigt.

Nachrichten

Die Nachrichten dienen vor allem der Mitteilung von besonderen Ereignissen im Rahmen des IKS-Projektes. Es handelt sich hierbei z.B. um Ankündigungen von Veranstaltungen wie dem Abschlußwettbewerb des LegoLab oder einer besonderen Vorführung von Forschungsergebnissen im Rahmen eines Kolloquiums, aber auch um Mitteilungen bezüglich des Laborbetriebes oder von Neuanschaffungen. Nachrichten können meistens einem spezifischen Bereich des IKS-Projektes — dem LegoLab, den Pioneers oder dem Krabat-

board — zugeordnet werden, es gibt aber auch Nachrichten von allgemeinerer Art, die das gesamte Projekt betreffen. Aus diesem Grunde gibt es vier Bereiche für Nachrichten (IKS, LegoLab, Pioneer und Krabat). Damit nun wichtige Neuigkeiten nicht an vier verschiedenen Stellen gesucht werden müssen, ist eine Nachricht als sogenannte Top-Nachricht markierbar. Solche Top-Nachrichten werden auf der Startseite des IKS-Projektes mit ihrem Titel, einem kurzen Teaser[5] und einem Link auf die vollständige Nachricht präsentiert. Auf den Seiten der vollständigen Nachricht werden die Titel, Teaser und Links der neuesten Nachrichten des jeweiligen Bereiches gezeigt, so kann der Nutzer direkt von Nachrichtentext zu Nachrichtentext gelangen. In einem Archiv der Nachrichten werden alle bisherigen Nachrichten zugänglich gemacht. Die Verwaltung der anzuzeigenden Nachrichtenteaser geschieht vollautomatisch. Auf der Startseite werden nur die jeweils neuesten Topnachrichten der Bereiche gezeigt, während auf den Nachrichtenseiten die letzten 10 Nachrichten gezeigt werden. Als letzte Forderungen an die Nachrichten sind noch aufzuführen, daß sie einem Autoren zugeordnet, daß sie gelöscht und daß sie zwar schon erstellt, aber noch nicht freigegeben sein können. Mit der letzten Forderung wird es möglich, Nachrichten vor der Freigabe zunächst von anderen Redakteuren gegenlesen zu lassen[6].

Definition der Datenstruktur Um die gewünschte Funktionalität bereitzustellen, setzt sich die Informationseinheit Nachricht also im wesentlichen aus folgenden Informationselementen zusammen: ID, Bereich, Topnews, Titel, Teaser, Text, Autor und Status.

```
class NewsDescription extends WCK_Description {
    function NewsDescription(){
        global $SCRIPT_NAME,$special;
        WCK_Description::WCK_Description();

        $this->table='news';
        $this->defaultOrder='status,date,top desc';
        $this->defaultRestriction='special="'.$special.'" and'.
                                  ' (status=0 or status=1) ';
        $this->field['id']=new WCK_ElementID('id',$this->table);
        $this->field['special']=new WCK_Element('special','Bereich',$special);
        $this->field['special']->changeable=false;
        $this->field['top']=new WCK_ElementCheckBox('top','Topnews');
        $this->field['top']->addon='special='.$special;  // Kopfzeilen
        $this->field['titel']=new WCK_ElementText('titel','Titel','',45,150);
        $this->field['titel']->addon='special='.$special; // Kopfzeilen
        $this->field['titel']->setMustNotEmpty();
        $this->field['intro']=new WCK_ElementText('intro','Teaser','',45,150);
        $this->field['body']=new WCK_ElementTextarea('body','Text','',35,10);
```

[5]Neugier erregendes Werbeelement [Brockhaus 2000].

[6]Es wird in diesem Projekt allerdings kein Workflow definiert, der ein Vier-Augen-Prinzip, also eine der Veröffentlichung vorhergehende Kontrolle durch einen Chefredakteur vorschreibt. Dies wurde unterlassen, damit die Verwaltung der Site so unbürokratisch wie möglich ist.

```
        $this->field['body']->setMustNotEmpty();
        $this->field['autor']=new WCK_ElementText('autor','Autor','',25,25);
        $this->field['status']=new WCK_ElementStatusDel('status','Status');
        $this->field['status']->addon='special='.$special;// Kopfzeilen
    }
}
```

Aussehen der Nachrichtenanwendung Die Administrationssicht der Nachrichten wird in der Abbildung 3.8 gezeigt. Sie besteht im wesentlichen aus den Listen der Nachrichten des Bereiches (im Hintergrund liegend) und dem Formular zur Bearbeitung einer Nachricht. An dieser Stelle sei noch auf den Quelltext des Formulares verwiesen, der im Anhang G vollständig wiedergegeben ist.

Für die Nutzer der Site stellen sich die Nachrichten wie in der Abbildung 3.9 dar. Gezeigt wird im Hintergrund die Startseite mit den aktuellen Topnews-Teasern und im Vordergrund die vollständige Ansicht des Nachrichtentextes mit den Teasern der weiteren Nachrichten des Bereiches (in diesem Fall des LegoLabs).

Beide Abbildungen (3.9 und 3.8) zeigen dieselbe Situation der IKS-Site. Interessant ist hier, das nur die aktuellste Topnews aus dem Bereich LegoLab auf der Startseite gezeigt wird.

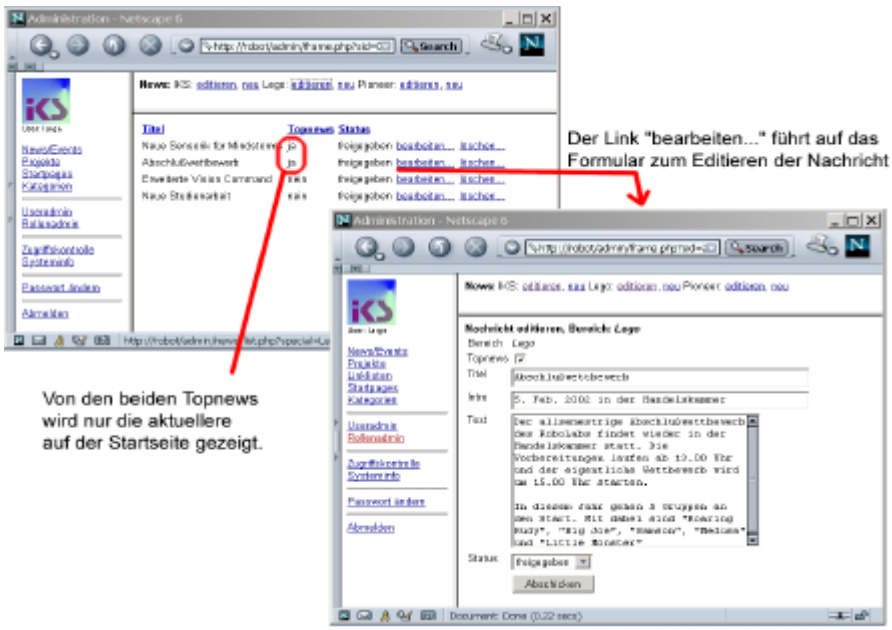

Abbildung 3.8: Administrationssicht der News

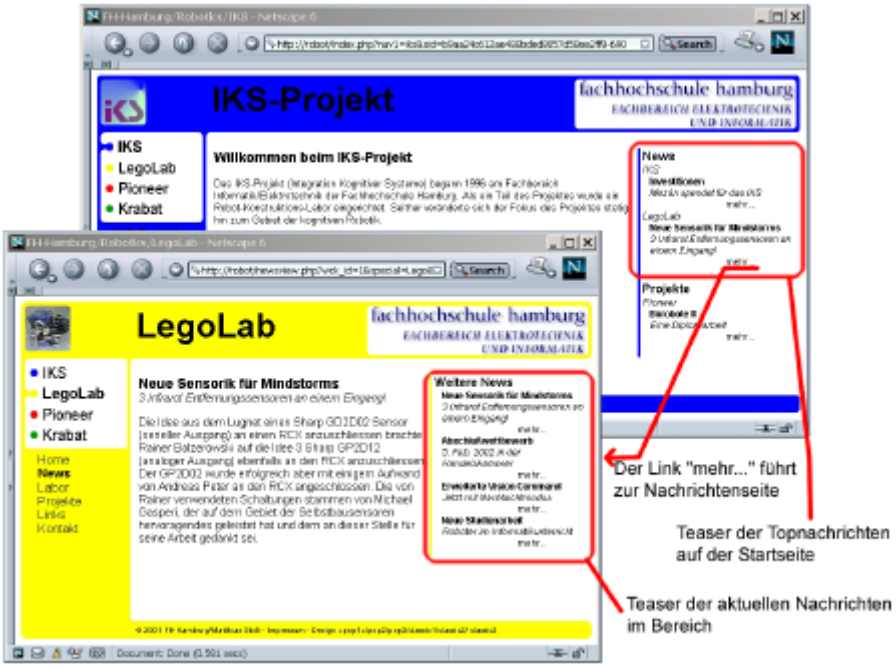

Abbildung 3.9: Nutzersicht der News

Linklisten

In der Anwendung Linklisten werden thematische Verweise in das World Wide Web zusammengestellt. Zu einem Link werden die folgenden Daten gesammelt: Titel, URL und Kurzbeschreibung. Die Verweise müssen, um die Übersicht zu behalten, in Kategorien eingeordnet werden. Diese Kategorien können von Bereich zu Bereich des IKS-Projektes unterschiedlich sein, zu dem kann der Bedarf nach neuen Kategorien entstehen, aus diesem Grund können die Kategorien nicht bei der Programmierung der Site festgelegt werden. Es besteht damit die Notwendigkeit, eine Hilfsanwendung *Kategorien* einzurichten (siehe Abschnitt 3.5.2).

Definition der Datenstruktur Das Besondere an der Datenbeschreibung zur Linkliste ist die Kategorieselektion `cat`. Es ist ein Eingabeelement, dessen Daten aus einer anderen Datenbanktabelle generiert werden.

```
class LinkDescription extends WCK_Description {
    function LinkDescription(){
        global $SCRIPT_NAME,$special;
        WCK_Description::WCK_Description();

        $this->table='links';
        $this->defaultRestriction='special="'.$special.'" and (status=0)';
```

```
        $this->field['id']=new WCK_ElementID('id',$this->table);
        $this->field['special']=new WCK_Element('special','Bereich',$special);
        $this->field['special']->changeable=false;

        // je nach Bereich werden unterschiedliche Kategorien ausgewaehlt.
        switch($special){
        case 'IKS';
            $this->specialRestrikt='ikscat=1';
            break;
        case 'Lego';
            $this->specialRestrikt='legocat=1';
            break;
        case 'Pioneer';
            $this->specialRestrikt='pioneercat=1';
            break;
        case 'Krabat';
            $this->specialRestrikt='krabatcat=1';
            break;
        default;
            $this->specialRestrikt='0';
        }
        $this->field['cat']=new WCK_ElementSelectTable('cat','Kategorie',
                                 'category','id','title',
                                 "status=0 and art='link' and ".
                                 $this->specialRestrikt,'ord');
        $this->field['titel']=new WCK_ElementText('titel','Titel','',45,150);
        $this->field['titel']->setMustNotEmpty();
        $this->field['url']=new WCK_ElementText('url','URL','',45,150);
        $this->field['url']->setMustNotEmpty();
        $this->field['comment']=new WCK_ElementTextarea('comment',
                                                    'Beschreibung','',35,5);

        $this->field['ikscat']= new WCK_ElementCheckBox('ikscat','IKS');
        $this->field['legocat']= new WCK_ElementCheckBox('legocat','LegoLab');
        $this->field['pioneercat']= new WCK_ElementCheckBox('pioneercat','Pioneer');
        $this->field['krabatcat']= new WCK_ElementCheckBox('krabatcat','Krabat');
    }
}
```

Aussehen der Linklisten In der Abbildung 3.10 wird das aufgeklappte Selektionsfeld der Kategoriezuordnung gezeigt, hier für den Bereich *LegoLab*. Die vorhandenen Einträge werden in der Hilfsanwendung *Kategorisierungen* definiert.

In der Nutzersicht der Linklisten (Abbildung 3.11) wird die Tatsache ausgenutzt, daß die Ausgabeobjekte für Listen nicht auf Datenbeschreibungen beruhen. Auf diese Weise ist es möglich, eine Darstellung von Listen (die eigentlichen Links) innerhalb von Listen (den Kategorien) zu generieren. Wie bei allen Applikationen gilt auch hier, daß nur ein Skript für die Darstellung der Linklisten in allen Bereichen und allen Designs zuständig ist. Das Skript ist im Anhang G wiedergegeben.

