GPicS

Entwicklung eines XML basierten Fotoarchiv

Rico Scholz
IIAm10
Fachbereich Informatik
Hochschule Zittau / Görlitz

Stefan Radusch
IIAm10
Fachbereich Informatik
Hochschule Zittau / Görlitz

Martin Schicht
IIAm10
Fachbereich Informatik
Hochschule Zittau / Görlitz

Markus Ullrich
IIAm10
Fachbereich Informatik
Hochschule Zittau / Görlitz

Lehrveranstaltungsleiter:
Prof. Dr. rer. nat. Cristian Wagenknecht

21. Juni 2011

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Todo Liste

15, Links anpassen
16, Footnotes reparieren...

Diese Arbeit befasst sich mit der Erstellung eines XML basierten Fotoarchivs...

1 Einleitung

In dieser Belegarbeit wird die Entwicklung einer Webanwendung unter Verwendung von XML-Technologien beschrieben. Als Fallbeispiel wurde dafür die Erstellung eines Online-Fotoarchivs gewählt. Die konkrete Aufgabenstellung ist ab Seite 9 beschrieben. Nach der Analyse dieser, wurde ein bestehendes Content Management System[1] namens Drupal auf die Anforderungen hin untersucht. Als Ergebnis stand fest, dass es nicht genügend den Anforderungen entspricht und somit eine Neuentwicklung von Nöten ist. Im Kapitel Technologien auf Seite 12 ff, wurden die verwendeten Frameworks vorgestellt. Nach der Beschreibung der Entwicklungsumgebung, die aus dem Webserver Tomcat, der IDE IntelliJ und GoogleCode besteht, findet sich im Kapitel 6 der Systementwurf. Auf diesen aufbauend befindet sich im darauf folgenden Kapitel die Implementation. Diese erstreckt sich von der Datenbank, über die einzelnen Seitencontroller und der Beschreibung wie die Frameworks verwendet wurden, bis hin zur Erläuterung wie die Benutzeroberfläche erstellt wurde. Anschließend folgt das Kapitel Tests, welches die Vorgehensweise zum Testen der einzelnen Komponenten beschreibt. Daran fügt sich die Beschreibung, wie die Aufgabe unter den 4 Projektmitgliedern aufgeteilt wurde und mit welche Absicht. Abschließend wird eine Zusammenfassung über das Projekt gegeben und ein Fazit gezogen.

[1] kurz: CMS

2 Problem

2.1 Aufgabenstellung

2.2 Anforderungsanalyse

3 Evaluation eines bestehenden Systems - Drupal

Viele Menschen wollten schon immer ihre eigene Website, ihren Blog, ihr Forum oder Ähnliches haben, doch sind immer an den HTML- und Programmierkenntnissen gescheitert. Das ist seit dem Beginn von Content-Management-Systemen (CMS) vorbei. Solche Systeme ermÃ¶glichen es in den meisten FÃllen ohne Programmier- und HTML-Kenntnisse eine Website im Handumdrehen zu erstellen. Bei einigen Webspaces oder Online-Hostern sind solche Systeme schon bereits vorinstalliert und kÃ¶nnen schnell und einfach genutzt werden oder sie sind dafÃ¼r geeignet und man muss sein CMS nur dort hochladen. Zu den bekanntesten Content Management Systemen zur Zeit zÃhlen Typo3, Joomla und mittlerweile auch Drupal. Da Drupal immer stÃrker im kommen ist und einen immer grÃ¶Ãer werdenden Zuwachs findet, wird Drupal im nÃchsten Abschnitt mal etwas genauer beleuchtet werden.

Drupal ist ein CMS und Framework, welches ursprÃ¼nglich vom belgischen Informatiker Dries Buytaert entwickelt wurde. Es handelt sich dabei um eine freie Software, die unter der General Public License (GNU) steht.

Wer Drupal nutzen mÃ¶chte benÃ¶tigt dafÃ¼r lediglich einen Webserver, auf dem sich PHP und eine SQL-fÃhige Datenbank befindet. Das sind alles Dinge, die die meisten Webspaces kostenlos oder fÃ¼r einen kleinen Preis zur VerfÃ¼gung stellen. PHP ist eine Skriptsprache, die hauptsÃchlich zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird. Sie wird deshalb benÃ¶tigt, da Drupal in PHP programmiert wurde.

