Universität Hamburg
Fachbereich Informatik

"Objektorientierte Softwareentwicklung"

Softwarearchitektur im JaZe

Baccalaureatsarbeit

zur Erlangung des Grades

B.Sc.

vorgelegt von

Savas Cetin
Matr.Nr: 520 27 99
StPo ‘98

31. August 2002

Betreuer: Dr. Guido Gryczan

Inhaltsverzeichnis

1. Vorbemerkung

Dieses Dokument und das zugehörige Programm entstanden im laufe des OOSE Projektes des Fachbereichs Informatik der Universität Hamburg im Sommersemester 2002. Die Leitung des Projektes hatte Dr. Guido Gryczan unterstützt durch Henning Wolf.

Die TeilnehmerInnen sollten in diesem Projekt die Prinzipien der objektorientierten Entwicklung interaktiver Anwendungssoftware nach dem Werkzeug & Material-Ansatz kennenlernen.
Die Lehrveranstaltung sollte den Teilnehmer/Innen eine Einführung in die wesentliche Konzepte der objektorientierten Entwicklung interaktiver Systeme. Anbieten. Dabei standen die Konzepte und Techniken des Werkzeug & Material-Ansatzes **[Zullighoven98]** im Vordergrund. Dazu kamen Themen wie Design Patterns (Entwurfsmuster)**[Gamma et. Al 96]** sowie bewährte Notationen der Unified Modelling Language (UML).

Ziel dieses Projektes war innerhalb von drei Gruppen à 10 Personen ein Zeiterfassungssystem in Java zu implementieren. Im Projekt sollten Dinge wie Teamarbeit und der Ansatz des „Xtreme-Programming“ stehen. Die Teamleitung der Gruppe Eins übernahm hierbei Fabian Dittberger, der sowohl als Ansprechpartner für die Leiter, als auch für die Teammitglieder war. In dieser Baccalaureatsarbeit wird versucht dem Leser die Architektur des Java-Zeiterfassungssystems nahe zu bringen und die zu Grunde liegenden Ideen zu verdeutlichen.

Hierzu wurden alle Dokument, die in der Entwurfs-, Implementations- und Dokumentationsphase entstandenen Dokumente noch hinzugefügt. In dieser Baccalaureatsarbeit werden neben ausführlichen Beispielen, noch die Möglichkeit gegeben, diese Anwendung unter http://swt2g.zoom:8080/jaze zu nutzen.

1.1 Leserkreis

Diese Baccalaureatsarbeit richtet sich in erster Linie an die nachfolgende Teilnehmer des Projektes im Wintersemester/Sommersemester 02/03, die mit der hier zu Grunde liegenden Anwendung weiterarbeiten sollen. Dieses Dokument soll als eine Hilfestellung betrachtet werden, um einen leichteren Einstieg in unsere Anwendung bzw. unser Ergebnis zu bekommen.

In zweiter Linie richtet sie sich an Interessenten des aktuellen Projektes. Dazu gehören die Projektteilnehmer der anderen Gruppen, die einen Einblick in unsere Anwendung verschaffen wollen, aber auch an einer Alternativlösung interessiert sind. Zu guter Letzt betrifft es auch unsere Gruppe, die eine Zusammenfassung und Ausarbeitung der eigenen Lösung haben wollen.

1.2 Gegenstandsbereich

Bevor ich anfange über die Funktionalitäten des „neuen" JaZe zu berichten, sollte ein Einblick über das „alte" Zeiterfassungssystem ZEIS geschaffen werden. Der Vergleich beider Lösungen sollte die Unterschiede aufdecken.

Das **ZEIS** (**ZEI**terfassungs**S**ystem) ist eine auf Java basierte Anwendung mit dem der Mitarbeiter einer Firma die Arbeitszeiten eintragen und andere Funktionalitäten wie z.B. einen Kalender nutzen kann.
Der *Mitarbeiter* trägt je nach belieben seine Arbeitszeiten bis spätestens Ende des Monats ins ZEIS ein. Dazu meldet sich der Mitarbeiter beim System mit seinem Nachnamen und einem Passwort an. In ZEIS arbeitet er nun die Tage nacheinander durch, für die er Zeiten in die Zeiterfassung eintragen möchte. Er kann dafür einzelne Tage eines Monats einsehen und bearbeiten.
Der Eintrag erfolgt dann für den jeweiligen Tag unter Angabe der Anfangs- und Endzeit, der Auswahl eines Projektes sowie einer Aktivität innerhalb dieses Projektes, zu der die eingetragene Zeit erarbeitet wurde. Eine zusätzliche Bemerkung bzw. Notiz ist möglich.

Sind die Zeiten vollständig erfasst, wählt der Mitarbeiter den Druck eines Monatsberichts mit den jeweiligen Projekten und bestimmt Jahr und Monat, für den er seine Arbeitszeiten drucken will.

Der *Administrator* kann darüber hinaus Projekte im System anzeigen, abändern, entfernen sowie neue erstellen. Diese Projekte kann er in einer Projektübersicht einsehen und verwalten:

Zum Anlegen erstellt er ein neues Projekt und gibt diesem einen Namen, worauf es in der Projektübersicht erscheint.
Diesem Projekt fügt er dann Aktivitäten hinzu. Diese werden in der Projektübersicht neu erstellt und nach Vergabe eines Namens dem Projekt zugeordnet. Genauso kann er aus einem Projekt nicht mehr benötigte Aktivitäten entfernen oder umbenennen.

Außerdem hat der *Administrator* die Möglichkeit, die im System erfassten Mitarbeiter anzuzeigen, abzuändern sowie Mitarbeiter wieder aus der Zeiterfassung zu entfernen oder neue anzulegen. Die Mitarbeiter, die ZEIS benutzen sollen, kann er in einer Mitarbeiterübersicht einsehen und verwalten:

Zum Anlegen erstellt er einen neuen Mitarbeiter und gibt dessen Namen und ein vorläufiges Passwort ein, worauf dieser in der Übersicht erscheint.. Ferner kann der Administrator die Sollarbeitszeit mit einstellen, damit das System auf deren Basis eine Soll/Ist-Stand der geleisteten Stunden in einem Monat errechnen kann. Passwort sowie Sollarbeitszeit können von entsprechenden Mitarbeiter geändert werden

Zum Erstellen eines Reports (Stundenbericht, Übersichtsreport [1] , Gesamtübersicht [2]) wählt der Administrator ein Projekt aus, welches er mit dem gewünschten Zeitraum ausdrucken möchte, sowie den gewünschten Report aus einer Übersicht über alle im System verfügbaren Berichte.

Ein Report enthält dabei immer dieselbe Form der Auswertung und Darstellung der Einträge aus ZEIS. Der Administrator hat keine Möglichkeit, die einzelnen Berichtsarten für spezielle Ausdrucke anzupassen bzw. bestimmte Informationen auf einem Report aus- oder einzublenden.

Ein weiteres Problem welches bei der Benutzung von ZEIS auftritt, ist, dass es sehr stark in einen bestimmten Arbeitsplatzkontext eingebunden ist.
So müssen bestimmte Netzverbindungen auf einem Arbeitsplatz, an dem mit ZEIS gearbeitet werden soll, vom Benutzer zur Verfügung gestellt werden, die über einen einfachen Zugriff auf ein firmeninternes Netzwerk hinaus gehen. Außerdem gibt es viele Mitarbeiter die nur „auswärts" arbeiten, welche aber keine Möglichkeit haben, sich zum Beispiel über das Internet an ZEIS anzumelden, um von außerhalb ihre Zeiten in das System einzupflegen.

Die automatische Verwaltung der Zeiten im Kontext von gesetzlichen Arbeitszeitbestimmungen (Pausen), Urlaubsanspruch, Überstunden, Krankheitsausfall etc. wird vom ZEIS nur unzureichend bzw. gar nicht übernommen.

---

[1] Übersichtsreports, die die in einem Projekt geleisteten Zeiten zusammenfassen.

[2] Gesamtübersichten über Projekte und den im Projekt beschäftigten Mitarbeiter mit deren Zeiten.

1.3 Zielsetzung

JAZE ist eine **JA**va-**ZE**iterfassungssoftware, die im Rahmen des Projektes OOSE entwickelt wurde. Sie soll die jetzige Zeiterfassung „ZEIS" ablösen. Die Software bietet neben dem Zeiteintrag auch die Möglichkeit Benutzer zu verwalten, Berichte zu erstellen, Projekte zu verwalten und sich einen guten Überblick mittels Kalender zu verschaffen.