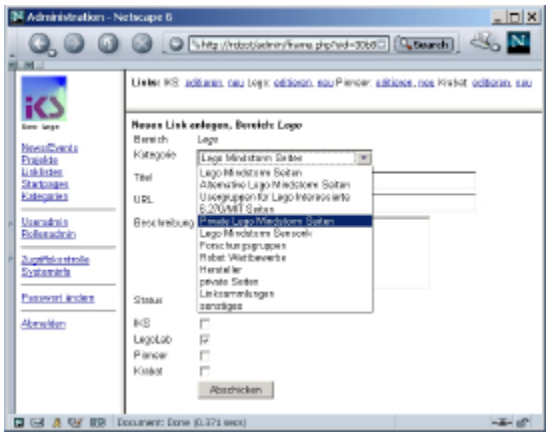

Abbildung 3.10: Administrationssicht der Linklisten mit Kategorieselektion

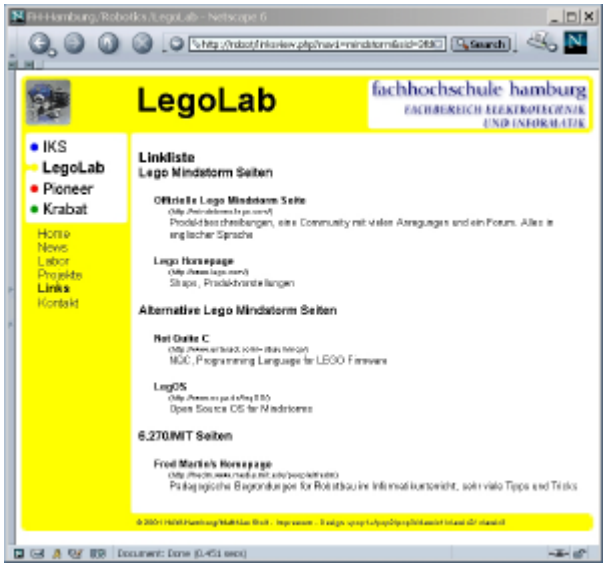

Abbildung 3.11: Nutzersicht der Linklisten

Projektvorstellungen

Bei den Projektvorstellungen handelt es sich um eine relativ simple Anwendung, die sehr schlicht definiert ist. Ähnlich wie bei den Nachrichten gibt es Titel, Teaser, Text, Status und die Kennzeichnung als Topprojekt. Wieder werden die aktuellsten Topprojekte der Bereiche *LegoLab, Pioneer* und *Krabat* auf der IKS-Startseite präsentiert. Zusätzlich gibt es, ähnlich wie bei den Linklisten, wieder die Notwendigkeit der Kategorisierung. Eine Erweiterung wäre die Möglichkeit, mit einem Projekt Dateidownloads zu verknüpfen, dies wurde aber im Rahmen der Arbeit nicht durchgeführt.

Definition der Datenstruktur

```
class ProjectsDescription extends WCK_Description {
    function ProjectsDescription(){
        global $SCRIPT_NAME,$special;
        WCK_Description::WCK_Description();

        $this->table='projects';
        $this->defaultOrder='status,date,top desc';
        $this->defaultRestriction='special="'.$special.
                                  '" and (status=0 or status=1) ';

        $this->field['id']=new WCK_ElementID('id',$this->table);
        $this->field['special']=new WCK_Element('special','Bereich',$special);
        $this->field['special']->changeable=false;
        $this->field['cat']=new WCK_ElementSelectTable('cat','Kategorie',
                                'category','id','title',
                                "status=0 and art='project'",'ord');
        $this->field['top']=new WCK_ElementCheckBox('top','Topprojekt');
        $this->field['titel']=new WCK_ElementText('titel','Titel','',45,150);
        $this->field['titel']->setMustNotEmpty();
        $this->field['intro']=new WCK_ElementText('intro','Intro','',45,150);
        $this->field['body']=new WCK_ElementTextarea('body','Text','',35,10);
        $this->field['body']->setMustNotEmpty();
        $this->field['autor']=new WCK_ElementText('autor','Autor','',25,25);

        $this->field['status']=new WCK_ElementStatusDel('status','Status');
    }
}
```

Einzelseitenteile

Um die Vorstellungsabschnitte auf der Startseite IKS und den jeweiligen Unterprojekten LegoLab, Pioneer und Krabat ebenfalls im Administrationsbereich editierbar zu machen, wird die Anwendung Einzelseitenteile eingeführt. Für die oben aufgeführten Teile des Projektes werden hier formularbasierte Eingabemöglichkeiten bereitgestellt. Zur Zeit werden nur zwei Informationselemente benötigt: eine Textzeile für die Überschrift und ein Textfeld für den eigentlichen Text. Die Eingaben für den Text werden bis auf weiteres auf den jeweiligen Seiten direkt als HTML-Teile ausgegeben. Dies ist aber nur eine Notlösung. Im Sinne der Trennung von Layout und Inhalt sollte eigentlich die Eingabe von direktem HTML-Code unterbleiben. Sinnvoll wäre hier ein von der Klasse `WCK_ElementTextArea` abgeleitetes Eingabeelement, in dem eine eigenständige Auszeichnungssprache (oder auch eine XML Anwendung) ein nur semantisches Markup vornimmt, das dann mit Hilfe eines Formatierungsobjektes (oder auch XSLT) in HTML übersetzt und so der Inhalt in die Seite eingefügt wird. Damit ist dann wieder eine Trennung von Inhalt und Layout erreicht. Hier liegt ein Ansatzpunkt für den weiteren Ausbau des Frameworks vor (siehe Abschnitt 4.2).

Definition der Datenstruktur Die Datenstruktur für die Einzelseitenteile besteht vor allem aus der nicht änderbaren Kennung für den darzustellenden Teil, einem Titel (welcher besonders formatiert wird) und dem eigentlichen Text.

```
class StartPagesDescription extends WCK_Description {
    function StartPagesDescription(){
        global $SCRIPT_NAME,$special;
        WCK_Description::WCK_Description();

        $this->table='startpages';
        $this->defaultOrder='';
        $this->defaultRestriction='special="'.$special.'"';

        $this->field['id']=new WCK_ElementID('id',$this->table);
        $this->field['special']=new WCK_Element('special','Bereich',$special);
        $this->field['special']->changeable=false;
        $this->field['titel']=new WCK_ElementText('titel','Titel','',45,150);
        $this->field['titel']->setMustNotEmpty();
        $this->field['body']=new WCK_ElementTextarea('body','Text','',35,10);
        $this->field['body']->setMustNotEmpty();
    }
}
```

Aussehen der Anwendung In der Abbildung 3.12 wird gezeigt, wie die Felder des Administrationsbereiches in der Nutzersicht abgebildet werden.

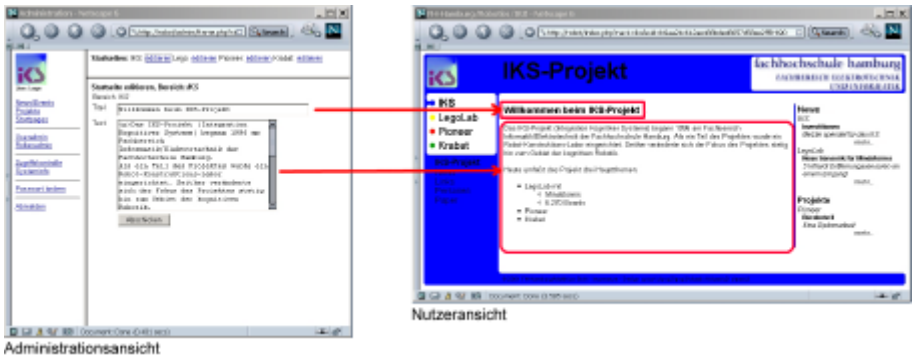

Abbildung 3.12: Administrations- und Nutzersicht für Einzelseitenteile

3.5.2 Hilfsanwendungen

Nutzer- und Rechteverwaltung

Abbildung 3.13: Anmeldung am Administrationsbereich

Eine große Hilfsanwendung ist die generalisierte Nutzer- und Rechteverwaltung. Hier werden die Zugangsberechtigungen zu den verschiedenen Administrationsbereichen festgelegt (siehe auch Abschnitt 3.5.3). Bevor ein Nutzer den Administrationsbereich der Site betreten darf, muß er sich zunächst authentifizieren. Dies erfolgt über einen Namen und ein Passwort. Nach der erfolgreichen Anmeldung werden dem Nutzer die Bereiche präsentiert, zu denen er die entsprechenden Rechte hat. Alle anderen Bereiche werden den Nutzern gar nicht gezeigt (siehe Abbildung 3.13). Die eigentliche Rechtevergabe wurde schon im Abschnitt 2.3.11 dargestellt.

Kategorisierungen

Für einige Anwendungen (Links, Projekte) werden Kategorisierungen benötigt. Um die zur Verfügung stehenden Kategorien nicht durch feste Codierung einzuschränken, wird die Hilfsanwendung *Kategorisierungen* eingeführt. Mit ihrer Hilfe werden die Kategorien für die anderen Anwendungen in der Datenbank gespeichert und sind so jederzeit einfach erweiterbar. Die Kategorien können je nach Bereich unterschiedlich belegt sein. Auch müssen die Kategorien sortierbar sein, so daß nach Vorgabe eines Redakteurs die Reihenfolge der Darstellung der Kategorien bestimmbar ist.

Definition der Datenstruktur Es wird ein Primärschlüssel benötigt, eine Ordnungszahl, ein Titel der Kategorie und jeweils eine Checkbox für den Einsatzbereich der Kategorie. Da Kategorisierungen eine Aufgabe ist,

die in vielen Sites auftaucht, wurde zunächst eine generalisierte Datenbeschreibung für Kategorien im allgemeinen entworfen, diese kann in anderen Projekten wiederverwandt werden (die Klasse `CategoryDescription`) und dann davon die konkrete Datenbeschreibung für die IKS-Site abgeleitet (die Klasse `RobotCategoryDescription`). Die Administrationsseiten der generalisierten Kategorieerstellung konnten einfach in Kopie eingesetzt werden, da alle notwendigen Änderungen nur in der Datenbeschreibung vorlagen. Dies ist ein gutes Beispiel für die Erweiterbarkeit von existierenden Anwendungen im Framework.

Die notwendige Datenbeschreibung sieht mit ihren wichtigsten Elementen wie folgt aus:

```
class CategoryDescription extends WCK_Description {
    function CategoryDescription(){
        global $SCRIPT_NAME;

        WCK_Description::WCK_Description();

        $this->table='category';
        $this->defaultOrder=' ord asc ';
        $this->defaultRestriction=' status=0 ';
        // Persistente Elemente
        $this->field['id']= new WCK_ElementID('id',$this->table);
        $this->field['ord']= new WCK_ElementIntNumber('ord','Reihenfolge',0);
        $this->field['title']= new WCK_ElementText('title','Kategorie','',50,100);
        $this->field['status']= new WCK_ElementStatus('status','Status');
        // Hilfselemente
        $this->field['edit']= new WCK_ElementButton('id','bearbeiten...',
                                    $SCRIPT_NAME,'','job=edit&state=1');
        $this->field['moveup']= new WCK_ElementButtonPic('id',
                                    '/img/admin/arrowup.gif','10','10',
                                    'hoch',$SCRIPT_NAME,'','job=moveup');
        $this->field['movedown']= new WCK_ElementButtonPic('id',
                                    '/img/admin/arrowdown.gif','10','10',
                                    'runter',$SCRIPT_NAME,'','job=movedown');
        $this->field['movetop']= new WCK_ElementButtonPic('id',
                                    '/img/admin/arrowtop.gif','10','10',
                                    'Anfang',$SCRIPT_NAME,'','job=movetop');
        $this->field['movebottom']= new WCK_ElementButtonPic('id',
                                    '/img/admin/arrowbottom.gif','10','10',
                                    'Ende',$SCRIPT_NAME,'','job=movebottom');
    }
}

class RobotCategoryDescription extends CategoryDescription{
    function RobotCategoryDescription(){
        CategoryDescription::CategoryDescription();

        $this->field['art']= new WCK_ElementSelect('art','Bereich',1,false);
        $this->field['art']->translation=array(
                                     'link'=>'Link',
                                     'download'=>'Download',
```

```
                                        'project'=>'Projekt'
                                       );
        $this->field['ikscat']= new WCK_ElementCheckBox('ikscat','IKS');
        $this->field['legocat']= new WCK_ElementCheckBox('legocat','LegoLab');
        $this->field['pioneercat']= new WCK_ElementCheckBox('pioneercat','Pioneer');
        $this->field['krabatcat']= new WCK_ElementCheckBox('krabatcat','Krabat');
    }
}
```

Aussehen im Administrationsbereich

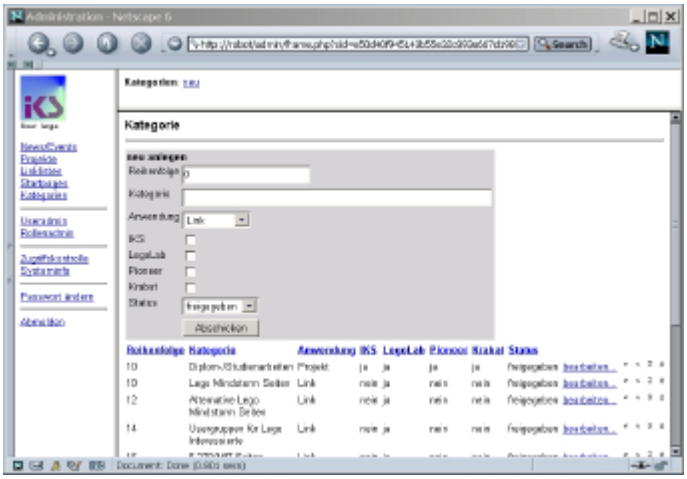

Abbildung 3.14: Administration der Kategorien

3.5.3 Rollenbestimmung

Aus den Anwendungen lassen sich für die Rechteverwaltung folgende Rollen extrahieren: Für die Bereiche IKS, LegoLab, Pioneer und Krabat muß es jeweils eine Rolle für die Redaktion der Nachrichten, der Projekte, der Linklisten und der Startseiten geben. Desweiteren gibt es noch die Rollen der Kategorienverwaltung, und der Userverwaltung. Aus dem System der standardisierten Nutzer- und Rechteverwaltung kommen noch der Rollen- und der Zugriffskontrollenadministrator hinzu (die letzten beiden Rollen sind aber nur für Programmierer der Site relevant, da mit ihnen die Rechteverwaltung an sich gesteuert wird).