Der Aufbau von Drupal ist sehr einfach und lÃsst sich wie folgt beschreiben. Drupal besteht aus zwei Teilen, einem Core(Kern) und Modulen. Der Core beinhaltet die GrundfunktionalitÃt, welche mit weiteren Modulen erweitert werden kann. Zur GrundfunktionalitÃt zÃhlen Komponenten wie Template-Erstellung, Blogsystem, Benutzerverwaltung und Taxonomie. Diese Core-Funktionen reichen aus um eine simple Website zu erstellen. Man kann mit Hilfe der Template-Erstellung festlegen in welchen Bereichen der Website welcher Content angezeigt werden soll und auch die farbliche Gestaltung der gesamten Seite definieren. Die Benutzerverwaltung von Drupal ist sehr ausfÃ¼hrlich und detailliert. Sie zieht sich Ã¼ber mehrere Seiten und man kann dort Benutzergruppen definieren und ihnen Rechte zuweisen. Die Rechte kÃ¶nnen von Artikelarten die ein Benutzer schreiben darf, Ã¼ber die Bilderanzahl pro Artikel bis hin zu Kommentarfunktion und vielen mehr gehen. Mit hilfe von Taxonomie lassen sich MenÃ¼s erstellen und verschiedene Artikel

bzw. BeitrÃge der Website einer Kategorie zuweisen. Das kann Ã¼ber vorkommende Begriffe in BeitrÃgen oder Ã¼ber eine Auswahl der Kategorie beim Schreiben des Artikels passieren.

Der andere Teil neben dem Core sind die Module. Die sind dafÃ¼r da um je nach Anforderung Funktionen nachzurÃ¼sten. Diese Module werden meist von anderen Nutzer geschrieben und stehen dann allen zur VerfÃ¼gung und kÃ¶nnen per Download integriert werden. Was dabei zu beachten ist, ist die Menge der Module. Es gibt eine sehr groÃe Anzahl und man kann nicht immer gleich am Modulnamen erkennen, was dieses Modul kann bzw. welche Funktion es liefern soll. Zudem wurde in mehreren Test festgestellt, das viele Module noch nicht richtig Funktionieren und sich oft noch im Beta-Status befinden.

So muss man als Fazit sagen, dass Drupal eine echte Alternative in Sachen CMS ist zur schnellen und einfachen Erstellung einer Website oder eines WebBlogs, aber im Hinblick auf die Erstellung eines Foto-Archivs noch als eher ungeeignet darstellt. Da viele Funktionen die man in einem Foto-Archiv erwartet, wie eine Diashow oder eine AlbumÃ¼bersicht, Ãbersicht der Bilder nur schwer bis gar nicht realisieren lassen. Das liegt daran, dass dafÃ¼r weitere Module benÃ¶tigt werden, die zahlreich vorhanden sind, aber leider nicht zusammen arbeiten. So kann es sein das man nach langem Suchen ein passendes Modul finden aus der Vielzahl der Module, diese aber dann nicht mit den weiteren benÃ¶tigten Modulen funktionieren. Das kommt daher, das jeder der sich mit Programmierung auskennt, sein eigenes Modul schreiben und verÃ¶ffentlichen kann. Da aber jeder sein Modul so schreibt, dass es fÃ¼r seine Zwecke funktioniert, kommt es immer wieder zu Konflikten untereinander.

4 Analysierte Technologien

4.1 Metadatenextraktion

4.1.1 Kameraausgabeformate

Laut den Anforderungen, soll unser System in der Lage sein, aus den hochgeladenen Bildern, bereits wichtige Metadaten, wie die Zeit und den Ort, an dem das Foto entstanden ist, auszulesen. Dazu ist es zunächst notwendig, sich mit den einzelnen Formaten und deren Aufbau näher zu beschäftigen.

4.1.2 Exif

Für die Darstellung der Bilder in einer GoogleMaps-Karte sind natürlich die GPS-Koordinaten zwingend erforderlich. Wollte man früher GPS-Koordinaten mit einem Bild in Verbindung bringen musste man am Ort der Aufnahme zusätzlich noch ein GPS-Empfänger verfügbar haben und sich die Koordinaten notieren. Inzwischen gibt es aber Kameras, die die Koordinaten als Metadaten zu einem Bild mit abspeichern. Die in diesem Projekt verwendete Kamera speichert die Bilder im JPEG-Format. Bei JPEG gibt es die Möglichkeit, Metadaten mittels einem Exif-Datenblock abzuspeichern. Auf diese Weise kann man auf die GPS-Daten zugreifen.

Um nun mittels eines Programms die Daten zu extrahieren gibt es zwei Möglichkeiten. Die erste ist ein selbst entwickeltes Framework, mit denen man die Daten abgreift. Dabei muss eine ausführlich Analyse des Bild-Formats durchgeführt werden. Aufgrund der knappen Zeit und unter der Beachtung der Aufgabenstellung, dass ein Fotoarchiv und nicht ein Framework zum extrahieren von Exif-Daten entwickelt werden sollte, haben wir die zweite Möglichkeit gewählt. Dies war die Verwendung eines bestehenden Frameworks. Um solche Frameworks zu finden bietet sich immer eine Internetrecherche an. Diese ergab, das mittels des Frameworks „Metadata-Extractor“ die Anforderungen an ein solches Framework erfüllt wurden. Anhand des Quickstart-Guides ist zu erkennen, dass eine Verwendung obendrein relativ einfach ist.