Neben der Wahl lokal in der Firma mit dem Tool zu arbeiten, wird auch ein entfernter Zugriff übers Internet (http://swt2g.zoom:8080/jaze) ermöglicht. Das heißt, wenn jemand seine Arbeitszeiten eintragen möchte, kann er diesen Eintrag auch bequem übers Internet durchführen, ohne jedes mal in der Firma präsent zu sein.
Ein weiteres Problem stellen die monatlichen Berichte dar, die Aufschluss über ihrer gearbeiteten Stunden geben. Diese werden nämlich jeden Monat in

zweifacher Ausführung unterschrieben an die Abrechnung abgegeben und dienen dort der Buchhaltung und Rechnungsstelle. Dieser Aufwand wird in der neuen Zeiterfassung durch den Versand der Berichte per e-Mail aufgehoben.

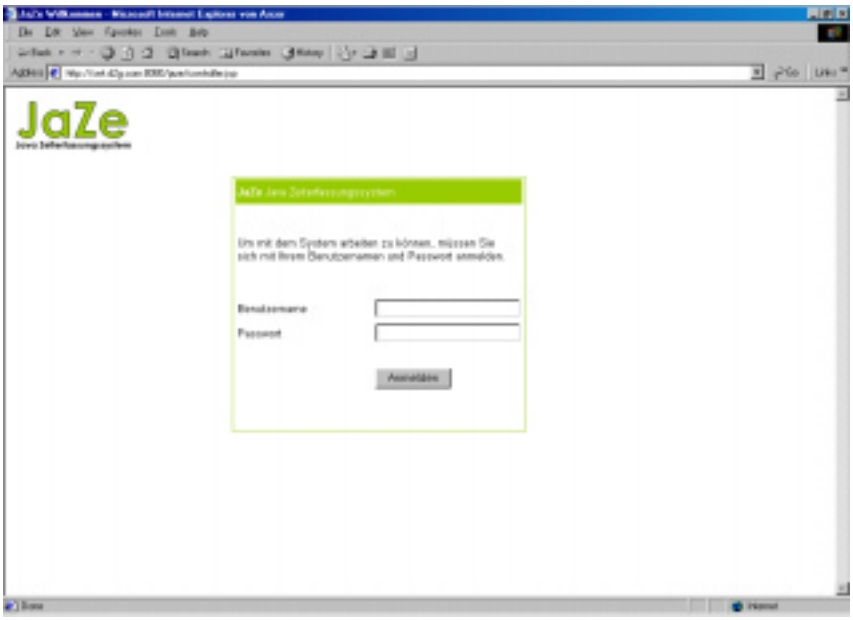
Vor der Entwurfsphase wurden jedoch ausführliche Gespräche mit den Veranstaltern des Projektes geführt, um sich einen Überblick über die zu realisierende Software zu verschaffen und um auf Wünsche einzugehen, die aus dem Interview hervorgegangen sind.

2. Die JaZe-Features

Die hier unten aufgelisteten Seiten geben einen Überblick über die Funktionsmerkmale, das Layout und die Interaktionsmöglichkeiten unserer Lösung wieder.

2.1 Anmeldung

Um die Dienste von JaZe nutzen zu können, müssen vorher ein Benutzername und ein Passwort eingegeben werden. Erst nach korrekter Anmeldung wird dem Benutzer der Zugang gewährt.



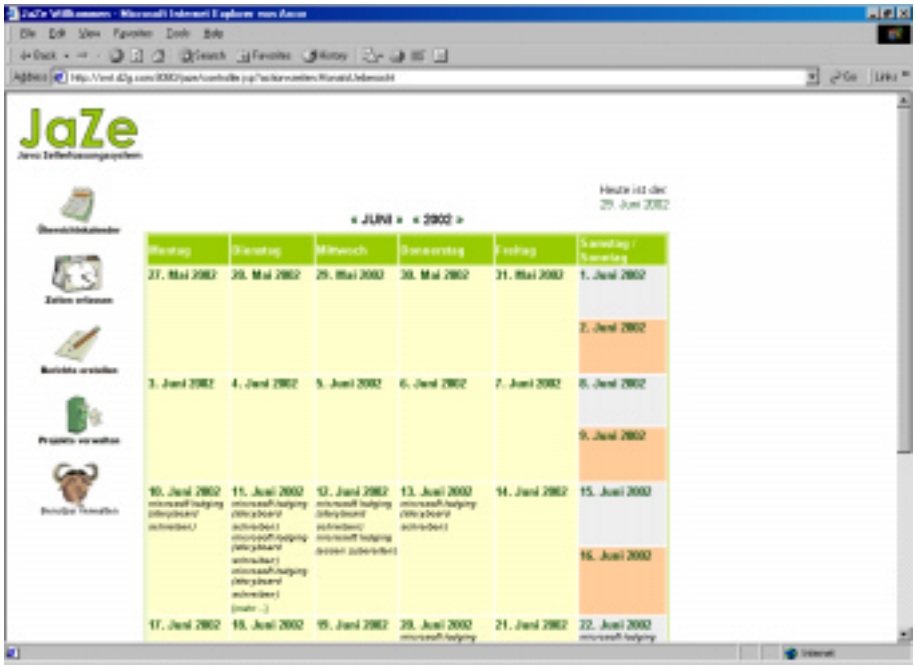
Bei unbekannten Benutzern oder falsch eingegebenen Daten wird eine Fehlermeldung auf dieser Seite angegeben.
Beim Anmelden des Administrators werden ihm zusätzliche Rubriken angezeigt, die ein "normaler" Benutzer nicht zu Gesicht bekommt. Zum Beispiel hat der Administrator die Möglichkeit Projekte und Benutzer zu verwalten usw. (siehe weiter unten).
Die Rubriken, die ein „normaler" Benutzer nicht verwenden kann, werden auch nicht in der linken Menüleiste angezeigt.

2.2 Übersichtskalender

Nachdem der Benutzer sich erfolgreich angemeldet hat und von dem System als Benutzer authentifiziert wurde, befindet er sich auf dem Startbildschirm. Dieser zeigt, wie oben schon erwähnt, Links ein Menü, über das er das Zeiterfassungs- und Verwaltungswerkzeug, sowie das Berichtswerkzeug aufrufen kann.

> „[...] Ein Werkzeug ist ein Gegenstand, mit denen Benutzer im Rahmen einer Aufgabe Materialien verändern oder sondieren können. Werkzeuge vergegenständlichen wiederkehrende Arbeitshandlungen wie z.B. ein Zeiteintrag. Materialien in unserem Kontext sind z.B. Benutzerdaten, die Teil eines Arbeitsergebnisses werden".**[Gryczan & Lammersdorf]**

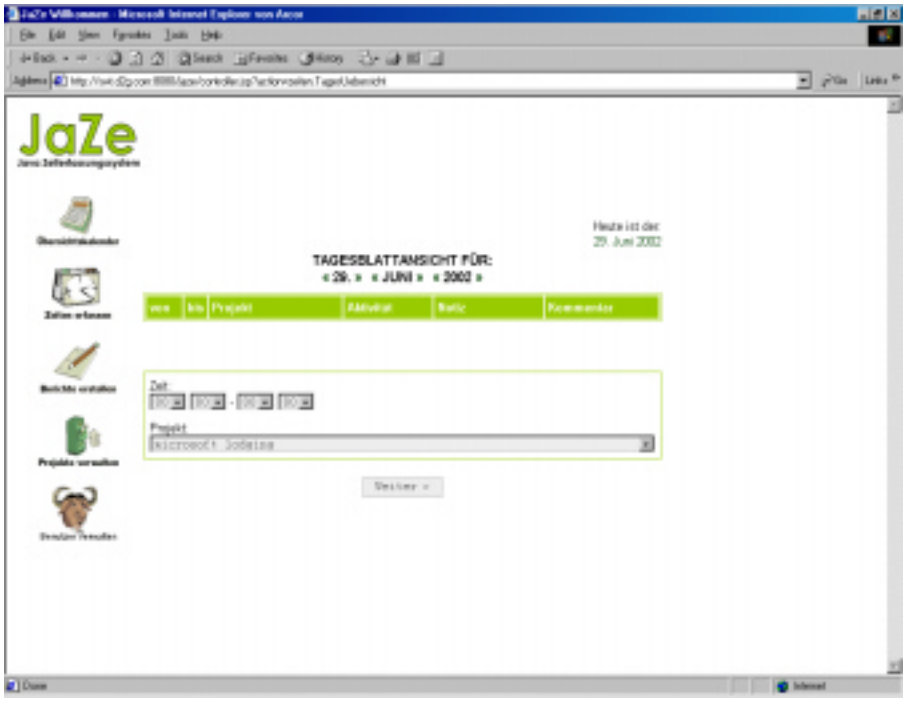
Der Hauptteil des Bildschirms wird von einer Kalenderübersicht ausgefüllt. In der Rubrik "Übersichtskalender" wird dem Benutzer eine grafische Monatsübersicht angezeigt. In den Tagen des Monats steht eine kurze Zusammenfassung der Projekte und Aktivitäten, die er an dem Tag geleistet hat bzw. Urlaub, Fortbildung, Krankheit oder ähnliches.
Mit einem Klick auf einen Tag (als Link dargestellt) ruft der Benutzer den entsprechenden Tageszettel auf. Sollte der Benutzer einmal ein Werkzeug aufgerufen haben, hat er die