Eine ausführliche Tabelle der Rollen und ihrer Rechte befindet sich im Anhang F.

3.5.4 Zusatz: unterschiedliche Designs

Zwecks Demonstration der Leistungsfähigkeit des eingesetzten Frameworks bei der Trennung von Inhalt, Logik und Design wurde die Site in sechs unterschiedlichen, frei umschaltbaren Designs erstellt.

Zur Auswahl der Designs wurde an den Fuß der Seiten eine Auswahlliste der verschiedenen Designs gesetzt.

In der technischen Realisierung wird das gewählte Design des Nutzers in den Sessionvariablen der Session abgespeichert. Auf diese Weise ist es nicht notwendig, die Designwahl in den URL's der Seiten zu kodieren. Zudem dient dieses Vorgehen als ein Beispiel für den Einsatz von Sessionvariablen. Der Quelltext ist im Anhang G zu finden.

Die Abbildung 3.15 zeigt die verschiedenen realisierten Designs (von links oben nach rechts unten: *pop1, pop2, pop3, classic1, classic2, classic3*). Die einzelnen Designs werden durch die Erstellung von nur sieben Programmdateien erstellt. Mit den sieben Dateien wird die gesamte Site in dem entsprechenden Design dargestellt.

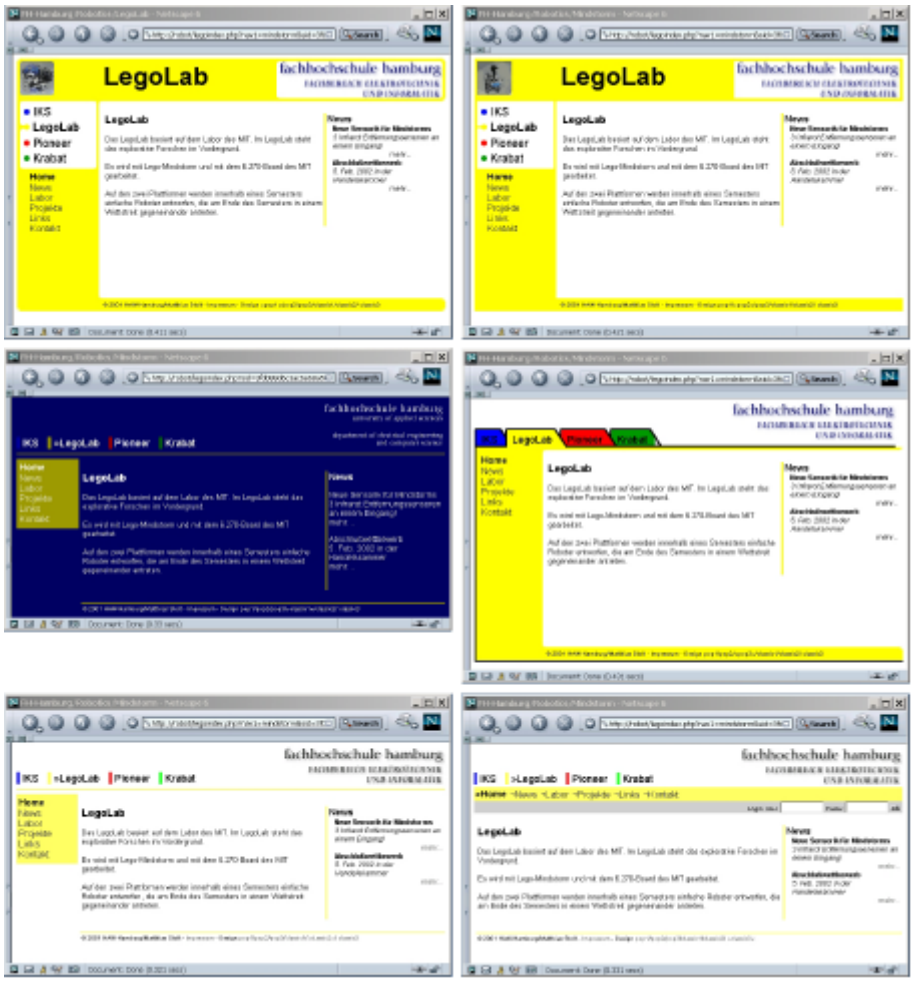

Abbildung 3.15: Verschiedene Designs der LegoLab-Startseite

Kapitel 4

Zusammenfassung und Ausblick

„It ain't over til it's over."
Yogi Berra

Das entwickelte Framework hat sich bei der Programmierung der alternativen Website des IKS-Projektes bewährt[1]. Mit seiner modularen Struktur ist das Framework an viele Problemstellungen anpassungsfähig und erweiterbar. Während der Erstellung der Arbeit wurden einige Themenstellungen aufgeworfen, die im Rahmen dieser Diplomarbeit nicht erschöpfend behandelt werden können. Aus diesem Grund folgen hier einige Hinweise auf zukünftige Erweiterungen und Themengebiete rund um das entwickelte Framework. Doch zunächst folgt eine Zusammenfassung der erreichten Ziele.

4.1 Erreichte Ziele

In den Abschnitten 1.8 und 2.2 wurden Anforderungen definiert, die durch das entwickelte Framework erfüllt werden. Hier nochmal die wichtigsten Forderungen im Überblick:

- Preisgünstige Lösung für individuelle Verwaltungssysteme von Websites
- Wiederverwendbarkeit von Anwendungen in anderen Projekten
- Gute Wartbarkeit der erstellten Anwendungen
- Einfache Erweiterbarkeit von wiederverwendeten Anwendungen in anderen Projekten

[1] Leider ist die Site zum Zeitpunkt der Fertigstellung dieser Diplomarbeit auf Grund von internen administrativen Schwierigkeiten an der Hochschule für Angewandte Wissenschaften noch nicht im realen Einsatz.

- Neudefinition von einfachen Anwendungen mittels Datenbeschreibungen
- Ausreichende Performance für mittelgroße Websites
- Browserbasierte Administration mit Zugangsberechtigungen
- Trennung von Funktionalität, Inhalt und Gestaltung
- Wahrung der Konsistenz des Layouts innerhalb einer Website
- Unabhängigkeit vom eingesetzten Datenbankmanagementsystem
- Funktionable User-, Rechte- und Sessionverwaltung

Im exemplarischen Einsatz des Frameworks im Kapitel 3 konnte die Praxisfähigkeit des Frameworks und an Hand des Benchmarks im Anhang A die Leistungsfähigkeit verifiziert werden.

4.2 Weiterer Ausbau des Frameworks

Als weitere Ausbaumöglichkeiten des Frameworks stehen vor allem

- die Entwicklung von Standardanwendungen,
- das Standardisieren weiterer Formularhandlingstypen
- das Entwicklung von neuen Eingabeelementen,
- die Erweiterung der Formatierungsmöglichkeiten für Tabellen und Listen und
- die Internationalisierung

als Betätigungsfelder bereit. Diese Punkte werden im Folgenden kurz weiter ausgeführt.

4.2.1 Entwicklung von Standardanwendungen

Um die in Abschnitt 3.3 angesprochene erhöhte Interaktivität mit den Nutzern zu erreichen, fehlen im bisherigen Ausbauzustand des Frameworks noch einige Anwendungen aus dem Communitybereich (Foren, Chats, Messenges, Newsletter, Abstimmungsmodule, u.ä.). Hier sind z.B. für die Anwendung *Foren* sicher auch neue Ausgabeobjekte und ein spezieller Datenbankleser zu entwickeln. Weitere gut wiederverwendbare Anwendungen wären eine generische Bilder-/Mediengalerie und die zugehörige Datei-Upload-Anwendung.

4.2.2 Standardisieren weiterer Formularhandlingstypen

Während der Programmierung der alternativen IKS-Site ist an einigen Stellen (z.B. der Zugriffskontrolle und den Kategorien) ein Formularhandlingstyp aufgetaucht, der es wert ist, standardisiert zu werden. Es geht hier um die Darstellung von Formular und Liste der mit dem Formular zu bearbeitenden Informationseinheiten auf einer Seite, wie in den Abbildungen 4.1 oder 3.14 gezeigt. Außerdem wird nach erfolgreichem Eintragen eines bearbeiteten Datensatzes zusätzlich zur Bestätigung gleich wieder das Formular für einen neuen Datensatz angezeigt.

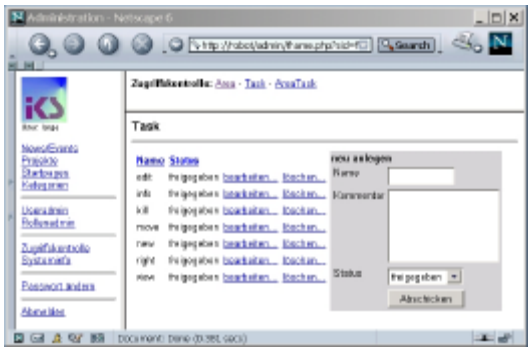

Abbildung 4.1: Neuer Formularhandlingstyp

4.2.3 Entwicklung von neuen Eingabeelementen

Eine wichtige Erweiterung des Klassenbaumes der Klasse `WCK_Element` wäre eine Texteingabemöglichkeit, in der durch ein semantisches Markup Besonderheiten des Textes kenntlich gemacht werden können.

Der Sinn eines solchen Textfeldes ist es, für Problematiken wie die der Einzelseiteneingabe (siehe Seite 94) eine saubere Trennung zwischen Layout und Inhalt zu erreichen.

In einer ersten Fassung eines solchen Markups sollten beispielsweise durch Leerzeilen getrennte Abschnitte als eigenständige Absätze formatiert werden, auch die Kennzeichnungen von Überschriften, Aufzählungen und ähnlichem wären sinnvoll. Als weitere Ausbaustufe sollte dann ein Markup von Querverweisen innerhalb der dargestellten Site hinzukommen, um so z.B. im IKS-Projekt eine Nachricht mit einem Projekt verknüpfen zu können. An dieser Stelle kann z.B. sehr gut über den Einsatz von `XML/XSLT`-Techniken nachgedacht werden.

Langfristig darf es nicht mehr möglich und notwendig sein, über die Eingabefelder des Frameworks echten `HTML`-Code einzugeben, denn dies ist ein Verstoß gegen die Forderung der Trennung von Layout und Inhalt.

4.2.4 Erweiterung der Formatierungsmöglichkeiten für Tabellen und Listen

Eine schöne Erweiterung der Formatierungsmöglichkeiten für Tabellen und Listen ist die Definition von Darstellungen für die letzten Elemente eines Datensatzes.

Was hiermit gemeint ist, sei an folgendem Beispiel erläutert. Gegeben sei eine Datenbeschreibung mit einem Titel, einem Lösch- und einem Edit-Button. Mit dieser Datenbeschreibung soll eine Tabelle ausgegeben werden. Hierbei soll die Spalte mit den Löschbuttons rot und die mit den EditButtons grün eingefärbt werden. Bei der bisherigen Formatierung werden die Spaltenfarben von links nach rechts festgelegt. Wird nun ein Feld (z.B. ein Status) zwischen Titel und Löschbutton eingefügt, dann verschiebt sich die Farbgebung der Spalten so, daß nun der Löschbutton grün dargestellt wird. Als Lösung für dieses Problem könnte man eine Formatierung so entwickeln, daß man zusätzlich zur normalen Farbgebung der Spalten von links nach rechts eine Farbgebung von rechts nach links definiert, die die vorhergehende überschreibt. Man definiert in dem Beispiel also die Farben der letzten beiden Spalten als rot und grün. Damit bleibt die Farbgebung der Buttons erhalten. Auch in der Nutzersicht ist solch eine Formatierungsmöglichkeit gut einzusetzen.