Das Framework bietet vorgefertigte Methoden, um die Metaddaten auszulesen. Dazu werden die Daten in einem Metadata-Objekt abgespeichert. Mittels eines Iterators kann man anschließend durch sämtliche Metadaten-Tags iterieren.

```
Metadata metadata = JpegMetadataReader.readMetadata(file);
Iterator directories = metadata.getDirectoryIterator();
while (directories.hasNext()) {
```

```
Directory directory = (Directory) directories.next();
Iterator tags = directory.getTagIterator();
...
}
```

4.2 Webservices

4.3 KML

4.4 Yaml

In heutigen Webseiten verfolgt man oft die strikte Trennung von Gestalltung/Layout und Inhalt. Das Layout wird durch die Cascading Style Sheets definiert. Um dies schnell und einfach zu erledigen gibt es ein Framework namens YAML. YAML steht fÃ¼r "Yet Another Multicolumn Layoutünd ist ein (X)HTML/CSS Framework zur Erstellung von Layouts. Es liefert ein valides GrundgerÃ¼st aus validem XHTML- und CSS-Code, die eine hohe BrowserkompatibilitÃt bieten, d.h. eine browserÃ¼bergreifende korrekte Darstellung garantieren. So muss sich nicht der Programmierer oder Designer um die verschiedenen Browser mit ihren Eigenschaften und SchwÃchen beschÃftigen.

Dies ist mÃ¶glich, da durch YAML bereits viele Browser-Bugs abgefangen werden um die sich nicht speziell gekÃ¼mmert werden muss. Mit Hilfe von YAML ist eine fast vollstÃndige Trennung zwischen Layout und spÃteren Inhalt mÃ¶glich.

4.5 JSF

Schon immer wird in der Programmierung versucht bestehendes wiederzuverwenden. Dieser Ansatz findet sich auch in der Programmierung von Web-Applikationen. Für die Programmierung von Web-Applikationen ist so ein Framework JSF. Dieses Framework basiert auf Servlets und JSPs und bietet eine gute Möglichkeit komfortabel Webseiten zu entwickeln.

Natürlich gibt es noch weitere Frameworks, so dass die Frage im Raum steht warum in diesem Projekt JSF als Basis-Framework verwendet wird. Zum einem wäre hier zu nennen, dass JSF schon eine Weile auf dem Markt ist, was für die Programmierung einen großen Vorteil darstellt. Es geht nichts über eine gute Dokumentation über ein Framework. Außerdem gibt es im Internet fast immer eine Lösung für ein Problem, da fast immer jemand anderes vorher das Problem auch hatte. Durch diesen Vorteil konnte die Entwicklung beschleunigt werden.

Ein weiterer Grund für die Verwendung von JSF ist die Anforderung des Auftraggebers. Dieser hat vorgeschlagen das Projekt mit JSF zu verwirklichen. Da dies aus oben genannten Gründen ein guter Vorschlag war, wurde JSF verwendet.

In diesem Projekt wird die Version 2.0.1 von Mojarra verwendet. Sie bietet eine gute Unterstützung für Facelets und ist im Moment die aktuelle Version von JSF. Die Version von Mojarra bietet die gleichen Funktionalitäten wie die Referenzimplementierung von Apache MyFaces. Der Grund für die Verwendung von Mojarra ist darin begründet, dass die IDE bei der Entwicklung Mojarra als einzige Möglichkeit für JSF 2 vorgeschlagen hat. Da wie bereits erwähnt keine Unterschiede zu anderen Implementierungen bestehen, haben wir diesen Vorschlag angenommen. Ein weiterer Grund liegt in der Verwendung von Primefaces. Wie in Abschnitt 4.6 beschrieben, verwenden wir Primefaces 2.2.1, diese benötigt unbedingt JSF 2 mit Facelets.

4.6 Primefaces

In heutigen Webanwendungen werden häufig viele Technologien verwendet am eine ansprechende Benutzeroberfläche zu realisieren. Da gewisse Anforderungen immer wieder auftreten, wie zum Beispiel ein Dateiupload, ist es wünschenswert ein Framework zu verwenden. Wie in Anschnitt 4.5 erläutert wird, ist das Basis-Framework für dieses Projekt JSF. Mit diesem können zwar grundsätzlich alle Anforderungen erfüllt werden (u.a. Dateiupload), doch es ist immer eine Erleichterung, wenn es dafür schon vorgefertigte Komponenten gibt. Es wird daher Primefaces verwendet. Damit können alle Anforderungen schnell umgesetzt werden.