Möglichkeit über das Menü wieder zur Übersicht zu kommen. Nun werden die Angaben dynamisch ergänzt bzw. modifiziert. Über dem Kalender besteht die Möglichkeit mit einem Mausklick entweder direkt über einen "Drop-Down-Menü" zu einem bestimmten Monat bzw. Jahr zu wechseln, oder durch erneutes klicken zum gewünschten Monat bzw. Jahr, als Pfeil dargestellt, zu gelangen.

2.3 Zeiterfassung

Der Benutzer öffnet das Werkzeug zur Erfassung und Verwaltung seiner Zeiten. In dem Hauptteil der Bildschirmauflösung öffnet sich eine Wochenübersicht. Der Zeiteintrag erfolgt folgendermaßen:
Für den Beginn und das Ende der Arbeitszeit trägt man sich selbsterklärend unter "Zeit" ein. Wichtig ist, dass die Angabe bis wann man gearbeitet hat größer ist, als die Zeit bei der man angefangen hat.



Ansonsten kommt man beim absenden dieser Daten auf diese Seite erneut zurück. Falls die Minutenangabe nicht angegeben wurde, wird automatisch mit "00" zur vollen Zeit abgerundet. Anschließend kann das Projekt mit der jeweiligen Aktivität angegeben werden, die dafür notwendig waren. Man achte aber darauf, dass die Projekte und Aktivitäten vorher schon eingetragen wurden sind (Projekte verwalten), weil ansonsten in der "Drop-Down-Menü" keine Einträge auszuwählen sind; diese Angaben sind notwendig.

Nach dem Absenden dieser Daten werden diese in die Datenbank übernommen und dynamisch der Seite hinzugefügt. Dieser Eintrag erscheint dann unter der grünen Menüleiste.

Bei Bedarf können die eingegebenen Angaben auch wieder rückgängig bzw. modifiziert werden. Dazu brauch man nur die gewünschte Zeile der Liste anzuklicken und kann seine Angaben korrigieren!

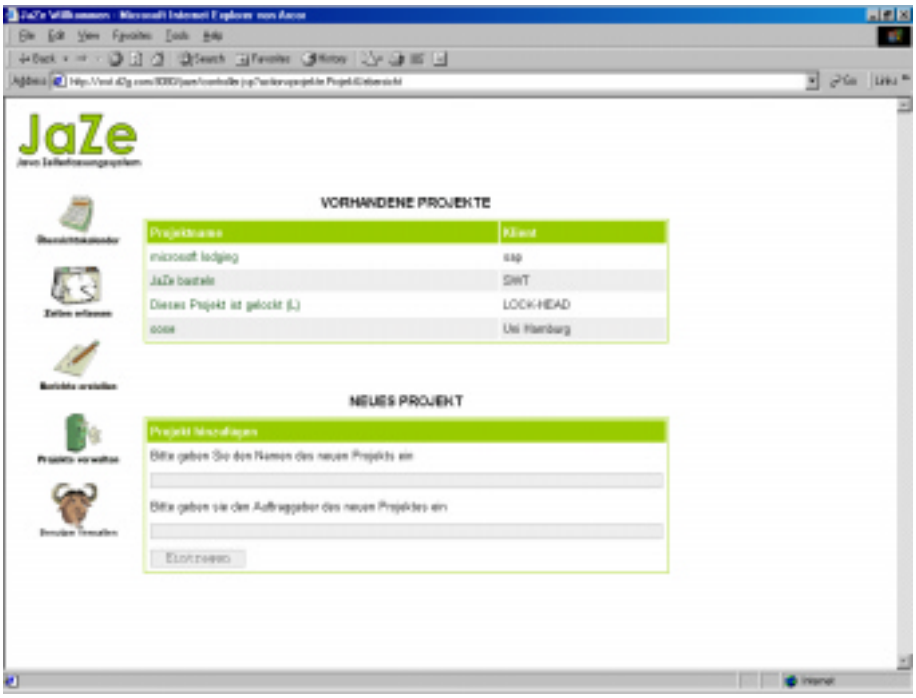
Auf der linken Seite befindet sich eine vertikale Menüleiste, von der man aus in die andere Bereiche kommen kann. Eine eingeschränkte Menüauswahl gibt es nur bei Anmelden als „normaler" User.

2.6 Berichtverwaltung

Der Benutzer öffnet das Werkzeug zur Berichtsverwaltung. Zuerst muss sich der Benutzer eine Berichtsart (Monatsbericht, Krankheits-/Urlaubsbericht, Individueller Bericht) auswählen.
Ist der Monatsbericht ausgewählt kann der Benutzer einen Monat auswählen, über den der Bericht erstellt wird. Zusätzlich hat er die Wahl sich alle oder ein bestimmtes Projekte, an denen er in dem Monat gearbeitet hat, anzeigen zu lassen. Der Monatsbericht kann von dem Benutzer abgeschlossen werden, was die bisherige Vorgehensweise, den Bericht zu drucken und unterschrieben abzugeben, ersetzen soll. Einträge, die einen geschlossenen Bericht beeinflussen kann der Benutzer nicht mehr ändern oder löschen. Der Krankheits-/Urlaubsbericht bietet ein wenig mehr als sein Name vermuten lässt. Es ist klar, dass Eintragungen notwendig sind, die zu keinem Projekt gehören können. Dazu gehören z.B. Krankheit und Urlaub, aber auch Fortbildung und ähnliches. Mit dieser Berichtsart kann man sich einen Bericht über die eben genanten „Aktivitäten“ geben lassen. Hierbei kann entweder alles oder nur bestimmte „Aktivitäten“ angezeigt werden. Der Zeitraum ist entweder ein Monat oder ein frei wählbarer Zeitraum.
Wünscht sich der Benutzer einen individuellen Bericht, wählt er diese Berichtsart. Hier kann er sich entweder einen der vorgegebenen Zeiträume auswählen (Monat, Quartal, Halbjahr, Jahr) oder einen freien Zeitraum bestimmen. Der Bericht über den von ihm gewählten Zeitraum kann über bestimmte Projekte oder Aktivitäten gehen oder einfach über alles.
Jeder der Berichte kann gedruckt werden.

2.4 Projektverwaltung

Über das Projektverwaltungswerkzeug können Projekte hinzugefügt, gesperrt oder bearbeitet werden. Gesperrte Projekte können nicht mehr zu Zeiteintragungen benutzt werden. Sie gelten als abgeschlossen.
In der Übersicht sieht man eine Liste mit allen Projekten und Ereignissen. Unter Ereignissen verstehen wir „Aktivitäten“, die keinem Projekt zugeordnet werden (Urlaub etc.). Um ein Projekt hinzuzufügen, wählt der Administrator die entsprechende Option und muss nun einen Namen und den jeweiligen



Auftraggeber für das Projekt vergeben. Auch hier werden die abgeschickten Daten in die Datenbank aufgenommen und dynamisch der grünen Menüleiste ausgegeben.