4.2.5 Internationalisierung

Zur Zeit ist das Framework auf den Einsatz in einsprachigen Sites ausgerichtet. Der weitaus größte Teil aller Website ist einsprachig. Der Wunsch nach einer Mehrsprachigkeit gerade von wissenschaftlichen Sites ist zwar groß, aber der organisatorische Aufwand deutlich größer, zusätzlich kommt der deutlich erhöhte redaktionelle Aufwand in der Pflege einer mehrsprachigen Site hinzu.

Das Framework an sich ist prinzipiell durchaus mehrsprachig gestaltbar. Die Methode der Wahl wäre der Einsatz des *GNU gettext* Paketes. Dieses Paket stellt eine NLS (Native Language Support) API zur Verfügung, die für die Internationalisierung von PHP Programmen eingesetzt werden kann. Das Vorgehen ist im groben das folgende: alle Textausgaben werden mit der Funktion `gettext()` gekapselt. `gettext()` sucht in einer mit dem Paket *GNU gettext* erstellten Datenbasis nach dem gekapselten String und liefert dann, wenn sie den String gefunden hat, eine Übersetzung oder wenn sie den String nicht gefunden hat, das Original zurück. Damit können alle einfachen Textausgaben des Frameworks internationalisiert werden.

Ein anderes Problem ist die Organisation und Übersetzung der Datenbankinhalte, hier müßten für die Informationseinheiten jeweils sprachabhängig unterschiedliche Daten vorhanden sein, z.B. als zusätzliche Felder in der gleichen Tabelle oder als eine Aufspaltung in Basisdaten, Verknüpfungsta-

belle und Übersetzungen.

4.3 Migrationsproblematik

Selten wird man ein Webprojekt auf einer „grünen Wiese“ aufbauen dürfen. Meist sind schon Webseiten oder Datenbestände vorhanden, deshalb ist es notwendig, Migrationspfade von verschiedenen Ausgangsbasen in das Framework zu bieten.

Durch die Offenheit des Frameworks stehen verschiedene Wege bereit, um eine Migration durchzuführen. Es kommt dabei immer darauf an, auf welchem Niveau dabei gestartet werden kann bzw. gestartet werden muß. In den Projekten des Abschnittes 1.7 wurde bei zwei Projekten eine Migration durchgeführt.

Immer, wenn erhöhte Ansprüche der Nutzer der Website — vor allem in Bezug auf Interaktivität — erfüllt werden sollen, ist allerdings eine vollständige Migration aller Inhalte der Site auf das Framework angezeigt, denn nur so kann die Propagation der Session-ID über alle Seiten der Site ohne den Einsatz von Cookies erfolgen (siehe Abschnitt 2.3.11).

Durch den Einsatz einer Datenbank als Datenhaltungsmechanismus ist es auch möglich, eine Migration der Daten aus dem Framework heraus durchzuführen. Hierzu können die Daten beispielsweise in XML-Dateien gespeichert werden, deren Schema aus den zugehörigen Metadatenbeschreibungen der Datensätze automatisch gebildet wird.

4.4 Skalierbarkeit

Ein weiterer Punkt, der im Framework Beachtung fand, ist die Möglichkeit der Skalierbarkeit der Lösung. Mit zunehmendem Datenverkehr und steigenden Nutzerzahlen bzw. Seitenabrufen sollte die Lösung mitwachsen können.

Ein erster Schritt ist die Auftrennung von Webserver und Datenbankserver auf zwei Maschinen. Eine weitergehende Lösung dieser Problematik ist bislang zwar nur theoretisch betrachtet worden, aber in der Konzeption des Frameworks schon berücksichtigt. Es ist der Einsatz eines Serverclusters von „einfachen“ Rechnern. Hierbei werden jeweils Paare von Datenbank- und Webservermaschinen über einen DNS-Roundrobbing reihum mit den Anfragen der Clients versorgt. Die Synchronisierung der Datenbanken erfolgt über den jeweils vorhandenen Replikationsmechanismus[2], hierbei werden von den Webservern schreibende Datenbankzugriffe ausschließlich an den Master geleitet, während die lesenden Zugriffe an den jeweiligen zugeordneten Datenbankrechner gehen.

[2] Bei MySQL eine Master/Multiple-Slave-Lösung.

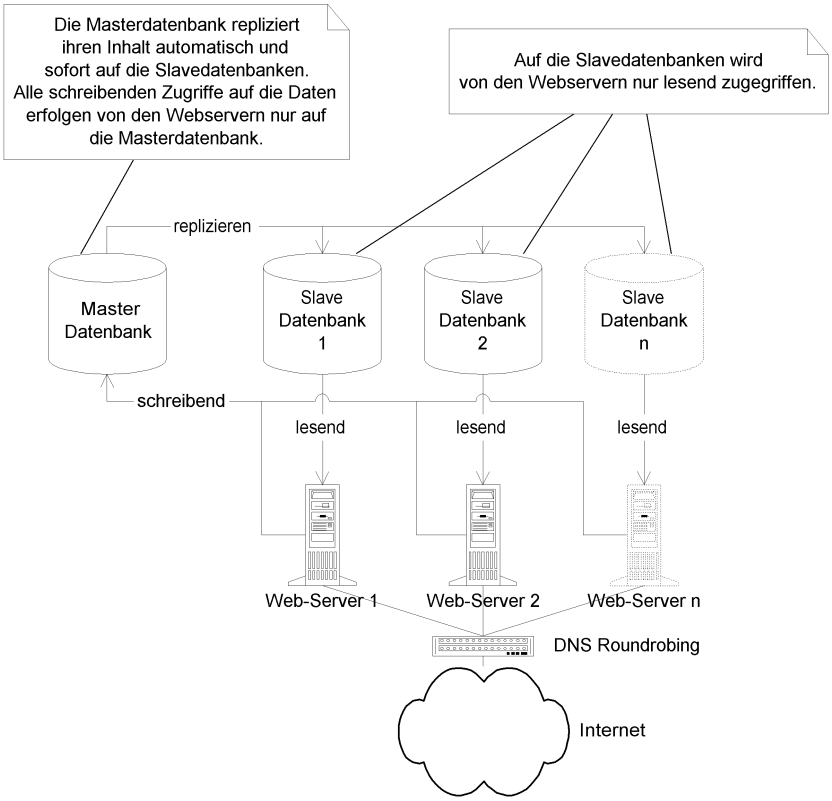

Abbildung 4.2: Skalierung des Webangebotes

Diese Lösung ist praktikabel, da das Verhältnis von lesenden zu schreibenden Zugriffen typischerweise sehr hoch ist und durch die Datenbank-Slaves entsprechende Rechenleistung zur Verfügung steht, ohne den Master mit der Verarbeitung der Schreibzugriffe zu überlasten.

Ein weiterer Vorteil dieser Lösung ist die verringerte Gefahr gegenüber Totalausfällen des Webangebotes. Es besteht zwar noch der Single-Point-Of-Failure des Masterdatenbankrechners, aber der Ausfall von Webserver- oder Slave-Rechnern hat nur noch eine Verringerung der Gesamtleistung des Clusters zur Folge. Der Betrieb kann aber von den verbliebenen Rechnern weitergeführt werden.

4.5 Integration

Häufig müssen Websites mit anderen in einem Unternehmen schon vorhandenen Systemen kooperieren bzw. mit in die Systeme integriert werden. Als Beispiele seien hier Legacy Systeme zur Zahlungsabwicklung, Fulfilment für

Shopsysteme, Warenwirtschaft und der Import von Daten sogenannter Content Syndication Provider genannt.

Die dem Framework zu Grunde liegende Technologie ist mit vielen Standardschnittstellen ausgestattet, so daß die für eine Integration notwendige Anpassung zum Beispiel auf WDDX, SOAP, EDI, XML, WebServices, XML-RPC oder auch schlichtem E-Mail-Austausch und FTP basieren kann. Für die Anbindung von Zahlungsabwicklungen stehen ebenfalls fertige Schnittstellen bereit [wwwProduct:PHP].

4.6 Andere Ausgabemedien

Da die zu Grunde liegende Technologie auch die Ausgabe von beispielsweise WAP/WML, SMIL, VRML und XML/XSLT unterstützt, kann auch das Framework prinzipiell diese Ausgaben unterstützen. Eigentlich können alle datenzentrierten und textorientierten Ausgabemedien, also Medien, in denen ein Datenpaket auf Anfrage erzeugt und zum Benutzer geschickt wird, generiert werden. Für WAP/WML existieren schon Lösungen, die auf der Basistechnologie beruhen [wwwCite:WML Tutorial].

Für einen Einsatz von WAP/WML im Framework müssen die Elemente des Klassenbaumes `WCK_Elements` und die Formatierungsobjekte `WCK_TableFormat`, `WCK_ListFormat` und `WCK_FormFormat` angepaßt werden.

4.7 Pflege unstrukturierter Daten

Zur Pflege unstrukturierter Daten bietet sich eine Lösung mit Hilfe eines modifizierten WikiWebs [Leuf, Cunningham 2001] an. Hierbei werden die Daten als Seiten des WikiWebs abgelegt und vom System für die Nutzer als nur lesbare Version publiziert.

In einer vertrauensvollen Umgebung (bzw. bei vertrauenswerten angemeldeten Nutzern) kann der WikiWeb-Ansatz auch für die Nutzer geöffnet werden. So wird ein Werkzeug zur Kollaboration unterhalb der Nutzer bereitgestellt.

Anhang A

Lasttest

„The unexamined website is not worth serving.“
Mike Pav (frei nach Sokrates)

Die Lasttests wurden unter Zuhilfenahme der Software *Socrates*[1] ermittelt. Hierfür wurde der folgende Versuchsaufbau genutzt.

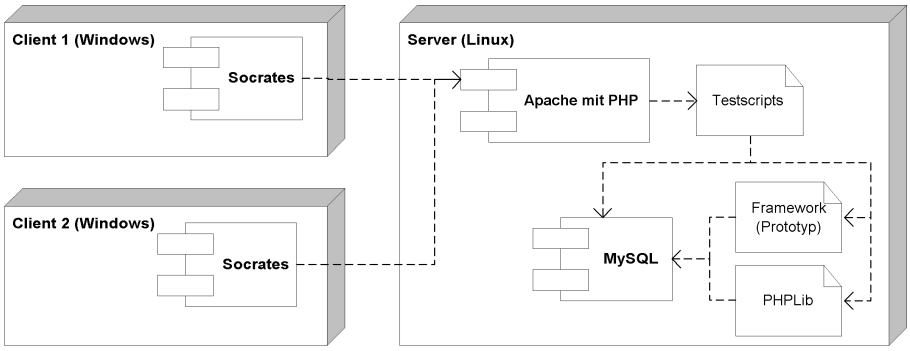

Abbildung A.1: Versuchsaufbau für Benchmark

Die zwei Clients (600 MHz/256 MB/Pentium III unter Windows) greifen via 10 MBit/s Ethernet auf den Server (266 MHz/128 MB/Pentium II unter Linux) zu.

Auf dem Server lief eine Standardinstallation von Suse 7.3 mit den mitgelieferten Komponenten Apache(1.3.19), PHP(4.0.4pl1) und MySQL(3.23.37). Die PHPLib lag in der Version 7.2d vor. Es wurde keinerlei Tuning vorgenommen.

Gemessen wurden verschiedene Teile des Frameworks. Dies waren im wesentlichen die Ausgaben von Tabellen und Listen, der Datenbankzugriff über die Datenbeschreibungen und die User-/Sessionverwaltung. Beim Lasttest wurde jeweils die Anzahl der ausgelieferten Seiten pro Sekunde (hier

[1] Socrates ist eine „post-card-ware“ von Mike Pav. Zu finden unter `http://www.jump.net/~bpav/socrates/` [wwwProduct:Socrates].

ist mehr besser) und die mittlere Latenzzeit in Sekunden (hier ist weniger besser) erfasst. Normalerweise wurde mit 10 simulierten Clients (5 von der Maschine *Client 1* und 5 von der Maschine *Client 2*) gleichzeitig auf die Testseiten zugegriffen, um den Server auch auszulasten.