Außerdem soll natürlich die Benutzeroberfläche immer ansprechend gestaltet sein. Bei Primefaces sind die Komponenten schon mit einem hübschen Design vorhanden. Dieses vorgefertigte Design muss nicht unbedingt einen Nachteil bieten, und in diesem Projekt ist es auch kein Nachteil, da dadurch ein aufwendiger Designentwurf für diese Komponenten entfallen konnte. Es ist sogar möglich dieses Design ein wenig zu beeinflussen.

Wie jeder weiß gibt es natürlich eine ganze Reihe solcher Frameworks, die einen das Leben erleichtern. An dieser Stelle wären unter anderem RichFaces und IceFaces zu nennen. Aber unsere Wahl ist trotzdem auf Primefaces gefallen, weil es zum Einem eine Anforderung war Primefaces zu nutzen und zum Anderem bietet Primefaces nützliche Funktionen für dieses Projekt. So bietet Primefaces bereits eine Komponente für die Einbindung einer GoogleMaps-Karte. Mit dieser Komponente ist es außerdem möglich, Bilder in eine GoogleMaps-Karte anzuzeigen. Die Kernanforderung, das Anzeigen von Bildern in einer GoogleMaps-Karte, kann also mit Primefaces umgesetzt werden und erspart eine aufwendige Neuentwicklung.

Primefaces wird in diesem Projekt in der Version 2.2.1 verwendet. Dies ist die aktuelle stabile Version. Im Moment befindet sich die Version 3 in Entwicklung und es gibt auch schon Vorabversionen davon. Es wurde trotzdem die Version 2.2.1 verwendet, da einige Projektmitglieder bereits schlechte Erfahrungen mit Vorabversionen in anderen Projekten gemacht haben.

4.7 eXist

Ein zentraler Bestandteil unseres Fotoarchivs ist die Datenbank. Hier werden alle wichtigen Informationen zu Nutzern, Bildern und Alben gespeichert. Da wir XML-Dokumente verarbeiten, liegt es nahe, eine native XML-Datenbank zum Speichern der Daten zu verwenden. Dabei haben wir uns für eXist entschieden, was wir im Folgenden begründen wollen.

4.7.1 Warum kein relationales Datenbankmanagementsystem?

Relationale Datenbanken basieren auf einem gut durchdachtem Konzept, das sich jahrelang bewährt hat und bieten wichtige Vorzüge, die man bei einer nativen XML-Datenbank wie eXist nicht vorfindet:

1. Referentielle Integrität:

   In nativen XML-Datenbanken kann dieser Punkt leider nicht garantiert werden. Hier muss von Hand sichergestellt werden, dass alle Referenzen auf ein gelöschtes Objekt ebenfalls gelöscht werden. Da unser Datenmodell aber überschaubar ist, stellt dieser Punkt kein großes Problem dar.

2. Sequenzen zur Vergabe eindeutiger Primary Keys:

   Gerade bei einem Multinutzersystem ist dieser Punkt klar von Vorteil. Bei der Verwendung nativer XML-Datenbanken, muss dieses Problem in der Regel auf eine andere Art und Weise gelöst werden. Näheres dazu unter Abschnitt 7.1.

Links anpassen

Da relationale Datenbanken aber nicht primär für die Speicherung von XML-Dateien geeignet sind, müssen die, vom unserem XML-basierten Fotoarchiv, verwendeten Dokumente zunächst transformiert werden, um diese in der Datenbank zu speichern. Möglichkeiten dazu sind hier beschrieben: http://www.urz.uni-heidelberg.de/imperia/md/content/urz/programme/db_und_xml.pdf. Das kann aber schnell sehr ineffizient werden, wenn die Dateien immer größer werden oder es können nur Dateien mit einem bestimmten Aufbau verwendet werden.
Eine Möglichkeit wäre noch, die Dateien einfach als reine Zeichenketten in der Datenbank zu hinterlegen. Doch auch dieser Ansatz ist sehr ineffizient, da das Durchsuchen der Dateien nach bestimmten Informationen so viel zu lange dauert. Zwar gibt es auch relationale Datenbanken mit einer speziellen Erweiterung für XML-Dateien wie Oracle oder MS-SQL, doch sind diese in der Regel schlicht und einfach zu mächtig, also sehr groß und dadurch in der Anwendung vergleichsweise kompliziert.