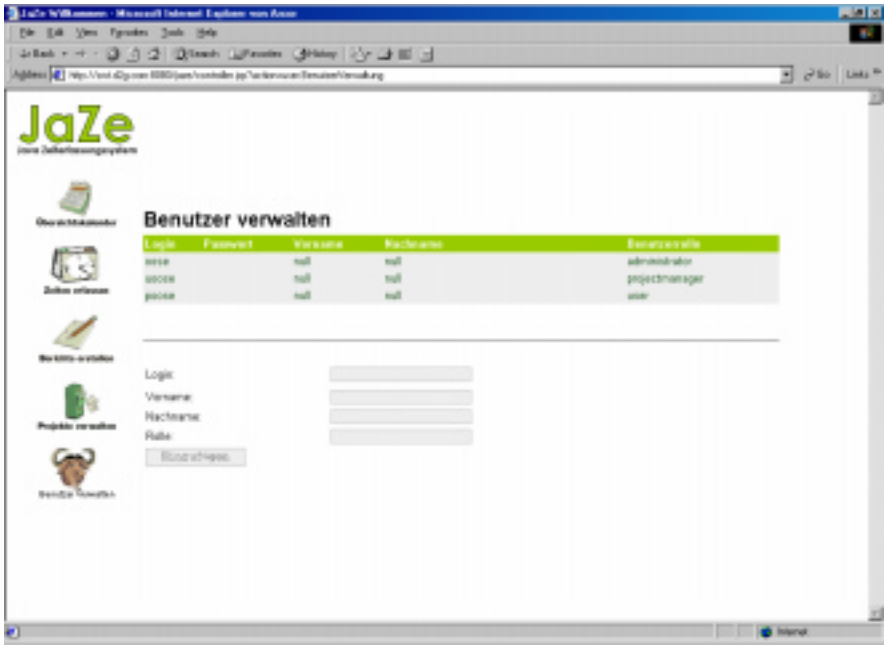
Diese können später in der Detailansicht bearbeitet werden. Hier findet sich auch die Funktion, um die Aktivitäten eines Projektes zu verwalten. Man kann Aktivitäten hinzufügen, ändern oder sperren. Gesperrte Aktivitäten können nicht mehr eingetragen werden.

2.5 Benutzerverwaltung

Das Werkzeug zum Benutzerverwalten gibt dem Administrator die Möglichkeit neue Benutzer ins System einzutragen, oder vorhandene Benutzer zu bearbeiten. In der Benutzerübersicht ist eine Liste aller Benutzer zu sehen. Hier kann man Benutzer hinzufügen oder sperren. Gesperrte Benutzer können sich nicht mehr im System anmelden und Einträge von ihnen können nicht mehr verändert werden.

Wenn ein neuer Benutzer eingetragen wird, müssen seine persönlichen Daten eingegeben werden. Alle anderen Daten werden über die Detailansicht des Benutzers bearbeitet. Hierzu wählt man einen Benutzer der Liste aus und

wechselt in die Detailansicht. Hierzu können alle Daten die zu den Benutzer gehören gesehen und verändert werden. Dazu gehören persönliche Daten (Loginname, Vorname, Nachname,..), Sollstunden des Mitarbeiters, Status des Mitarbeiters (Softwarearchitekt, Entwickler,...) und Urlaubstage.



Anschließend ist der neue Benutzer unter der grünen Menüleiste zu erkennen. Falls man Benutzer modifizieren bzw. löschen möchte, muss zunächst dies durch anklicken des jeweiligen Eintrages in der Liste erfolgen. Danach werden die Personenangaben in den Felder aufgefüllt, so dass diese dann verändert werden können. Zum Schluss werden die neuen Angaben abgesendet oder der ausgewählte Benutzer aus der Liste und somit auch aus der Datenbank gelöscht.

3. Service Architektur

Die „Architektur" soll versuchen zu erläutern wie das Konzept der Fachlichen Services in unserem Projekt strukturiert sind. Ich gebe zu Beginn einen kleinen Überblick im Abschnitt 3.1 zu den einzelnen Schichten der Architektur und werde darauf später ausführlicher eingehen.

Ich erläutere dann unter anderem etwas über das in der Anwendung verwendete Singleton-Entwurfsmuster und werde dann übergehen zum Konzept des OR-Mapping. Die Aufgabe des OR-Mapping ist es nämlich, die Objekte, die für sich individuell und einzigartig sind, mit den Attributen und ihrer Identität auf die Datenbank abzubilden.

3.1 Architekturüberblick

Wie wir eben bei den JaZe-Features (siehe 2. „JaZe-Features") gesehen haben, stellt unser Frontend auf dem Client die zahlreichen JSP's dar. Über diese Benutzungsschnittstelle werden die Seiten dargestellt („Präsentation") und man hat als Benutzer die Möglichkeiten mit dem System zu interagieren („Handhabung"). Im Abschnitt "Request Controller Architektur" befassen wir uns noch ausführlicher damit, wie unsere Webanwendung eine bessere Übersichtlichkeit durch eine stärkere Trennung von Präsentation und Inhalt bekommen kann.

Der Interaktionsservice stellt die Schnittstelle zwischen unserem Frontend und den Diensten, die wir anbieten dar. Es ermöglicht uns mit Hilfe des Remote-Zugriffs (siehe 6. „Remote Method Invocation") die Dienste des JaZe auf dem Server so zu nutzen, als würde sich die Dienste lokal befinden.

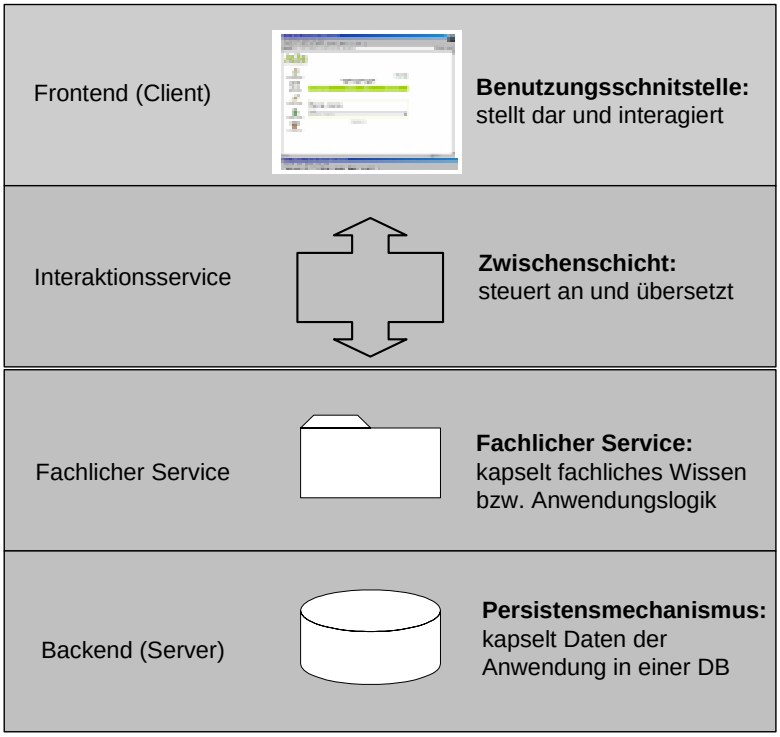


Abb.: Fachlicher Service: Das Konzept **[Gryczan & Lammersdorf]**

In unserem Backend bzw. Datenbank kapseln wir die Daten unserer Anwendung.
Das OR-Mapping (siehe 3.3 „Einordnung von OR-Mapping im Projekt"), hat nämlich die Aufgabe Objekte, die für sich individuell und einzigartig sind, mit den Attributen und ihrer Identität auf die Datenbank abbilden zu können. Das bringt den wesentlichen Vorteil, das Materialien in der Datenbank versionsunabhängig sind, gegenüber einer Serialisierung in einer Datei, wo Versionskonflikte entstehen können.
Dieses Konzept macht den Zugriff auf die Datenbank eleganter und einfacher.

In diesem Fall sind es die Fachlichen Services, die unser fachliches Wissen bzw. die Anwendungslogik des Anwendungsbereichs zusammenfassen und den Benutzer über die eben erwähnte Benutzungsschnittstelle angeboten werden. Fachliche Services lassen sich einfach durch Session-Beans realisieren.

Unsere Anwendung bietet folgende fachliche Services an, z.B. den **UserService**, **AuthenticationService**, **CalenderService** und den **ProjectService**.