Verglichen wurde die Ausgabe des reinen HTML-Codes bzw. direkt programmierter PHP-Seiten (die Geschwindigkeit des Apache ist hier bestimmend) mit jeweils einer Lösung unter Verwendung des Frameworks bzw. der PHPLib. Die benutzten Skripte liegen auf der CD bei (siehe Anhang D).

	hard coded		**PHPLib**		**Framework**	
Test	[pages/sec]	[sec]	[pages/sec]	[sec]	[pages/sec]	[sec]
reine HTML-Seite[1]	270	0,02	-		-	
Tabellenausgabe						
ohne Datenbank, einfach (5x5)[2]	118	0,05	7,6	1,1	24	0,35
ohne Datenbank, groß (20x100)[3]	7,2	0,14	0,6	12	0,6	12
ohne Datenbank, groß (20x100), nur 5 Clients[4]	7,5	0,06	0,66	5,42	0,67	5,7
ohne Datenbank, geschachtelt[5]	155	0,03	8,7	1,05	27,7	0,3
Listenausgabe						
ohne Datenbank, einfach[6]	143	0,03	7,6	1,16	39	0,2
mit Datenbank, einfach[7]	77,2	0,09	6,6	1,23	10,5	0,85
User-/Sessionverwaltung						
neue Session pro Zugriff[8]	52	0,15	9,5	0,93	28,5	0,29
wiederholter Zugriff[9]	150	0,03	9,5	0,92	26,5	0,32
gemischter Zugriff						
verschiedene Seitentypen[10]	45	0,13	5,5	1,5	6	1,5

Tabelle A.1: Benchmark zur Entscheidung: Templates vs. Programming

[1] Dieser Test ist nur als Referenz zum simplen Webserver da. Ausgeliefert wurde eine 1kb große Datei.
[2] Es wird eine einfache Tabelle mit 5 mal 5 Feldern erzeugt.

[3] Es wird eine einfache, aber große Tabelle mit 20 mal 100 Feldern erzeugt (Dateigröße 104 kb). Der Server wird mit diesem Test überlastet.
[4] Es wird die gleiche Tabelle wie bei [3] erzeugt, aber nur mit 5 Clients gleichzeitig zugegriffen.
[5] Hier wird eine Tabelle innerhalb einer anderen Tabelle erzeugt. Beide Tabellen sind recht klein (2x2 und 3x2), aber mit Formatierungen versehen.
[6] Ausgabe von 10 Datensätzen in Listenform. Die Daten sind im File fest kodiert.
[7] Wie [6], allerdings kommen nun die Daten aus der Datenbank.
[8] Bei jedem Zugriff wird eine neue Session gestartet.
[9] Es wird auf eine Session immer wieder zugegriffen.
[10] Dies ist eine Mischung aus gleichverteilten Aufrufen verschiedener Seitentypen. Das Mischungsverhältnis ist je dreimal die Seitentypen 2, 5, 6, 7, 8 und einmal Typ 3.

Ergebnis

Aus der Messung 10 ergibt sich für das Framework ein Wert von ca. 6 ausgelieferten Hits pro Sekunde bei maximaler Auslastung und gleichverteilten Zugriffen über alle Seitenarten. Damit käme man bei Volllast auf einen rechnerischen Wert von 15,5 Millionen Hits pro Monat. Dieser Wert ist allerdings utopisch, da der Zugriff auf die Seiten sicher nicht gleichverteilt rund um die Uhr stattfindet. Die folgende Grafik stellt einen verallgemeinerten Verlauf der Zugriffszahlen für einen Tag aus einem realen Webprojekt dar.

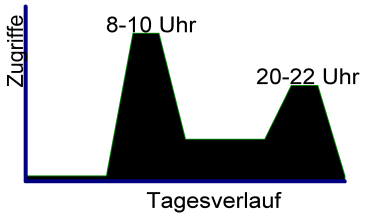

Abbildung A.2: Exemplarischer Zugriffsverlauf

Mit einem solchen Zugriffsverlauf käme man immer noch auf ca. 5 Millionen Hits pro Monat, die von diesem Server bewältigt werden könnten. In realen Projekten wird diese Zahl sicherlich abweichen, und man wäre gewiss auch an einer Reduzierung der Latenzzeit interessiert, aber der eingesetzte Server ist mit seinen 266 MHz/128 MB RAM keine Hochleistungsmaschine.

Lasttest gegen Seiten des IKS-Projektes

Gegen die Seiten des alternativen IKS-Projektes wurden nach der Fertigstellung ebenfalls Lasttests durchgeführt, um die Belastbarkeit einer mit

Test	**Ergebnis** [pages/sec]	[sec]
Startseite des IKS-Projektes	3,12	0,77
Startseite des LegoLab-Subprojektes	4,42	0,68
Newsseite des LegoLab-Subprojektes	4,80	1,36
Linkliste des LegoLab-Subprojektes	3,66	0,61
Projektseite des LegoLab-Subprojektes	4,81	1,40
Gemischter Zugriff auf alle obigen Seiten	4,09	0,97

Tabelle A.2: Benchmark der alternativen IKS-Site

dem Framework realisierten Site zu testen. Die Ergebnisse sind in der Tabelle A.2 aufgeführt.

Bei dem Zugriffsverlauf der Abbildung A.2 und den gemessenen 4 Seiten bei gemischtem Zugriff auf die alternativen IKS-Seiten werden immer noch ca. 3,3 Millionen Seiten pro Monat ausgeliefert. Damit kann das Framework als leistungsfähig angesehen werden.

Anhang B

Styleguide

„Um frei sein zu können, bedarf es Recht und Ordnung.“
Marcus Tullius Cicero

In diesem Styleguide werden die stilistischen Grundlagen für die Programmierung des Frameworks und seiner Anwendungen festgelegt. Die hier festgelegten Regeln sind für alle Programmteile verbindlich.

Der Sinn des Guides ist es, die Quelltexte für die Nutzer des Frameworks einheitlich und lesbar zu gestalten. Ebenfalls sollen die benutzten Konventionen des Frameworks dargelegt werden. Dies wird in besonderem Maße wichtig, wenn das Framework von Dritten erweitert wird.

Dieser Styleguide ist in verschiedene Teile gegliedert, in denen einzelne Aspekte festgelegt werden. Die Bereiche lauten:

- Namensgebung
- Kodierung
- Formatierung
- Dokumentation

B.1 Namensgebung

Die Namensregelung umfaßt die folgenden Elemente:

- Dateinamen und Dateinamenerweiterungen
- Variablen, Funktionen, Methoden, Klassen und Konstanten

Generell gilt für alle Namen:

- Es sind englische Namen zu benutzen. Deutsche Namen sind zu vermeiden.
- Namen sollen eine Bedeutung besitzen, d.h. sie sollen die Funktion charakterisieren. Dabei soll es sich stets um den Ausdruck handeln, der in der Fachwelt dafür gebraucht wird.

- Ähnliche Namen wie `$solutionPath` und `$solutionPaths` sind zu vermeiden.
- Besteht ein Name aus mehreren Worten, dann beginnt jedes Wort mit einem Großbuchstaben „studly caps“, z.B. `$rowCounts`. Unterstriche werden nicht zur Worttrennung benutzt (einzige Ausnahme hiervon bilden die Namen für Konstanten und die Abtrennung des Präfixes WCK_ bzw. wck_).

Das Präfix WCK_ steht für die Abkürzung von *Web Construction Kit.*

B.1.1 Dateinamen

Dateinamenerweiterungen

Aufgrund der Erweiterung eines Dateinamens soll sofort der Inhalt der Datei zu erkennen sein. Es gibt verschiedene Arten von Dateien, die jeweils ihre eigenen Erweiterungen haben und im Folgenden aufgelistet sind.

`html` Diese Dateien beinhalten reine HTML-Seiten, die via Angabe einer URL angezeigt werden.

`php` Diese Dateien beinhalten Seiten, die via Angabe einer URL angezeigt werden. Die Seiten enthalten PHP-Quelltexte.

`prt` Hierin sind Teile von Seiten enthalten. Diese Dateien können nicht direkt zur Ausgabe von Seiten benutzt werden.

`pod` Klassendefinitionen in PHP werden in diesen Dateien abgelegt.

`sql` Hierin werden die Definitionen von SQL-Tabellen und deren Dokumentation festgehalten. Diese Dokumentationsdateien befinden sich im Dokumentationszweig unter `/doc/database/`. Diese Dateien können direkt von MySQL als Eingabefiles gelesen werden.

`txt` Dies sind Dateien, in denen kurzzeitig Bemerkungen abgelegt werden. Es sind schlichte Textdateien. Sie dienen der Erläuterung oder Kommentierung der anderen Dateien im selben Verzeichnis.

Dateinamen

Für die Dateinamen selbst gelten folgende Regeln:

- Dateinamen sind komplett in Kleinbuchstaben zu halten.
- Alle Dateien, die zum Kern des Frameworks gehören, haben mit dem Präfix `wck_` zu beginnen.
- Klassen werden immer in einer eigenen Datei abgelegt, die den Namen der Klasse (in Kleinbuchstaben) und die Erweiterung `pod` trägt.
- Im Userbereich werden die Skripte einer Applikation mit einem gemeinsamen Präfix versehen. Als Beispiel sei eine Lexikonapplikation

genannt:

```
lexikonliste.php
lexikonbegriff.php
```

Auf diese Weise sind die zu der Applikation gehörigen Skripte leichter zu identifizieren, und in sortierten Verzeichnissen stehen die Dateien beieinander. Das benutzte Präfix wird auch für die Dateien im Includeverzeichnis genutzt. Die Administrationsskripte zu einer Applikation werden im Adminbereich in einem Unterverzeichnis mit dem Namen des Präfixes gesammelt.

```
admin/lexikon/liste.php
admin/lexikon/begriff.php
```

B.1.2 Benennung von Variablen, Methoden und Funktionen, Klassen, Konstanten

Variablen

Variablen beginnen immer mit einem Kleinbuchstaben, z.B. `$outputData`. Als Ausnahme von den sprechenden Namen sind für Laufvariablen in Schleifen die Kurznamen `$i, $j, $k, ...` erlaubt.

Alle globalen Variablen, die zum Kern des Frameworks gehören, tragen das Präfix `wck_`, z.B. `$wck_user`.

Methoden und Funktionen

Methoden und Funktionen führen Aktionen aus, daher sollte der Name ein Verb enthalten: `print()` oder `readData()`.

Geht das relevante Objekt, mit dem die Aktion durchgeführt wird, nicht aus der Parameterliste hervor, dann ist das Objekt in den Namen aufzunehmen: `countRows()`.

Besteht die Wirkung einer Methode oder Funktion nur im Lesen oder Setzen von Instanzvariablen, dann sollte der Name mit `get` bzw. `set` beginnen, wenn nicht durch einen anderen Namen eine bessere Lesbarkeit erreicht wird: `setColor()` und `getColor()`.

Klassen

Klassennamen beginnen immer mit einem Großbuchstaben, wie `MyPrinter`.

Alle Klassen, die zum Kern des Frameworks gehören, tragen das Präfix `WCK_`, z.B. `WCK_Printer`, `WCK_PrinterFormat`.

Konstanten

Da es in PHP keine echten Konstanten gibt, werden zur besseren Erkennung von Variablen, die als Konstanten betrachtet werden, die Namen komplett in Großbuchstaben geschrieben.

Die meisten Konstanten sollen ja als Platzhalter für bestimmte Einstellungen gelten. Wie z.B. die Anzahl der auf einer Seite auszugebenden Beiträge: `$NUM_OF_ARTICLES`. Diese Konstanten werden in einer Konfigurationsdatei angelegt, dort initialisiert und in den Skripten dann nicht mehr verändert.

B.2 Kodierung

Es gibt einige Regeln zur Kodierung, die für einfachere Lesbarkeit und sauberere/fehlerfreiere Programme sorgen.

- Blockkonstrukte
- Anweisungen
- ASP style Tag
- Datenbankabfragen

B.2.1 Blockkonstrukte

Nicht benutzt werden dürfen Blockkonstrukte wie `endif`, `endwhile`, da diese Konstrukte veraltet sind.

Jedes Blockkonstrukt hat *immer* einen Anweisungsblock in Klammern zur Folge. Dies gilt auch für einzelne Anweisungen, die ausgeführt werden. Falsch sind also die folgenden Beispiele, da diese Art der Programmierung sehr fehleranfällig ist:

```
// falsch
if($bedingung==1) echo "wahr"; else echo "falsch";

// falsch
for($i=0,$i<10,$i++)
    echo $i;
```

Korrekt ist:

```
// richtig
if($bedingung==1){
    echo "wahr";
}else{
    echo "falsch";
}
```

```
// richtig
for($i=0,$i<10,$i++){
    echo $i;
}
```

B.2.2 Anweisungen

Je Anweisung wird mit einem Semikolon abgeschlossen. Auch wenn dies zunächst nicht notwendig erscheint.
Beispiel (falsch):

```
<? echo $title ?> das ist falsch
```

Beispiel (richtig):

```
<? echo $title; ?> das ist richtig
```

Beispiel (noch besser):

```
<?
echo $title;
?> das ist richtig
```

B.2.3 Kennzeichnungen von PHP-Teilen

ASP-Tags Prinzipiell bestünde die Möglichkeit, in PHP sogenannte ASP-style-Tags einzusetzen.

```
<% echo ("You may optionally use ASP-style tags"); %>
<%= $variable; # This is a shortcut for "<%echo .." %>
```

Diese Möglichkeit von PHP darf nicht benutzt werden. Sie ist nur im Zusammenhang mit Portierungen sinnvoll, das Framework ist aber keine Portierung.