4.7.2 Native XML-Datenbanken

Aus den oben genannten Gründen, haben wir uns also entschieden, eine native XML-Datenbank zu verwenden, die folgende Eigenschaften besitzt:

1. Geringe Größe:

Da wir auf unserem Testserver nur begrenzten Speicherplatz zur Verfügung haben und dieser größtenteils für die Bilder genutzt werden soll, darf die Datenbank nicht zu viel Platz belegen.

2. Schnelle Installation und Einrichtung:

   Die Datenbank stellt ein wichtiges Kernelement unseres Systems dar und sollte demnach so früh wie möglich einsatzfähig sein, um erste lauffähige Versionen unseres Fotoarchivs ausgiebig testen zu können.

3. Möglichst einfaches Auslesen und Verwalten der Daten:

   Auch dieser Punkt ist wichtig, damit die Datenbank so früh wie möglich einsatzfähig ist. Zudem muss es möglich sein, eventuelle Fehler schnell beheben zu können, wie durch das manuelle Löschen eines fehlerhaften Eintrags aus einer Datei.

4. Kostenfrei

Zudem sollte die Datenbank einige wichtige Eigenschaften relationaler Datenbanksysteme beherrschen, unter anderem:

1. Effiziente und strukturierte Speicherung:

   Viele native XML-Datenbanken, verwenden einen modellbasierten Ansatz zur Speicherung der Daten, was eine schnellere und effizientere Suche, als bei XML fähigen Datenbanken, die eine rein zeichenkettenbasierte Speicherung der Daten vornehmen, ermöglicht.

2. Indizes:

   Deren Verwendung bewirkt ebenfalls schnellere Suchvorgänge. eXist zum Beispiel, erzeugt bereits einige wichtige Indizes,welche in der Regel auch ausreichen um vor allem XPath und XQuery zu beschleunigen. Es können aber noch weitere Indizes in den Konfigurationsdateien zu jeder Collection vom Nutzer definiert werden.

3. Transaktionssicherheit:

   Die Datenbank sollte in der Lage sein nach einem Absturz, alle vollständig beendeten Transaktionen wiederherzustellen und alle nicht abgeschlossenen Transaktionen zurückzusetzen.

eXist erfüllt alle diese Vorraussetzungen und tatsächlich gibt es kaum weitere native XML-Datenbanksysteme, die als Alternative in Frage kämen (vgl. http://dbs.uni-leipzig.de/files/projekte/XML/paper/XMLDB_IAOforum.pdf). Wir haben diesbezüglich nur 3 weitere Systeme gefunden, welche im Folgenden kurz mit eXist verglichen werden sollen.

Footnotes reparieren...

Entwickler	Datenbank	Besonderheiten	Aktuelle Version
Wolfgang Meier	eXist[1]	Open Source, ständige Updates, große Community, beherrscht XQuery, gut dokumentiert	1.4 (23.12.2010)
Apache	Xindice[2]	kostenlos verfügbar unter der Apache License	1.2m1 (01.12.2007)
EMC	xDB[3]	beherrscht XQuery, gut dokumentiert, wird immer noch weiterentwickelt, benötigt Registrierung	10.1.0 (21.07.2010)
Software AG	Tamino[4]	beherrscht XQuery, gut dokumentiert, benötigt Registrierung, 3 Versionen (eine kostenfreie)	4.4.1 (unbekannt)

Dabei ist eXist das einzige System, das komplett kostenlos zur Verfügung steht und ständig weiterentwickelt wird. Zudem ist es durch die inzwischen sehr große Community sehr einfach, auftretende Probleme schnell zu klären ohne sich durch lange Dokumentationen kämpfen zu müssen. Insgesamt erscheint eXist von allen auch am übersichtlichsten. Somit ist es das System, welches am besten für unsere Anwendung geeignet ist.

5 Entwicklungsumgebung

5.1 Tomcat

5.2 IntelliJ

5.3 GoogleCode

Ein weiterer wichtiger Bestandteil der Entwicklungsumgebung ist ein System, dass automatisch allen Teammitgliedern den aktuellen Entwicklungsstand zugänglich macht. Anbieter von sogenannten Repositorys gibt es viele, so sind sowohl kostenlose als auch kostenpflichtige zu finden. Im Laufe des Studiums wurde bisher das System von ProjectLocker[1], sowie ein Subversion-Server der Hochschule genutzt. Da das Projekt 4 Studenten bearbeiteten, konnte ProjectLocker nicht verwendet werden, da dieses in der kostenlosen Variante nur 3 Benutzer erlaubt. Eine kostenpflichtige Nutzung hätte das Problem behoben, kam aber auf Grund von verfügbaren kostenlosen Alternativen nicht in Frage. Die Nutzung eines Hochschul-Servers wurde in der Gruppe besprochen, jedoch angesichts des Umzuges des Fachbereiches in ein anderes Gebäude und der daraus resultierenden Ausfallzeit abgelehnt.