Ich möchte an dieser Stelle den **ProjectService** kurz erläutern:

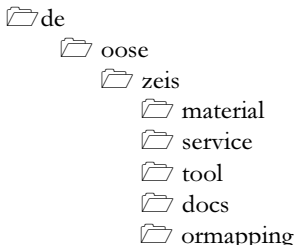
```
public interface ProjectService extends Remote {

public void saveProject (Project project)
       throws java.rmi.RemoteException;
public boolean existProject (String id)
       throws java.rmi.RemoteException;
public boolean existProjects ()
       throws java.rmi.RemoteException;
public String[] getAllProjects ()
       throws java.rmi.RemoteException;
public Project getProject (String id)
       throws java.rmi.RemoteException;
public void deleteProject (String id)
       throws java.rmi.RemoteException;
}
```

Dieser Service bietet die Möglichkeit an Projekte zu verwalten. Das heißt, man kann vorhandene Projekte an diesen Service übergeben, der dann entweder gespeichert oder wieder gelöscht wird. Hinzu kommt noch, dass der Benutzer die gesamten Projekte bzw. ein bestimmtes Projekt zurückgeben lassen kann.

Der Service ist also nichts anderes als ein „Ort“ an den man seine Materialien –in diesem Fall sind es die Projekte- übergibt und anschließend die angebotenen Dienstleistung nutzt. (siehe Autoreperatur).

3.2 Paketstruktur

Aufgrund der Übersichtlichkeit und der klaren Trennung der fachlichen und technischen unterscheidbaren Bereichen wird die Paketstruktur folgendermaßen gewählt.



Der Pfad **de.oose.zeis**, in dem das Projekt liegt, ist an die in Java üblichen Struktur angelegt. In diesem Fall finden sich im "Hauptpfad" **zeis** die Pakete "**material**", welcher alle Materialien enthalten soll. Das Paket **service**, in dem sich die verschiedene Services, auf der Anwendungen läuft, befinden, sowie **tool**, welches die Werkzeuge und ihr grafische Oberflächen enthält. Im Paket **docs** finden sich nützliche Dokumente wie z.B. die "Programmierrichtlinien". Diese Richtlinien geben den beteiligten Entwickler in der Gruppe eine Hilfestellung bezüglich der Programmierung (z.B. JavaDoc, Kommentare, Methoden,…).

Zu guter letzt enthält der Ordner **ormapping** die notwendigen Klassen, die für das Mapping relevant sind; Details gibt es im Abschnitt „OR-Mapping".

3.3 Entwurfsmuster „Singleton"

„Ein Entwurfsmuster ist eine Abstraktion einer konkreten Form, die wiederholt in bestimmten Kontexten auftritt" ***[Gtyczan & Lammersdorf].*** *Das Singleton-Entwurfsmuster wird in unserer Anwendung für den „ServiceProviderManager" verwendet, welches alle verschiedene Services enthält und diese bei Bedarf wiederholt den Werkzeugen oder anderen Services zur Verfügung stellt. Da es in unserer Anwendung nur einen „Service Provider Manager"- Exemplar geben soll, haben wir uns für das Singleton entschieden.*

Das Singleton sichert ab, dass eine Klasse genau ein Exemplar besitzt.
Bei manchen Klassen ist es wichtig, dass es genau ein Exemplar gibt.

Eine globale Variable ermöglicht den Zugriff auf ein Objekt und stellt sicher, dass eine Klasse über genau ein Exemplar verfügt und dass einfach auf diese Exemplar zugegriffen werden kann.
Es ist somit besser, die Klasse selbst für die Verwaltung ihres einzigen Exemplars zuständig zu machen. Die Klasse kann durch Abfangen von Befehlen zur Erzeugung neuer Objekte sicherstellen, dass kein weiteres Exemplar erzeugt wird, und sie kann die Zugriffsmöglichkeit auf das Exemplar anbieten. Die ist die Notwendigkeit des Singletonmusters.

Das Singletonmuster besitzt mehrere Vorteile:
1. Zugriffskontrolle auf das Exemplar
2. Eingeschränkter Namensraum
3. Verfeinerung von Operationen und Repräsentation
4. Variable Anzahl von Exemplaren
5. Flexibler als Klassenoperationen

Zum Schluss möchte an einem Beispielcode unserer Anwendung zeigen, wie wir das Singleton - Entwurfsmuster verwendet haben:

```
public class ServiceProviderManager {
 private static ServiceProviderManager _instance = null;
protected ServiceProviderManager(){…;}
public static getServiceProviderManager(){
if(_instance == null){
_instance = new ServiceProviderManager();
}
return _instance;
}. . .
```

Die Member-Variable *_instance* (die einen Zeiger auf sein einziges Exemplar enthält) wird mit *null* initialisiert, und die statische Member-Funktion *getServiceProviderManager* gibt seinen Wert zurück, wobei sie ihn mit dem einzigen Exemplar initialisiert, wenn er null ist.
Beachten Sie, dass der Konstruktor geschützt ist, das heißt als *protected* deklariert ist. Versucht ein Klient, ein Singleton direkt zu erzeugen, so ergibt sich zur

Übersetzungszeit ein Fehler. Dies stellt sicher, dass nur ein Exemplar erzeugt wird.

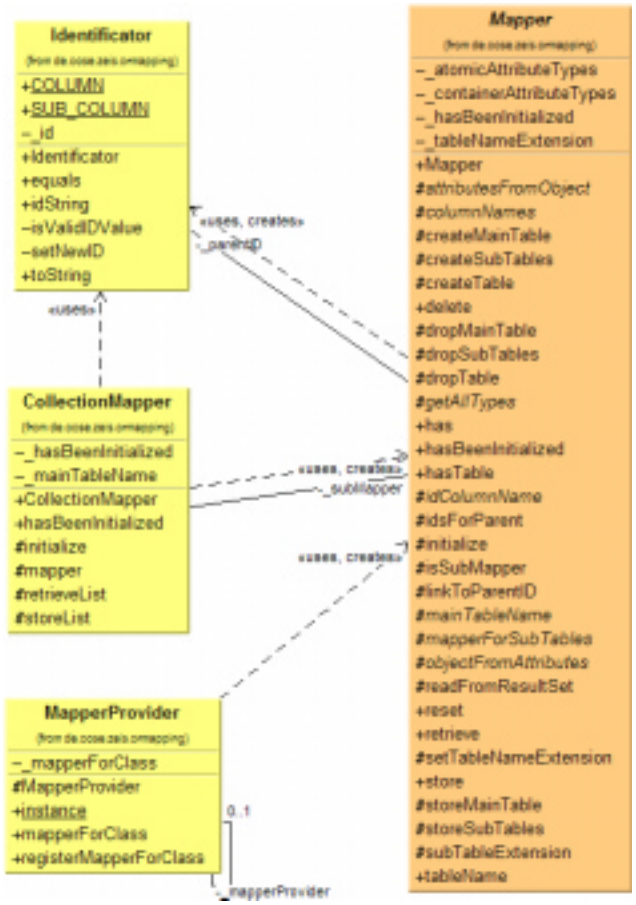
3.4 Einordnung von OR-Mapping im Projekt

Im JaZe-Projekt der Gruppe 1 werden die eingegebenen Daten über fakturierte Zeiten, laufende Projekte sowie Mitarbeiter (gleichzeitig Benutzer des Systems) etc. in einer MySQL-Datenbank persistent gehalten. Der Zugriff auf die Materialien, welche diese Daten für die Anwendung kapseln, erfolgt über die sogenannten Services (z.B. einem ProjectService, welcher die anwendungsinterne Verwaltung der Projekte übernimmt), welche über sogenannte Mapper auf der Datenbank arbeiten. Während die Services auf der einen Seite die Handhabung der Materialien in der Anwendung übernehmen und somit die Anwendung freihalten von Details der Archivierung, dem Suchen nach bestimmten Materialien sowie der Ausgabe von Materialien etc., verdecken die Mapper die Grundfunktionalitäten der Datenbank vor den benutzenden Services, wie zum Beispiel das Schreiben und Lesen von Materialien in der Datenbank, das Anlegen der zugrundeliegenden Tabellen, das Verknüpfen von enthaltenen Materialien mit deren „Besitzern“, eben das Mappen von in den Tabellen einer Datenbank stehenden Werten auf Attribute eines Materials (oder mehrerer verknüpfter). Die Mapper übernehmen also die Translation von einer Instanz einer Materialklasse in die entsprechenden Tabellen(spalten) und umgekehrt.