B.3 Formatierung

Die Formatierung soll die Struktur des Quelltextes hervorheben. *Wenn ein Quelltext umgeschrieben wird, so muß auch die Formatierung bereinigt werden.* Es werden folgende Aspekte der Formatierung besprochen[1]:

[1] Es ist durchaus bekannt, daß es für alle Varianten der Formatierung von Quelltexten gute Argumente existieren. Insofern ist die Wortwahl *richtig* oder *falsch* nur auf die Erfordernisse in diesem Projekt bezogen. Zur Wahrung der Konsistenz im Framework müssen schlicht die Regeln festgeschrieben werden.

- Leerzeilen
- Leerzeichen
- Einrückungen
- Klammern
- `switch`-Struktur

Leerzeilen

Leerzeilen sollen Blöcke von logisch zusammenhängenden Anweisungen trennen. Jeder Funktionsdeklaration soll eine Leerzeile vorausgehen, ebenso jedem PHPDoc-Abschnitt.

Leerzeichen

Der Einsatz von Leerzeichen ist zur Betonung von Gruppierungen vorgesehen, nicht zum Abtrennen von Funktionsnamen und deren Argumenten (denn diese gehören zusammen und sollen nicht getrennt werden!). Die Kontrollkonstrukte (`if, for, while`) werden in diesem Zusammenhang wie Funktionen betrachtet.
Beispiel (falsch):

```
if ( ( $bedingung == 1 ) || ( $bedingung > 10 ) ) {
    print ( "Halleluja, " . ( $bedingung + 1 ) . " Kekse" ) ;
    fun ( $arg1, "La" . "le" . "lu" ) ;
}
```

Beispiel (richtig):

```
if(($bedingung==1) || ($bedingung>10)){
    print("Halleluja, ".($bedingung+1)." Kekse");
    fun($arg1, "La"."le"."lu");
}
```

Bei Zuweisungen darf das Gleichheitszeichen nicht abgerückt werden, auf diese Weise kann im Skript leichter nach Zuweisungen zu einer Variablen gesucht werden.

```
$kekse='Lecker';
```

Einrückungen

Der Quelltext wird entsprechend der Blockstruktur eingerückt. Die Einrückung erfolgt mittels eines TAB pro Ebene. Leerzeichen sind zu vermeiden. Die Editoren dürfen keine Wandlung TAB $\leftrightarrow$ Leerzeichen durchführen und sind entsprechend einzustellen (Breite des TAB idealerweise vier Zeichen pro TAB).
Beispiel (falsch); diese Art der Programmierung ist sehr fehleranfällig:

```
if($bedingung==1){
echo "wahr";
}else{
echo "falsch";
for($i=0,$i<10,$i++){
echo $i;
}}
```

Beispiel (richtig):

```
if($bedingung==1){
    echo "wahr";
}else{
    echo "falsch";
    for($i=0,$i<10,$i++){
        echo $i;
    }
}
```

Klammern

Wie im obigen Beispiel zu sehen ist, werden Klammern direkt hinter das öffnende Konstrukt gesetzt. Die schließende Klammer befindet sich in gleicher Ebene wie das öffnende Konstrukt. Dies gilt für alle Kontrollkonstrukte (`if, else, elseif, switch, for, while, do..while, foreach`) und auch für die Definitionskonstrukte (`function, class`).
Die Teile `else` und `elseif` werden immer von den Klammern umgeben.

```
if($bedingung==1){
    // foo
}elseif($bedingung==2){
    // foobar
}else{
    // barboo
}
```

Nebenbei bemerkt: `elseif` wird nur sehr selten benötigt, meist ist das bessere Konstrukt ein `switch`.

switch-Konstrukt

Ein `switch` wird wie folgt formatiert.

```
switch($bedingung){
case 1:
```

```
        // foobar
        break;
    case 2:
        // foobarboo
        break;
    default:
        // barboo
    }
```

B.4 Dokumentation

Zur Dokumentation des Quellcodes des Frameworks wird das System *PHPDoc* eingesetzt, eine Adaption von Javadoc. *PHPDoc* ist in der PHP-Programmiergemeinde mittlerweile ein anerkannter Standard und wird ebenfalls zur Dokumentation der *PEAR*-Klassen benutzt, die ein Teil der PHP-Distribution sind. Unter `http://www.phpdoc.de` [wwwProduct:PHPDoc] ist eine veraltete Version hinterlegt. Neuere Versionen sind Bestandteile von *PEAR*.

Aus der Quelltextdokumentation im *PHPDoc*-Format wird automatisch die API-Dokumentation generiert. Das folgende Beispiel erhebt keinen Anspruch auf Vollständigkeit (oder sinnvolle Funktion):

```
<?
/**
* Datenbeschreibung
*
* Mit diesem Objekt werden im Framework alle Informationseinheiten definiert.
* Es regelt die Verwaltung von Elementen einer Informationseinheit und die
* Abhaengigkeiten zwischen den einzelnen Elementen.
* @link Element.html Siehe auch die Elementfunktionen
* @package Datenbeschreibung und Elemente
* @access public
*/
class WCK_Description {

    // ... lot of stuff deleted ...

    /**
    * Test der einzelnen Elementdaten
    *
    * Ueberprueft ob jedes sichtbare Element einen gueltigen Wert enthaelt
    * @access public
    * @return boolean
    *                  true - alle Elementwerte sind gueltig
    *                  false - mindestens ein Elementwert ist ungueltig
    */
    function verifyElements(){
```

```
        $ok=1;
        reset($this->order);
        $ok=$ok && $this->field[current($this->order)]->verify();
        while(next($this->order)){
            $ok=$ok && $this->field[current($this->order)]->verify();
        }
        return $ok;
    }
}
?>
```

Anhang C

Sichten auf eine Site

Eine Site kann aus verschiedenen Gesichtspunkten betrachtet und bearbeitet werden. Mit jedem dieser Gesichtspunkte sind verschiedene Fragen und Fragensteller verknüpft.

Fragen, die vorrangig vom Betreiber und seinem Informationsarchitekten bearbeitet werden:

- Wie lautet die Mission der Site?
  - Welche Ziele verfolgt der Betreiber?
  - Welche Ziele verfolgen die Nutzer?
- Welche Nutzer sollen von der Site angesprochen werden?

Fragen, die vorrangig vom Informationsarchitekten bearbeitet werden:

- Informationsarchitektur
  - In welcher Weise werden die Informationen in der Site präsentiert?
  - Welche Bezeichner werden für die Bereiche gewählt?
  - Wie werden die Bereiche miteinander verknüpft?
- Verteilung der Contentelemente
  - Welche Übersichtslisten werden wo platziert?
  - Welche Teaser werden eingesetzt?
  - Welche Kombinationen an Informationen werden bereitgestellt?
- Welche Applikationen werden benötigt?

Fragen, die vorrangig vom Gestalter bearbeitet werden:

- Welche Farben werden auf der Site eingesetzt?
- Wie werden einzelne Elemente gestaltet?
- Wie wird die Navigation gestaltet?
- Wie wird die Orientierung gestaltet?
- Wie werden Interaktionselemente gestaltet?
- Wie werden Strukturierungselemente gestaltet (Punkte, Linien, Hintergrundflächen)?

Fragen, die vorrangig vom Programmierer bearbeitet werden:

- Clientseitige Technologie
  - Welche Clients (Browser, PDA, etc.) kommen beim Zielnutzer zum Einsatz?
  - Welche Clienttechniken (HTML, Flash, dHTML, Javascript, Cookies, etc.) sollen eingesetzt werden?
- Logik der Applikationen
  - Wie laufen Geschäftsvorfälle ab?
  - Welche Rückfragen werden gestellt?
  - Welche Interaktionselemente werden benötigt?

Anhang D

Inhalt der CD

Auf der CD befinden sich die folgenden Pakete:

- WCK-Framework
  - Quelltext des Frameworks
  - Datenbankanweisungen für Nutzer-, Rechte- und Sessionverwaltung (MySQL)
- Projektwebsite IKS
  - Quelltext der Exemplarischen Anwendung
  - Datenbankanweisungen der Exemplarischen Anwendung (MySQL)
- PDF-Version dieser Diplomarbeit
- Software
  - PHP-Sourcen für Linux
  - MySQL-Sourcen für Linux
  - Apache-Sourcen für Linux
  - Windowsinstaller für PHP/MySQL/Apache
  - PHPMyAdmin
  - PHPDoc
- Daten und Programme für den Benchmark
  - PHPLib 7.2d
  - Socrates
  - Beispielseiten

Unter der Adresse `http://www.stolt.de/wck` [wwwCite:WCK] werden aktualisierte Versionen des Frameworks, der exemplarischen Anwendung und der Diplomarbeit selbst zum Download angeboten. Dort finden Sie auch Informationen über die Weiterentwicklung des Frameworks.

Anhang E

Internetadressen

Beispielseiten Die folgenden Sites sind Beispiele, die in Kapitel 1 zur Verdeutlichung verschiedener Typen von Sites herangezogen wurden.

`http://www.lucy-planning.de` [wwwExample:LucyPlanning]
`http://www.sport1.de` [wwwExample:Sport1]
`http://www.otto.de` [wwwExample:Otto]
`http://www.redseven.de` [wwwExample:RedSeven]
`http://www.superbad.com` [wwwExample:SuperBad]
`http://www.financialtimes.de` [wwwExample:FinancialTimes]

Hersteller bzw. Projekthauptseiten Nachfolgend die Links zu den Hauptseiten der eingesetzten Open Source Tools bzw. zu den Herstellern von kommerzieller Software, die Alternativen bzw. Werkzeuge bereitstellen.

PHP Hypertext Processor: `http://www.php.net` [wwwProduct:PHP]
Kommerzieller Support für PHP: `http://www.zend.com` [wwwProduct:Zend]

Datenbank
unter GPL (oder ähnlichen Lizenzen)
MySQL: `http://www.mysql.org` [wwwProduct:MySQL]
PostgreSQL: `http://www.postgresql.org` [wwwProduct:PostgreSQL]
kommerzielle Datenbanken
Oracle: `http://www.oracle.com` [wwwProduct:Oracle]

Editoren
Macromedia: Flash, Shockwave, Dreamweaver `http://www.macromedia.com` [wwwProduct:Macromedia] und
HomeSite (ehemals von Allaire `http://www.allaire.com` [wwwProduct:Allaire])
SoftQuad: HoTMetaL `http://www.softquad.com` [wwwProduct:Softquad]

Microsoft: Frontpage `http://www.microsoft.de` [wwwProduct:Microsoft]
NetObjects: Fusion `http://www.netobjects.de` [wwwProduct:NetObjects]
Adobe: GoLive `http://www.adobe.de` [wwwProduct:Adobe]

CMS/Applicationserver
CoreMedia: CoreMedia Publisher `http://www.coremedia.com` [wwwProduct:Coremedia]
Vignette: Vignette Content Management Server `http://www.vignette.com` [wwwProduct:Vignette]
Gaus Interprise: VIP ContentManager `http://www.gauss-interprise.de` [wwwProduct:Gauss]
IBM: WebSphere `http://www.ibm.com/software/webservers/appserv/` [wwwProduct:Websphere]
BEA: WebLogic `http://www.bea.com` [wwwProduct:Bea]
Macromedia: ColdFusion mit Spectra `http://www.allaire.com/products/spectra/` [wwwProduct:Allaire]

W3C/RFC/andere Nachfolgend einige Standards, die in dieser Arbeit erwähnt wurden bzw. die diese Arbeit beeinflußt haben.

RFC2616, Hypertext Transfer Protocol – HTTP/1.0
`http://www.w3.org/Protocols/rfc1945/rfc1945` [wwwCite:RFC1945]
RFC2616, Hypertext Transfer Protocol – HTTP/1.1
`http://www.w3.org/Protocols/rfc2616/rfc2616.txt` [wwwCite:RFC2616]
Session Identification URI
`http://www.w3.org/TR/WD-session-id.html` [wwwCite:W3C Session]
XML-RPC Remote Procedure Calls
`http://www.xml-rpc.com` [wwwCite:XML-RPC]

Anhang F

Rollen im IKS-Projekt

In der folgenden Tabelle werden alle Rollen des exemplarischen IKS-Projektes mit ihren zugehörigen Rechten aufgeführt (siehe auch Abschnitt 3.5.3). Die Tabelle spiegelt die Inhalte der beiden Datenbanktabellen `Group` und `GroupPerm` aus der Abbildung 2.11 des Abschnittes 2.3.11 zur Userverwaltung wieder.