Ein Alternative zu den genannten Varianten ist das GoogleCode-Projekt. Voraussetzung zur Eröffnung eines Projektes bei GoogleCode ist, dass man dieses unter einer OpenSource-Lizenz veröffentlicht. Anschließend kann man den vollen Funktionsumfang der Plattform nutzen, der Link zur Projektseite lautet http://code.google.com/p/geopicture-service/. Zu den Funktionen zählen neben dem SVN-Repository, ein Wiki, welches zum Publizieren der aktuellen Fortschritte bei der Programmierung, sowie zum Veröffentlichen der gehaltenen Zwischenstandspräsentationen in der Lehrveranstaltung diente. Außerdem ist ein Issue-Tracker vorhanden, dieser dient dazu, gefundene Fehler dem Entwickler direkt mitzuteilen. Dies ermöglicht eine schnelle und für andere nachvollziehbare Möglichkeit aufgetretene Probleme zu kommunizieren. Diese Art der Fehlerkorrektur wurde jedoch eher selten genutzt, da größtenteils die Entwickler sich in ein und dem selben Raum befanden. Eine direkte Kontaktierung ermöglichte eine schnellere und effektivere Behebung des Fehlers. Bei größeren Projekten und einer größeren räumlichen Distanz zwischen den Programmierern, ist die Nutzung des Dienstes jedoch sehr zu empfehlen.

Während der Bearbeitung des Projektes kam es zu keinerlei größeren Problemen mit dem Service. Lediglich das Einreichen von geänderten Code-Abschnitten konnte sehr

[1] URL: http://www.projectlocker.com/

selten nicht durchgeführt werden. Ein Grund dafür konnte nicht gefunden werden. Internetrecherchen ergaben nur, dass dies sehr selten und unregelmäßig auftreten kann. Nach einer kurzen Wartezeit von wenigen Minuten funktionierte es wieder und der Quellcode konnte erfolgreich in das Repository aufgenommen werden.

6 Systementwurf

6.1 UML-Diagramme

7 Implementation

7.1 Datenbank

7.2 Applikation

7.2.1 Controller

7.2.2 Scopes

7.2.3 Beanzugriff

7.2.4 MessageProperties

7.2.5 KML

7.2.6 File-Upload

Für das Hochladen der Bilder wird im Projekt die File-Upload Komponente von PrimeFaces verwendet. Dadurch konnte diese Anforderung relativ schnell umgesetzt werden. Zu Beginn traten allerdings einige Probleme auf. So benötigt diese Komponente eine Reihe von Biblotheken damit der Upload funktioniert. Dies wurde allerdings weder auf der PrimeFaces-Webseite noch in dem PDF erwähnt. Letztendlich musste eine Recherche im Internet gemacht werden, wo dann in einem Forum erwähnt wurde, welche Bibliotheken benötigt werden.

Ansonsten ist die Verwendung der Komponente ziemlich einfach. Es ist sogar möglich die Schaltflächen umzubenennen. Diese Möglichkeit wurde natürlich genutzt. Sobald man ein Bild hochlädt, wird ein FileUploadEvent ausgelöst. Mit dem entsprechenden Event-Objekt kann man auf die Daten zugreifen. Wir haben uns dazu entschieden das Bild im Dateisystem abzuspeichern und in der Datenbank lediglich den Pfad abzuspeichern. Der Grund dafür ist, dass sonst bei der Speicherung in der Datenbank die Bilder erst aufwendig encodiert werden müssten. Sobald dann später die Bilder aus der Datenbank geholt werden, müsste man außerdem wieder die Daten zu einem JPEG-Bild konvertieren. Diesen Mehraufwand konnten wir durch die Speicherung im Dateisystem umgehen. Außerdem gibt es mit unserem Verfahren weniger Probleme wenn mehrere Bilder hochgeladen werden.

Das Upload-Verzeichnis an sich liegt irgendwo im Dateisystem. Die einfachste Möglichkeit wäre zwar der web-Ordner der Applikation gewesen, aber dort wäre die Bilder bei

jedem Neudeploy gelöscht worden. Aus diesem Grund gibt es ein extra Verzeichnis wo alle Bilder abgespeichert werden.