3.5 Die OR-Mapping-Klassen und ihre Funktion

Der grundlegende Aufbau des OR-Mappings im JaZe-Projekt ist im folgenden beschrieben: alle für das OR-Mapping relevanten Klassen liegen im Package `de.oose.zeis.ormapping`; einzige Ausnahmen bilden die Konfigurationsdatei der Anwendung (in der möglicherweise die Verbindungsparameter für die Datenbank stehen) sowie der Datenbank-Service (DatabaseService), welcher sich im Package `de.oose.zeis.service` befindet. Im Letzteren wird der Verbindungsaufbau zur Datenbank gehandhabt. Diese Verbindungen (Connections) zur Datenbank werden für die Benutzung der Mapper unbedingt benötigt, weswegen alle Operationen an Mappern eine solche Verbindung verlangen.

Im OR-Mapping-Package sind die hier aufgeführten Klassen enthalten:



- **`Mapper.java`**: eine abstrakte Klasse, welche die grundlegende Mapperfunktionalität enthält. Das umfasst das Speichern, das Laden sowie das Löschen von Materialien. Dabei wird immer eine Verbindung und eine Material-ID, beim Speichern auch noch das zu speichernde Material mitübergeben. Weiterhin sind im Mapper auch die Methoden zum Erstellen und Löschen der dazu nötigen Datenbanktabellen vorgesehen.

- **`MapperProvider.java`**: am Mapper-Provider werden bei Anwendungsstart alle für die Anwendung benötigten Materialmapper registriert, und zwar derart, daß jeweils die Materialklasse sowie der dazugehörige Mapper der `register()`-Methode übergeben wird. Innerhalb der Services, die lesend oder schreibend auf die vorhandenen Materialien (also auf die Datenbank) zugreifen sollen, kann dann über den MapperProvider der für ein Material passende Mapper geholt werden.

- **`MappingFactory.java`**: an der MappingFactory werden die atomaren Attributstypen registriert, die sich direkt in die Datenbank schreiben lassen, so zum Beispiel String oder Integer. Mithilfe der MappingFactory läßt sich somit ein datenbank-spezifisches Format für verschiedene Werttypen festlegen. So können zum Beispiel Strings in der Datenbank in Feldern mit dem Typ VARCHAR und einer bestimmten Länge abgelegt werden, oder Festkommazahlen in Felder, die auf zwei Nachkommastellen festgesetzt sind (für z.B. Währungen). Anhand der in der MappingFactory registrierten Typen kann der abstrakte Mapper schon größtenteils generisch für alle Materialien Tabellen erstellen und Materialien in die Datenbank schreiben bzw. aus der Datenbank holen.

- **`Identificator.java`**: eine Klasse zur Identifizierung von Materialinstanzen (liegt im Package de.oose.zeis.material). Für jedes neuerzeugte Material wird eine eindeutige ID angelegt, über die das Material anwendungsweit eindeutig identifiziert werden kann. In den Mappern wird diese ID zusätzlich dazu benutzt, um Datensätze in der

Datenbank eindeutig Materialien zuzuordnen und Verknüpfungen zwischen Materialien und in ihnen enthaltenen Materialien herzustellen.

- **`CollectionMapper.java`**: in JaZe werden zwei Formen von Collections unterstützt, LinkedLists und HashMaps. Für die erste Form von in Materialien enthaltenen Collections wird in der Datenbank einfach eine Subtabelle für die in der Liste enthaltene Materialklasse angelegt, in der das Besitzermaterial (in welchem die Liste vorhanden ist) über seine ID referenziert wird. Bei der Hashmap gibt es eine Referenztabelle, welche zum einen die Besitzer-ID für jeden eingetragenen Datensatz enthält sowie jeweils eine Schlüssel- und eine Wert-Referenz-ID, welche die zugehörigen Materialien in den entsprechenden Subtabellen für die Schlüssel- und Wert-Materialklassen enthält.

- **`DatabaseService.java`**: verbindet die Anwendung mit der Datenbank und kann einzelne Verbindungen zum Ausführen von Abfragen herausgeben. Die Unterstützung von Transaktionsklammern sowie von Caching wurde nicht implementiert, da es von MySQL anscheinend in der derzeitigen Version nicht richtig unterstützt wird.

Man kann zusammenfassend sagen, dass das objektrelationale Mapping immer dann zum Einsatz kommt, wenn relationale Datenbanken und objektorientierte Programmiersprachen verwendet werden. OR-Mapping ist somit die Antwort auf die Frage „...wie bekomme ich die Daten von den Java Objekten in die relationale Datenbank, und wie verwandle ich die relationalen Tabellen wieder in schöne Java Objekte?“
Der Gebrauch des objektrelationalen Mapping bedeutet aber auch, dass man viel weniger Code schreiben muss, und abhängig von der Datenverwaltung höhere Performanz erreicht.

4. Request Controller Architektur

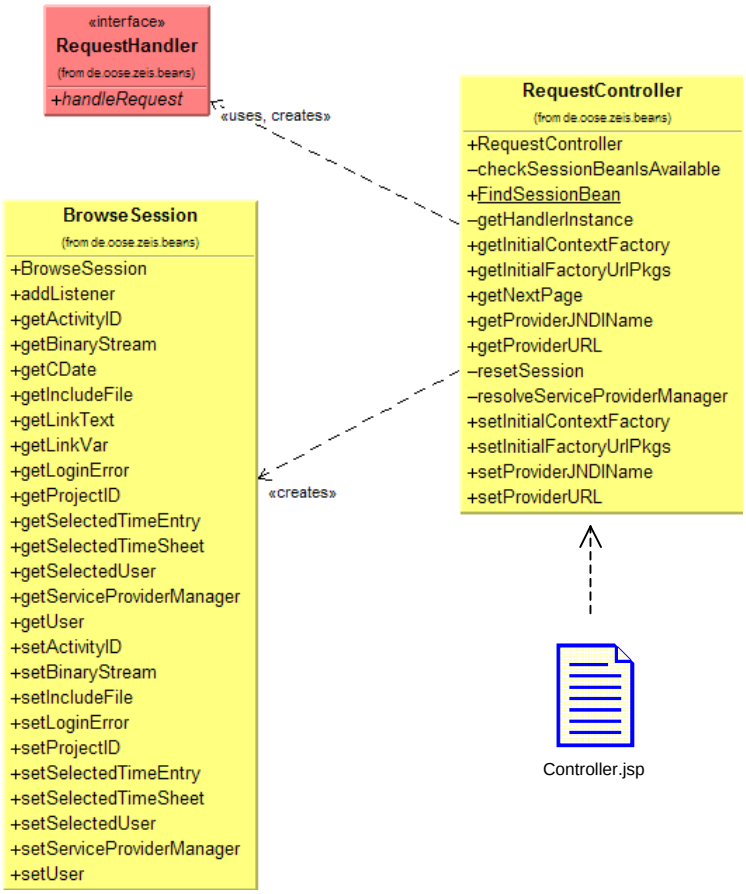
In diesem Abschnitt befassen wir uns mit der "Request Controller Architektur", die in unsere Webanwendung eine bessere Übersichtlichkeit durch eine stärkere Trennung von Präsentation und Inhalt bringen soll. Somit wird verhindert, dass keine Businesslogik mit Repräsentationslogik vermischt wird.

4.1 Einführung

Dieser Abschnitt soll einen Überblick über die von uns gewählte Architektur der Webanwendung liefern. Es wird sowohl das Zusammenspiel der einzelnen Komponenten, als auch deren Aufgaben erläutert. Zu guter letzt wird dann noch beschrieben, wie man neue Seiten in die Anwendung integriert; diese Beschreibung ist daher so wichtig, da die Programmierer sich leichter damit befassen können.
Um diesen Abschnitt zu verstehen sollte man allerdings über Vorkenntnisse bezüglich JSP's und Servlets verfügen. Nach dem lesen dieses Abschnitts sollte man dann in der Lage sein, neue Seiten in die Anwendung zu integrieren.
Für das bessere Verständnis werden ausführliche Codebeispiele gezeigt!