Tabelle F.1: Rollen in der Nutzerverwaltung des IKS-Projektes

Rolle	**Bereich**	**Aufgabe**	**Spezial**
Rollen für Redakteure			
IKSStartPage	StartPage	editieren	IKS
LegoStartPage	StartPage	editieren	Lego
PioneerStartPage	StartPage	editieren	Pioneer
KrabatStartPage	StartPage	editieren	Krabat
IKSNachrichten	Nachrichten	neu anlegen	IKS
	Nachrichten	löschen	IKS
	Nachrichten	editieren	IKS
LegoNachrichten	Nachrichten	neu anlegen	Lego
	Nachrichten	löschen	Lego
	Nachrichten	editieren	Lego
PioneerNachrichten	Nachrichten	neu anlegen	Pioneer
	Nachrichten	löschen	Pioneer
	Nachrichten	editieren	Pioneer
KrabatNachrichten	Nachrichten	neu anlegen	Krabat
	Nachrichten	löschen	Krabat
	Nachrichten	editieren	Krabat
IKSProjekte	Projekte	neu anlegen	IKS
	Projekte	löschen	IKS
	Projekte	editieren	IKS
LegoProjekte	Projekte	neu anlegen	Lego
	Projekte	löschen	Lego
	Projekte	editieren	Lego
PioneerProjekte	Projekte	neu anlegen	Pioneer
	Projekte	löschen	Pioneer
	Projekte	editieren	Pioneer

wird fortgesetzt

Fortsetzung

Rolle	**Bereich**	**Aufgabe**	**Spezial**
KrabatProjekte	Projekte	neu anlegen	Krabat
	Projekte	löschen	Krabat
	Projekte	editieren	Krabat
IKSLinkListen	Links	neu anlegen	IKS
	Links	löschen	IKS
	Links	editieren	IKS
LegoLinkListen	Links	neu anlegen	Lego
	Links	löschen	Lego
	Links	editieren	Lego
PioneerLinkListen	Links	neu anlegen	Pioneer
	Links	löschen	Pioneer
	Links	editieren	Pioneer
KrabatLinkListen	Links	neu anlegen	Krabat
	Links	löschen	Krabat
	Links	editieren	Krabat
Kategorien	Kategorien	editieren	
Rollen für die Nutzerverwaltung			
Useradministrator	User	neu anlegen	
	User	löschen	
	User	editieren	
	User	Rollen zuordnen	
Rollen für die Systemverwaltung			
Rollenadministrator	Gruppen	neu anlegen	
	Gruppen	editieren	
	Gruppen	Rechte zuordnen	
Zugriffskontrolle	Zugriffskontrollen	neu anlegen	
	Zugriffskontrollen	löschen	
	Zugriffskontrollen	editieren	
Admininistrator	Systeminfo	ansehen	

Anhang G

Ausgewählte Quellcodes

In diesem Anhang werden einige ausgewählte Quellcodes wiedergegeben. Alle Quellcodes sind auf der beigelegten CD ebenfalls vorhanden. Die Wiedergabe hier erfolgt nur, damit es nicht notwendig ist, für einen ersten Eindruck über das Framework die Quellcodes am Rechner zu suchen und zu öffnen.

Skripting des Nachrichtenformulares

Das folgende Skript sorgt für die gesamte Abwicklung des Editierens und des Neuanlegens von Nachrichten. Es übernimmt sowohl die Darstellung des Formulars als auch das Auslesen der zu editierenden Daten aus der Datenbank bzw. das Belegen des Formulars mit den Standardwerten und zudem auch das Abspeichern der bearbeiteten Daten. Bei einer Änderung der zu Grunde liegenden Datenbeschreibung muß keinerlei Änderung erfolgen, damit die Änderungen auch im Formular und dem Handling der Nachricht in der Datenbank zum Tragen kommen.
Quelltext der Datei `/robot/html/admin/news/edit.php`:

```
<?
// Sessionmanagement und Kopf der Adminseite ausgeben
include('wck/admin/parts/config.prt');
include('wck/admin/parts/adminverify.prt');
include('wck/admin/parts/adminhead.prt');

// Benoetigte Teile des Framework laden
require_once('robot/descriptions/newsdescriptionedit.pod');
require_once('wck/core/wck_dbreader.pod');
require_once('wck/admin/formats/adminformformat.pod');
require_once('wck/core/wck_formprinter.pod');
require_once('wck/core/wck_form.pod');
require_once('wck/core/wck_formhandler.pod');

// Formularhandling vorbereiten
$description = new NewsDescriptionEdit();
$formFormat = new AdminFormFormat();
$formPrinter = new WCK_FormPrinter($formFormat);
```

```
$form = new WCK_Form($description,$formPrinter);
$formHandler = new WCK_FormHandler($form);

if(!isset($wck_formTask)){$wck_formTask='new';}

// Funktion zur Ausgabe einer Fehlermeldung bei Rechteverletzung
function notAllowed(){
    global $wck_generalFormat,$dontDoIt;
    echo $wck_generalFormat->font('Sorry, Ihre Rechte reichen'.
                                  ' fuer die gewuenschte Aktion nicht aus.');
    $dontDoIt=true;
}

// Rechte ueberpruefen
$dontDoIt=false;
if(!($special=='IKS' || $special=='Lego' ||
     $special=='Pioneer' || $special=='Krabat')){
    echo $wck_generalFormat->font('Sorry, der Bereich '.$special.' ist unbekannt.');
    $dontDoIt=true;
}else{
    switch($wck_formTask){
    case 'new';
        if(!$wck_user->permission('news','new',$special)){
            notAllowed();
        }else{
            echo $wck_generalFormat->font('<b>Neue Nachricht anlegen,'.
                                          ' Bereich: <i>'.$special.'</i></b><br>');
        }
        break;
    case 'edit';
        if(!$wck_user->permission('news','edit',$special)){
            notAllowed();
        }else{
            echo $wck_generalFormat->font('<b>Nachricht editieren,'.
                                          ' Bereich: <i>'.$special.'</i></b><br>');
        }
        break;
    default;
        notAllowed();
  }
}

// Das eigentliche Formularhandling initiieren
if(!$dontDoIt){
   $formHandler->doit();
}

// Fuss der Adminseite ausgeben
include('wck/admin/parts/adminfoot.prt');
?>
```

Darstellung einer Nachricht

Das folgende Skript sorgt für die Darstellung einer einzelnen Nachricht in der Nutzersicht. Hierbei werden alle Bereiche (IKS, LegoLab, Pioneer und Krabat) in allen Designs (siehe Abschnitt 3.5.4) von nur diesem einen Skript dargestellt. Quelltext der Datei `/robot/html/newsview.php`:

```
<?
$nav2='news';
if(!isset($nav1)){$nav1='iks';}

// Sessiondaten und Datenbankzugriff initialisieren
include('robot/parts/config.prt');
include_once('wck/core/wck_dbreader.pod');
include_once('robot/descriptions/newsdescriptionview.pod');

if(isset($wck_id)){
    // gewuenschte Meldung holen
    $description = new NewsDescriptionView();
    $description->defaultRestriction='id="'.$wck_id.'" and (status=0 or status=1) ';
    $db = new WCK_DBReader($description);
    $db->fetchGlobals();
    $data=$db->getArray();
    $singledata=$data[0];
    $special=$singledata[0];
    // anzuzeigenden Bereich auswerten
    switch($special){
    case 'Lego';
        $nav1='mindstorm';
        break;
    case 'IKS';
        $nav1='iks';
        break;
    case 'Pioneer';
        $nav1='pioneer';
        break;
    }
}else{
    // Aktuellste Meldung holen (im Falle des fehlerhaften Aufrufes des Skriptes)
    // anzuzeigenden Bereich auswerten
    switch($nav1){
    case 'mindstorm';
        $special='Lego';
        break;
    case 'iks';
        $special='IKS';
        break;
    case 'pioneer';
        $special='Pioneer';
        break;
    }
    $description = new NewsDescriptionView();
    $db = new WCK_DBReader($description);
```

```
    $db->fetchGlobals();
    $data=$db->getArray();
    $singledata=$data[0];
    $special=$singledata[0];
}

// Kopf und Navigation ausgeben
include('robot/designs/'.$design.'/header.prt');
include('robot/designs/'.$design.'/navigation.prt');

// Inhalt der Seite ausgeben
?>
<table width="100%">
<tr>
<td width="*" valign="top">
<?
// herausgesuchte Nachricht ausgeben
$wck_generalFormat->corpusHeadline($singledata[1]);
$wck_generalFormat->corpusIntro($singledata[2]);
$wck_generalFormat->corpus($singledata[3]);
?>
</td>
<?
$wck_generalFormat->verticalBar();
?>
<td width="200" valign="top">
<?
// Nachrichtenliste ausgeben
$wck_generalFormat->corpusHeadlineSmall('Weitere News');

require_once('robot/designs/'.$design.'/teaserlistformat.pod');
require_once('robot/designs/'.$design.'/listprinter.pod');
include_once('robot/descriptions/newsdescriptionteaser.pod');
// Daten aus der Datenbank holen
$description=new NewsDescriptionTeaser();
$wck_DBrows=$description->defaultRows;
$db = new WCK_DBReader($description);
$db->fetchGlobals();
$data=$db->getArray();

// Und die Daten darstellen
$teaserListFormat = new TeaserListFormat();
$tp = new ListPrinter($data,$teaserListFormat);
echo $tp->output();
?>
<img src="img/trans.gif" width="200" height="4" alt="" border="0">
</td>
</tr>
</table>

<?
// Fuss der Seite darstellen
include('robot/designs/'.$design.'/footer.prt');
?>
```

Ausgabe von Listen in Listen

Das folgende Skript gibt Linklisten aus. Hier ist das Besondere, daß innerhalb einer Liste weitere Listen ausgegeben werden. Quelltext der Datei `/robot/html/linkview.php`:

```
<?
$nav2='link';
if(!isset($nav1)){$nav1='iks';}
include('robot/parts/config.prt');

include('robot/designs/'.$design.'/header.prt');
include('robot/designs/'.$design.'/navigation.prt');

?>
<table width="100%">
<tr>
<td width="*" valign="top">
<?
switch($nav1){
case 'iks';
    $special='IKS';
    break;
case 'mindstorm';
    $special='Lego';
    break;
case 'krabat';
    $special='Krabat';
    break;
case 'pioneer';
    $special='Pioneer';
    break;
}

$wck_generalFormat->corpusHeadline('Linkliste');

include_once('wck/core/wck_dbreader.pod');
include_once('robot/descriptions/robotcategorydescriptionuse.pod');
include_once('robot/descriptions/linkdescriptionview.pod');

// Auslesen aller Kategorien
$description = new RobotCategoryDescriptionUse();
$db = new WCK_DBReader($description);
$db->fetchGlobals();
$data=$db->getArray();
$categories=array();
reset($data);
while(current($data)){
   $aktData=current($data);
   $categories[$aktData[0]]=$aktData[1];
   next($data);
}

$description = new LinkDescriptionView();
```

```
$db->setDescription($description);

require_once('robot/designs/'.$design.'/linkcategorylistformat.pod');
require_once('robot/designs/'.$design.'/linklistformat.pod');
require_once('robot/designs/'.$design.'/listprinter.pod');
$dummy=array(array());
$linkListFormat = new LinkListFormat();
$linkPrinter = new ListPrinter($data,$linkListFormat);

$pagedata=array();
$i=0;
reset ($categories);
while (list ($key, $val) = each ($categories)) {
    $wck_DBrestrict="cat='$key'";
    $db->fetchGlobals();
    $data=$db->getArray();
    if($data>0){
        $linkPrinter->setData($data);
        $pagedata[$i][0]=$val;
        $pagedata[$i][1]=$linkPrinter->output();
        $i++;
    }
}

$linkCategoryListFormat = new LinkCategoryListFormat();
$linkPrinter = new ListPrinter($pagedata,$linkCategoryListFormat);
echo $linkPrinter->output();
?>
</td>
</tr>
</table>
<?
include('robot/designs/'.$design.'/footer.prt');
?>
```

Umschaltung der Designs

Das folgende Skript sorgt für das Umschalten zwischen verschiedenen Designs (siehe Abschnitt 3.5.4), es wird auf allen Seiten des Projektes inkludiert. Quelltext der Datei `/robot/inc/parts/config.php`:

```
<?
include('robot/parts/dbconnect.prt');
include('robot/robotuser.pod');
// Userobjekt anlegen
if(!isset($sid)){$sid='';}
$wck_user=new RobotUser($sid);

// Der folgende Code speichert das zu verwendende Design in den Sessionvariablen
// ab, bzw. lie"st das gewaehlte Design aus den Sessionvariablen aus.
// Wird ein neues Design verlangt oder ist bisher kein Design gewaehlt worden?
if(isset($design) || !$wck_user->existsVar('Design')){
    // ja: Design umstellen
```

```
    if(!isset($design)){$design='pop1';} // Vorgabewert
    switch($design){  // nur bekannte Designs koennen gewaehlt werden
    case 'pop1';
    case 'pop2';
    case 'pop3';
    case 'classic1';
    case 'classic2';
    case 'classic3';
        break;
    default;
        $design='pop1';  // Fallback fuer unbekannte Designwahl
    }
    $wck_user->setVar('Design',$design);
    $wck_user->close();
}else{
    // nein: Design aus der Sessionvariablen lesen
    $design=$wck_user->getVar('Design');
}
// Initialisieren diverser Formatierungsfunktionen, je nach Design
require_once('robot/designs/'.$design.'/robotgeneralformat.pod');
$wck_generalFormat=new RobotGeneralFormat();
?>
```