7.2.7 EMail

Für die Benutzerkontoerstellung und das Zusenden eines neuen Passworts haben wir in unserem Projekt die Möglichkeit vorgesehen EMails zu versenden. Da der EMail-Versand zwingend ein EMail-Account erfordert, haben wir einen Mailserver eingerichtet, womit diese EMails verschickt werden konnten. Allerdings haben wir relative schnell Probleme festgestellt. So funktionierte das Schicken der Mails nur an die Hochschul-EMail-Adresse. Bei anderen Mail-Anbietern kamen keine EMails an. Sie wurden noch nicht einmal im Spam Ordner angezeigt. Der Grund hierfür liegt wahrscheinlich darin, dass die anderen Mail-Anbieter unsere Adresse nicht als vertrauenswürdig einstuften.

Dies war aber nicht das einzigste Problem. Probleme mit unserem VServer erzwangen ein Neustart des Mail-Servers. Danach hat das Senden von Mails nicht mehr funktioniert. Wo die Gründe liegen ist leider nicht ohne eine aufwendige Problemanalyse raus zu bekommen. In Anbetracht der knappen Zeit haben wir uns dafür entschieden, bei GMX ein Mailkonto einzurichten und über diese Adresse die Mails zu schicken.

Nun steht natürlich die Frage im Raum warum wir das nicht von Anfang an gemacht haben und uns mit dem Aufwand, dem Einrichten eines Mailservers, viel Arbeit aufgeladen haben. Der Grund dafür waren bedenken, einen „Sinnlos-Account“ einzurichten. Es werden ja nur einige wenige EMails versendet. Außerdem macht eine eigene EMail-Adresse immer einen besseren Eindruck. Wäre mehr Zeit gewesen hätten wir sicher das Problem mit unserem Mailserver lösen können, aber so mussten wir eben diese Alternative verwenden.

7.3 Validatoren

Oft haben Applikationen mit fehlerhaften Eingaben zu kÃmpfen, die der Nutzer nicht einmal bewusst tÃtigt. Um daraus bedingten Fehlfunktionen und einem reibunglosen Ablauf zu garantieren, gibt es die MÃ¶glichkeit sogenannte Validatoren zu benutzen. Sie sind dafÃ¼r da, eine Eingabe auf eine bestimmte Art von Muster bzw. Standard zu Ã¼berprÃ¼fen. Da das Ganze aber automatisch bei jeder Eingabe ablaufen soll, kann dabei nur auf syntaktische Korrektheit geprÃ¼ft werden.

Um einen Validator in JSF nutzen zu kÃ¶nnen muss er in der faces-config.xml bekannt gemacht werden. Das sieht wie folgt aus.

```
<validator>
    <validator-id>EmailValidator</validator-id>
    <validator-class>de.hszigr.gpics.validation.EmailValidator</
        validator-class>
</validator>
```

Listing 7.1: Einbinden eines Validators ind die faces-config.xml

Alle Validatoren, die genutzt werden sollen, mÃ¼ssen sich in dem Tag "validator"der faces-config befinden. Jeder Validator muss Ã¼ber eine eindeutige ID verfÃ¼gen und der Klassenname muss auch hinterlegt werden.
Das allein reicht noch nicht aus damit der Validator auch aktiv wird. An den Stellen, wo die Eingabe validiert werden soll, muss er noch der jeweiligen Komponente zugewiesen werden. Das geschieht so:

```
<h:inputText id="createUserMail" value="#{userController.email}"
    required="true" requiredMessage="#{msg.forgotEmail}">
  <f:validator validatorId="EmailValidator"/>
</h:inputText>
```

Listing 7.2: Zuweisung eines Validators einer Komponente

Wie hier zu sehen ist, wird der Email-Validator einem Input-Textfeld zugewiesen. Das "f:äm Anfang des Tags ist das alias fÃ¼r jsf-core.

7.4 Primefaces

7.5 Benutzeroberflï¿½che

Die BenutzeroberflÃache besteht aus drei Komponenten, HTML, YAML und Facelets. Als GrundgerÃ¼st nutzen wir eine HTML-Seite die folgende Struktur besitzt.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
        <meta http-equiv="Content-type" content="text/html; charset=
            utf-8"/>
        <title>GPicS</title>
        <link href="yaml/css/my_layout.css" rel="stylesheet" type="
            text/css"/>
</head>
<body>
</body>
</html>
```

Listing 7.3: HTML-GrundgerÃ¼st

Dieser kurze Auszug zeigt nicht dem kompletten Quellcode aber die wichtigsten Grundelemente. // Grundelemente etwas erklÃren // Zudem kommt bei uns die View-Handler-Technologie Facelets zum Einsatz, welche eine Alternative zum JavaServer Faces(JSF) Framework bildet. Mit ihr ist es mÃ¶glich eine Seite als Layout zu definieren und diese immer mit dem aktuell anzuzeigenden Content zu fÃ¼llen. Facelets verfÃ¼gt Ã¼ber

component-aliasing, was dafÃ¼r sorgt, dass normale HTML-Tags statt der Tags fÃ¼r UI-Komponenten genutzt werden kÃ¶nnen. Um eine Verbindung zu den jeweiligen UI-Komponenten herzustellen reicht es aus das alias-Attribut jsfc im Tag anzugeben. Hier ein kleines Beispiel:

```
<html    xmlns:ui="http://java.sun.com/jsf/facelets" xml:lang="en"
    lang="en">
<body>
<ui:insert name="content"/>
</body>
</html>
```

Listing 7.4: Facelets HTML-Tag

Wie im Beispiel zu sehen ist, wird im Body ein HTML-Tag mit dem Namen "contentëingefÃ¼gt. Dieses muss im weiter Verlauf nur immer mit dem jeweiligen Inhalt versorgt werden. Dies passiert wie folgt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns="http://www.w3.org/1999/xhtml" xmlns:jsp="http
    ://java.sun.com/jsf/composite" xml:lang="en"
                lang="en"
                template="layout.xhtml"
                xmlns:ui="http://java.sun.com/jsf/facelets"
                xmlns:f="http://java.sun.com/jsf/core"
                xmlns:h="http://java.sun.com/jsf/html"
                xmlns:p="http://primefaces.prime.com.tr/ui">

    <ui:define name="content">
        Hier wÃ¼rde jetzt der Content der jeweiligen Seite definiert
            werden
    </ui:define>

</ui:composition>
```

Listing 7.5: Facelets HTML-Tag

Man erstellt eine neue XHTML-Seite und legt auf dieser Seite im UI-Tag "composition"mittels der Zuweisung "template="layout.xhtml das Layout was genutzt werden soll fest. In diesem Fall die "layout.xhtml". Der eigentliche Inhalt fÃ¼r den UI-Tag "content"legt man mit Hilfe des Tags «ui:define name="content fest.
Das hat den Vorteil das man sich nicht auf jeder Seite damit beschÃftigen muss, wo soll der Tag spÃter angezeigt werden, sondern man definiert ihn nur und er wird immer an der Stelle, wo er im Template festgelegt ist angezeigt.

Wie nun das HTML-Layout gestaltet ist wird mittels CSS durch das YAML-Framework festgelegt, welches wir bis auf ein paar kleine Änderungen in Sachen Hintergrundgrafik

fast ungeÃndert nutzen. Es bietet durch mehrere CSS-Style-Sheets den Vorteil in allen gÃngigen Browsern gut dargestellt werden zu kÃ¶nnen. Es enthÃlt speziel fÃ¼r den Internet-Explorer angepasste Sytle-Sheets, da dieser mit den Ã¼blichen CSS-Befehlen nicht zurecht kommt bzw. diese immer etwas anders umsetzt. Damit ist zum Beispiel gemeint, das ein deutlicher Unterschied zwischen Firefox und Internet-Explorer besteht wenn man einen Abstand mit 10px definiert. Das kann zu deutlichen Unterschieden fÃ¼hren, welche aber durch YAML bereits abgefangen werden. Zudem kÃ¼mmert sich YAML auch um die weiteren browserspezifischen Bugs, was den Vorteil hat, dass sich der Programmierer nicht noch mit jeden Browser beschÃftigen muss, sondern eine LÃ¶sung fÃ¼r alle Browser implementieren kann.

8 Tests

9 Aufteilung des Projekts

10 Zusammenfassung

11 Fazit

Literaturverzeichnis

[Grü07] Toni Grütze., *Implementation, Evaluation und Optimierung von Prognosealgorithmen und Entscheidungsregeln für ein System zur Wartungsvorhersage*, Diplomarbeit, Hochschule Zittau/Görlitz, 2007.

[MS09] Tilo Müller and André Suttmann, *Beleg im Fach ADBC 2*, 2009, ATRR-Beleg.pdf.

[tH08] Prof. Dr. Ing-Klaus ten Hagen., *Rules to evaluate repair recommendations*, 2008, PdM - Rules 2 Eval Recommendations - 2008-10-10_1.ppt.

12 Anhang

12.1 CD-Inhalt

Auf der beiliegenden CD befinden sich folgende *Dateien* und **Ordner**:

- **Beleg**
  - *Beleg.pdf* - dieses Dokument im PDF-Format
  - **src** - enthält alle LaTeX-Quelldateien für die Erstellung des Belegs
    - **img** - enthält alle im Beleg eingebundenen Grafiken

12.2 UML-Diagramme

Eidesstattliche Erklärung

Hiermit erklären wir an Eides Statt, dass wir den Beleg selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Görlitz, 21. Juni 2011