4.2 Die Architektur

Diese Architektur soll für eine stärkere Trennung von Präsentation und Inhalt sorgen. Änderungen des Zustands (wird im Session Bean gehalten) sollen nicht innerhalb der JavaServerPages (JSP) vorgenommen werden und auch die Flusskontrolle der Seiten soll nicht innerhalb der JSP's geregelt werden. Dies sorgt für größere Übersichtlichkeit in den JSP Seiten und so finden sich auch Webdesigner und Programmiere besser innerhalb der JSP's zurecht. Das hat den Vorteil, dass sich der Webdesigner um seine Webseiten und der Programmierer unabhängig vom Webdesigner seinen JavaCode schreiben



könnte, ohne sich in den Bereichen des anderen auszukennen. Des Weiteren lassen sich die Klassen (Requesthandler) wesentlich besser kommentieren als JSP's (Stichwort JavaDoc).

4.3 Komponentenübersicht

Das im Anschluss aufgeführte Sequenzdiagramm soll nun Aufschluss darüber geben, wie die einzelnen Komponenten nach der Request Controller Architektur aufgerufen werden und diese miteinander interagieren. Nähere Informationen zu den einzelnen Komponenten werden weiter unten behandelt.

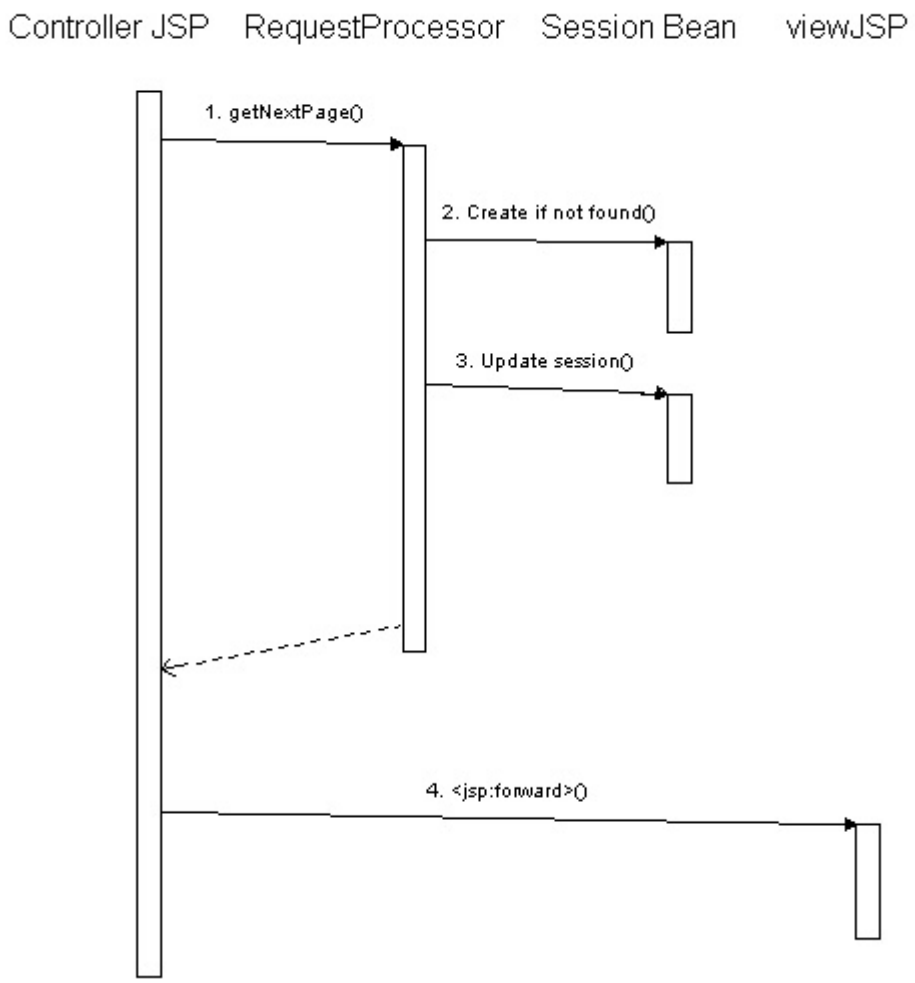


Abb: *Sequenzdiagramm*

Nun näheres zu den einzelnen Komponenten:

Controller JSP:

Eine relativ kleine JSP die eigentlich nur den RequestProcessor instanziiert und diesen dazu benutzt um die zu dem aktuellen Request als nächstes anzuzeigende Seite zu ermitteln.

```
<jsp:useBean id="controller" scope="session"
class="de.oose.zeis.beans.RequestController" />

<jsp:forward page="<%=controller.getNextPage(pageContext,
request)%>"/>
```

^

In Anwendungen ist es unter Umständen durchaus sinnvoll mehrere Controller zu benutzten, aber für unser kleines Projekt sollte einer reichen. Diese Seite beinhaltet keinen HTML-Code und ist somit für Webdesigner uninteressant, sie leitet lediglich auf die ermittelte JSP-Seite weiter.

RequestProcessor und Hilfsklassen:

Der RequestProcessor entscheidet für jeden Request welche View (JSP) als nächstes angezeigt werden soll.

```
public String getNextPage(PageContext pageContext,
                          HttpServletRequest request) throws
                        JspException {

 String action = request.getParameter("action");

// Sollte keine action gesetzt sein, wird der User auf die
login Seite geleitet

        if (action == null || action.equals("")) {

            resetSession(pageContext);
            return LOGIN_PAGE;
        }

// Ist action gesetzt wird der zugehörige requesthandler
instanziert


RequestHandler requestHandler = getHandlerInstance(action);

// das Session Bean wird erzeugt, sollte es noch nicht
exisistieren


        checkSessionBeanIsAvailable(pageContext);

// verarbeitet den Request und liefert die URL des JSP's
      try{
          return requestHandler.handleRequest(pageContext,
request);
      } . . .
. . .
      }
    }
```

Des Weiteren sorgt der RequestProcessor dafür, dass die Session aktualisiert wird und der aktuelle Zustand der Session kann auch mit auf die anzuzeigende JSP Einfluss nehmen. Es gibt zu jeder Session einen RequestProcessor. Zum RequestProcessor gehören in unserem Fall folgende Klassen:

RequestController (Liefert mit Hilfe der Reflection API den zuständigen RequestHandler), **RequestHandler** ist das Interface das die konkreten RequestHandler implementieren müssen.

Session Bean:

Das Session Bean hält den aktuellen Zustand der Session, so zum Beispiel den aktuellen User. Als weitere wichtige Funktion, kann über das Session Bean auf den **ServiceProviderManager** und mit Hilfe dessen dann auf die Services zugegriffen werden.

```
public class BrowseSession implements Serializable {
. . .

//setzt den neune User
    public void setUser(User paramUser) {
            _user = paramUser;
    }

//Liefert den an die Session gebundenen User
    public User getUser() {
            return _user;
    }

//Liefert den ServiceProviderManager
 public ServiceProviderManager getServiceProviderManager() {
            return _serviceProviderManager;
    }

//setzt den ServiceProviderManager
 public void setServiceProviderManager(ServiceProviderManager
_serviceProviderManager) {
        this._serviceProviderManager = _serviceProviderManager;
    }
```

View JSP's:

Die eigentlichen Webseiten die im Allgemeinen nur eine Präsentation des Zustandes (Session Bean) darstellen. Sie beinhalten im Idealfall keine eigentliche Funktionalität.

4.4 Wie werden neue Seiten integriert?

Als erstes sollte die zugehörige JSP Seite erstellt werden. Sollen auf dieser Seite in irgendeiner Form Daten aus dem Session Bean benutzt werden, ist die einfachste Syntax, um zu spezifizieren, dass eine Bean verwendet werden soll, ist wie folgt:

```
<jsp:useBean id="browseSession" scope="session"
type="de.oose.zeis.beans.BrowseSession" />
```

Hierbei zeigt `scope="session"` an, dass ein Bean-Objekt nicht nur an eine lokale Variable gebunden, sondern auch in dem mit der aktuellen Anfrage verknüpften Http-Session-Objekt gespeichert wird.

Um zu signalisieren, dass die Seite zur Session gehört ist folgender Eintrag nötig:

```
<%@ page session="true"%>
```

Dann müssen nur noch die Klassen importiert werden die benutzt werden sollen und dann kann ganz normal auf die Methoden des Session Bean zugegriffen werden.