Literaturverzeichnis

[Bauer 2001] Bauer, Herbert:
Unternehmensportale, Geschäftsmodelle, Design, Technologie.
1. Auflage, Bonn: Galileo Press GmbH, 2001

[Brockhaus 2000] Bibliographisches Institut + F. A. Brockhaus AG:
Der Brockhaus in Text und Bild.
Edition 2000/2001, Mannheim: Bibliographisches Institut + F. A. Brockhaus AG, 2000

[Büchner et. al. 2001] Büchner,Heino; Zschau, Oliver; Traub, Dennis; Zahradka, Rik:
Web Content Management, Websites professionell betreiben.
1. Auflage, Bonn: Galileo Press GmbH, 2001

[Cantor 1998] Cantor, Murray R.:
Object-Oriented Project Management with UML.
First Edition, New York: John Wiley & Sons, Inc., 1998

[Fowler 2000] Fowler, Martin:
Refactoring, Wie Sie das Design vorhandener Software verbessern.
1. Auflage, München: Addison-Wesley Verlag, 2000

[Grafinkel, Spafford 1997] Grafinkel, Simson; Spafford, Gene:
Web Security & Commerce.
First Edition, Cambridge: O'Reilly, 1997.

[Hunt, Thomas 2000] Hunt, Andrew; Thomas, David:
The Pragmatic Programmer.
First Edition, Reading, Massachusetts: Addison Wesley Longmann, Inc., 2000.

[Ibañez, Zee 1998] Ibañez, Ardith; Zee, Natalie:
HTML Artistry: More Than Code.
First Edition, Indianapolis, Indiana: Hayden Books, 1998.

[Kim 2001] KIM, AMY JO:
Community Building, Strategien für den Aufbau erfolgreicher Web-Communities.
1. Auflage, Bonn: Galileo Press GmbH, 2001, Übersetzung von [Kim 2000]

[Kim 2000] KIM, AMY JO:
Community Building on the Web, Secret Strategies for Successful Online Communities.
First Edition, Berkeley: Peachpit Press, 2000

[Leuf, Cunningham 2001] LEUF, BO; CUNNINGHAM, WARD:
The Wiki Way: Quick Collaboration on the Web.
First Edition, Boston: Addison-Wesley 2001.
Auch `http://www.wiki.org`

[Maslow 1954] MASLOW, ABRAHAM:
Motivation and Personality.
First Edition, New York: Harper, 1954.

[Rosenfeld, Morville 1998] ROSENFELD, LOUIS; MORVILLE, PETER:
Information Architecture for the World Wide Web.
First Edition, Cambridge: O'Reilly, 1998

[Siegel 1996] SIEGEL, DAVID:
Web Site Design, Creating Killer Web Sites.
First Edition, Indianapolis, Indiana: Hayden Books, 1996

[Pieper 2000] PIEPER, JAN:
Entwicklung eines internetfähigen Web Content Management Systems.
Hamburg, Fachhochschule Hamburg, Diplomarbeit, 2000

[Zimmerling, Werner 2001] ZIMMERLING, JÜRGEN; WERNER, ULRICH:
Schutz vor Rechtsproblemen im Internet. Handbuch für Unternehmen.
1. Auflage, Berlin: Springer-Verlag, 2001.

[Zschau 1999] ZSCHAU, OLIVER:
Web Content Management - Einführung, Konzepte, Software.
Mittweida, Fachhochschule Mittweida, Diplomarbeit, 2000

[EN-ISO 9241] NORM EN ISO 9241, Teile 1 und 10-17. 1996-99.
Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten.

Verzeichnis der zitierten Websites

[wwwCite:ContentManager] Feig & Partner:
Informationsportal zum Thema Content Management.
`http://www.contentmanager.de`; Stand: 18.1.2002

[wwwCite:IKS] Prof. Dr. v. Luck, Kai:
Startseite des IKS-Projektes der Hochschule für Angewandte Wissenschaften Hamburg.
`http://www.informatik.fh-hamburg.de/~robots/`; Stand: 18.1.2002

[wwwCite:IVW] IVW e.V.:
Informationsgemeinschaft zur Feststellung der Verbreitung von Werbeträgern e.V..
`http://www.ivw.de`; Stand: 18.1.2002

[wwwCite:RFC1945] Network Working Group:
RFC1945, Hypertext Transfer Protocol – HTTP/1.0.
`http://www.w3.org/Protocols/rfc1945/rfc1945`; Stand: 18.1.2002

[wwwCite:RFC2616] Network Working Group:
RFC2616, Hypertext Transfer Protocol – HTTP/1.1.
`http://www.w3.org/Protocols/rfc2616/rfc2616.txt`;
Stand: 18.1.2002

[wwwCite:WCK] Stolt, Matthias:
Web Construction Kit.
`http://www.stolt.de/wck`; Stand: 28. Januar 2002

[wwwCite:W3C Session] W3C:
Working Draft, Session Identification URI.
`http://www.w3.org/TR/WD-session-id.html`; Stand: 18.1.2002

[wwwCite:WML Tutorial] Robert Köpferl:
WML und PHP - Ein erster Schritt.
`http://www.php-center.de/artikel/php_und_wml.php3`; Stand: 21.1.2002

[wwwCite:XML-RPC] UserLand Software, Inc.:
Projektsite von XML-RPC.
`http://www.xmlrpc.org`; Stand: 18.1.2002

Verzeichnis der Beispielwebsites

[wwwExample:LucyPlanning] Lucy Planning GmbH:
Beispielsite für Archetyp: Imagesite.
`http://www.lucy-planning.de`; Stand: 18.1.2002

[wwwExample:FinancialTimes] Financial Times Deutschland GmbH & Co. KG:
Beispielsite für Archetyp: Informationsanbieter.
`http://www.financialtimes.de`; Stand: 18.1.2002

[wwwExample:Heise] Verlag Heinz Heise:
Beispielsite für Websiteelemente.
`http://www.heise.de`; Stand: 18.1.2002

[wwwExample:Otto] Otto Versand GmbH & Co:
Beispielsite für Archetyp: Shoppingsystem.
`http://www.otto.de`; Stand: 18.1.2002

[wwwExample:RedSeven] SevenOne Interactive GmbH:
Beispielsite für Archetyp: Kommunikationsplattform.
`http://www.redseven.de`; Stand: 18.1.2002

[wwwExample:Slashdot] Open Source Development Network, Inc.:
Beispielsite für Archetyp: Kommunikationsplattform.
`http://www.slashdot.com`; Stand: 18.1.2002

[wwwExample:Sport1] Sport1 AG:
Beispielsite für Archetyp: Informationsanbieter.
`http://www.sport1.de`; Stand: 18.1.2002

[wwwExample:SuJ] Springer und Jacobi:
Beispielsite für Archetyp: Imagesite.
`http://www.sj.com`; Stand: 18.1.2002

[wwwExample:SuperBad] BENJAMIN, BEN:
Beispielsite für reine Präsentation.
`http://www.superbad.com`; Stand: 18.1.2002

[wwwExample:Yahoo] YAHOO! DEUTSCHLAND:
Beispielsite für Archetyp: Kommunikationsplattform.
`http://www.yahoo.de`; Stand: 18.1.2002

Verzeichnis der Herstellerwebsites

[wwwProduct:Adobe] Adobe Systems Inc.:
Firmensite; Hersteller von GoLive.
`http://www.adobe.de`; Stand: 18.1.2002

[wwwProduct:Allaire] Allaire Corporation, jetzt Macromedia, Inc.:
Firmensite; Hersteller von HomeSite.
`http://www.allaire.com`; Stand: 15.11.2001

[wwwProduct:Apache] Apache Software Foundation:
Projectsite; Hersteller des Apache Webservers.
`http://www.apache.org`; Stand: 18.1.2002

[wwwProduct:Bea] BEA System, Inc.:
Firmensite; Hersteller von WebLogic.
`http://www.bea.com`; Stand: 18.1.2002

[wwwProduct:Coremedia] CoreMedia AG :
Firmensite; Hersteller von CoreMedia.
`http://www.coremedia.com`; Stand: 18.1.2002

[wwwProduct:Gauss] Gauss Interprise AG:
Firmensite; Hersteller von VIP.
`http://www.gauss-interprise.de`; Stand: 18.1.2002

[wwwProduct:HEXMAC] HEXMAC Software Systems AG:
Firmensite; Hersteller von HexBase Production System.
`http://www.icpro.de/`; Stand: 18.1.2002

[wwwProduct:Macromedia] Macromedia, Inc.:
Firmensite; Hersteller von Flash, Shockwave, Dreamweaver.
`http://www.macromedia.com`; Stand: 18.1.2002

[wwwProduct:Microsoft] Microsoft Corporation:
Firmensite; Hersteller von Frontpage.
`http://www.microsoft.de`; Stand: 18.1.2002

[wwwProduct:MySQL] MySQL AB:
Firmensite; Hersteller von MySQL
Beispielsite für Websiteelemente.
`http://www.mysql.com`; Stand: 18.1.2002

[wwwProduct:NetObjects] NetObjects, Inc.:
Firmensite; Hersteller von NetObjects Fusion.
`http://www.netobjects.de`; Stand: 18.1.2002

[wwwProduct:Oracle] Oracle Corporation:
Firmensite; Hersteller der Oracle Database.
`http://www.oracle.com`; Stand: 18.1.2002

[wwwProduct:PHP] The PHP Group:
Projektsite; Programmiersprache PHP.
`http://www.php.net`; Stand: 18.1.2002

[wwwProduct:PHPDoc] Ulf Wendel:
Projektsite; PHPDoc.
`http://www.phpdoc.de`; Stand: 18.1.2002

[wwwProduct:PHPLib] Erdmann, Boris; Köhntopp, Kristian; Schumann, Sascha:
Projektsite; PHPLib.
`http://phplib.sourceforge.net`; Stand: 18.1.2002

[wwwProduct:PostgreSQL] PostgreSQL Global Development Group:
Projektsite; Datenbank PostgreSQL.
`http://www.postgresql.org`; Stand: 18.1.2002

[wwwProduct:Socrates] Pav, Mike:
Projektsite; Tool zur Performancemessung von Websites.
`http://www.jump.net/~bpav/socrates/`; Stand: 18.1.2002

[wwwProduct:Softquad] SoftQuad Software, Ltd.:
Firmensite; Hersteller von HoTMetaL.
`http://www.softquad.com`; Stand: 18.1.2002

[wwwProduct:Vignette] Vignette Corporation:
Firmensite; Hersteller von Vignette Content Suite.
`http://www.vignette.com`; Stand: 18.1.2002

[wwwProduct:Websphere] IBM Corportion:
Firmensite; Hersteller von WebSphere.
`http://www.ibm.com/software/webservers/appserv/`;
Stand: 18.1.2002

[wwwProduct:Zend] Zend, Ltd.:
Firmensite; Hersteller von PHP-Tools.
`http://www.zend.com`; Stand: 18.1.2002

Danksagung

An dieser Stelle möchte ich mich bei allen an der Entstehung dieser Arbeit direkt oder indirekt beteiligten Personen bedanken. Insbesondere sind hier zu nennen: meine Eltern, die mich immer auf meinem Lebensweg bestärkt haben; Herr Steffen Lassahn, dessen Urlaub dazu führte, daß ich meine ersten Schritte im Umfeld des Webpublishings tat; Herr Wolfgang Büscher, der die Entstehung des Frameworks als Auftraggeber gefördert und mit seinen Ideen wertvolle Anregungen gegeben hat; Frau Barbara Krimpmann, die sich der Arbeit des Lektorates angenommen hat und nicht zuletzt Herrn Prof. Dr. Kai von Luck, dessen Betreuung bei der Erstellung dieser Diplomarbeit von unschätzbarem Wert war und ist.

Versicherung der Selbständigkeit

Hiermit versichere ich, daß ich die vorliegende Arbeit im Sinne der Prüfungsordnung *Technische Informatik PO 98* nach §24(5) ohne fremde Hilfe selbständig verfaßt und nur die angegebenen Hilfsmittel benutzt habe.

Matthias Stolt
Hamburg, den 28. Januar 2002