Solle es auf dieser Seite ein Formular geben, so sollten die Daten an die Controller JSP gesendet werden. Das Formular muss nun noch ein hidden Feld mit dem Namen „action" beinhalten, dessen value bestimmt welcher RequestHandler dann letztendlich benutzt werden soll, zum Beispiel:

```
<input type="hidden" name="action"
value="packagepfad.MyRequestHandler">
```

So wird vom RequestController die entsprechende Klasse mit dem angegebenen relativen Pfad benutzt. Der angegebene relative Pfad muss aber

nur unterhalb des package beans.requesthandlers angegeben werden, den Rest erledigt der RequestController von alleine.

Handelt es sich nicht um ein Formular, sondern um einen normalen link, so kann das Action-Attribut einfach nach dem Verweis auf die Controller.jsp nach einem Fragezeichen angehängt werden, zum Beispiel:

```
href="controller.jsp?action=packagepfad.MyRequestHandler"
```

Nun kann man im angegebenen Package den RequestHandler implementieren.
Die konkrete Requesthandlerklasse muss dann das RequestHandler Interface implementieren und in der Methode handleRequest wird sowohl das Aktualisieren der Session durchgeführt, als auch der Name der auzuzeigenden JSP zurückgeliefert.
In unserem Fall wird häufig einfach die template.jsp benutzt und so muss innerhalb des RequestHandlers das richtige includeFile in der Session gesetzt werden und dann die template.jsp als nextPage zurückgeliefert werden.

Nun kann diese Seite auch in anderen Requesthandlern als anzuzeigende Seite benutzt werden und ist somit voll in unsere Anwendung integriert.

5. Remote Method Invocation

Für unsere Anwendung ist dieser Mechanismus von großer Bedeutung, da die Benutzer in der Lage sein sollen, die Dienste des JAZE-Software auf einem entfernten Rechner nutzen zu können.
Das heißt, dass ein Mitarbeiter einer Firma nicht ständig in die Firma fahren muss, um seine Arbeitsstunden einzutragen oder um Projektberichte zu erstellen. Das zeigt, dass der Zugriff über das Netz auf diese Anwendung sehr wichtig ist.
RMI gehört nach dem Konzept unserer Architektur zur Interaktionsschicht, die als Schnittstelle zwischen der Präsentationsschicht und der Anwendungsschicht dient. Sie erlaubt den Benutzer die entfernten Dienstaufrufe.

5.1 Einführung

Der Mechanismus des Remote-Methodenaufrufs erlaubt den Zugriff eines Objektes auf Operationen eines Objektes auf einem anderen Computer. Das heißt, das ein Remote-Server seine Dienste und Regeln für deren Benutzung über eine Kommunikationsschnittstelle exportiert und ein Client diese Dienste nutzen bzw. importieren kann.

Natürlich müssen die Methodenparameter zum anderen Computer gelangen, das Objekt ist über die Ausführung der Methode zu informieren und den Rückgabewert der Methode muss zurückgeliefert werden. Remote Method Invocation behandelt diese Details. „ Die Codierung der Parameter bezeichnet man als Parameter-Marshalling. Der Zweck des Marshalling besteht darin, die Parameter in ein Format zu konvertieren, das sich für den Transport von einer virtuellen Maschine zu einer anderen eignet“ **[Horstmann & Cornell].**

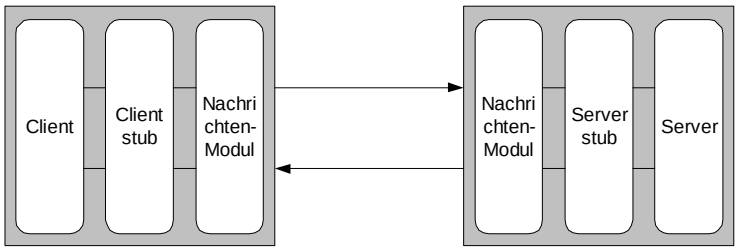
Durch **RMI** *(für Java-Objekte)* ist man somit in der Lage **zwei Objekte** auf unterschiedlichen Computer und somit auf *unterschiedlichen virtuellen Java-Maschinen* zu verwenden. D.h. ein Objekt befindet sich auf dem Client, wo der Benutzer ein Formular ausfüllt. Das Client-Objekt sendet eine Nachricht mit den Details der Anforderung an ein Objekt auf dem Server. Das Server-Objekt trägt die angeforderten Informationen zusammen, z.B. eine Datenbankabfrage. Nachdem das Server-Objekt die Antwort auf die Client-Anforderung hat,

sendet es sie zurück an den Client. Für den Anwender ist dieser Vorgang *transparent*!

Stub: Wenn der Client-Code eine ***Remote-Methode*** auf einem Remote-Objekt aufrufen will, ruft er ein gewöhnliche Java-Methode auf, die in einem ***Ersatzobjekt*** namens *Stub* verkapselt ist(siehe Abbildung unten). Der *Stub* befindet sich auf dem Client-Computer. Der Stub verpackt die im Remote-Methodenaufruf verwendeten ***Parameter*** als Block von Bytes.

Skeleton: Daten kommen an und die Parameter werden entpackt. Er sucht den Standort des aufzurufenden Objekts und ruft dann die gewünschte Methode auf. Er übernimmt den Rückgabewert oder die Ausnahme und schickt diese dann an den *Stub* des Client zurück

5.2 Aufruf entfernten Operation



Client:

- Client führt seinen lokalen Prozeduraufruf durch
- Client-Stub, der in der Sprache des Client geschrieben ist, verpackt die Argumente für den entfernten Prozeduraufruf.
- Das Nachrichtenmodul sendet die Anfrage

Server:

- Das Nachrichtenmodul nimmt die Anfrage entgegen
- Server-Stub, der in der Sprache des Servers geschrieben ist, entpackt die Argumente und recht sie weiter
- Server führt die Prozedur aus und gibt die Antwort als Rückgabe über den selben Mechanismus wieder zurück.

Voraussetzung ist die Standardisierung der Schnittstelle und des Protokolls, um eine Interoperabilität bezüglich der Anwendung gewährleisten zu können. Der Aufruf entfernter Methoden, die nach Außen wie ein lokaler Aufruf erscheint, nennt man *Verteilungstransparenz*.

Java RMI erweitert den Remote Procudere Call (RPC) Mechanismus vom Aufruf entfernter Prozeduren auf den Aufruf entfernter Methoden. Im grundsätzlichen Schema sind daher RPC und RMI ähnlich. RMI ist eine Java spezifischen Verkörperung des entfernten Methodenaufrufs. Es hat damit Ähnlichkeit zu der *Common Objekt Request Broker Architecture (CORBA)*, das aber auch Objekte verschiedener Programmiersprachen zu kapseln erlaubt. Durch die Einschränkung auf Java ist RMI in manchem einfacher wie CORBA, erlaubt aber auch Möglichkeiten, die im allgemeineren CORBA Rahmen nicht vorgesehen sind.

Um nun unsere fachlichen Services, wie z.B. den **ProjectService** entfernt nutzen zu können, wird dieses eben erwähnte Verfahren darauf angewandt. Für eine ausführliche Implementation und Beschreibung verweise ich auf den Anhang Kapitel 8.2.

6. Fazit

Wir haben auf Grund von Zeitproblemen leider am Ende des Projektes ein paar Dinge nicht mehr schaffen können

Dazu gehört einerseits eine SWING basierte Anwendung, um unsere Anwendung nicht nur über das Internet nutzen zu können.

Andererseits wären Eigenschaften wie die „Security“ (verschlüsselte Übertragung, Firewalls,...) hier sehr nützlich gewesen.

Der letztere Punkt ist sehr wichtig, da wir diese Lösung momentan nur über das Netz nutzen können, und die Übertragung von relevanten Daten ungeschützt verschickt werden.

7. Literaturverzeichnis

Bücher

[Coulouris et al. 02] George Coulouris, Jean Dollimore, Tim Kindberg, *Verteilte Systeme* Auflage, Addison Wesley, 2002, ISBN 3-8273-7022-1

[Flanagan 98] David **Flanagan**: *Java Examples in a Nutshell.* Auflage, O'Reilly, 1998, ISBN 3-89721-112-2.

[Gamma et al. 96]Gamma, Helm, Johnson, Vlissides, *Entwurfsmuster –Elemente wiederverwendbarer objektorientierter Software* Addison Wesley

[Gryczan & Lammersdorf] Folien der Vorlesung „**Verteilte Softwaresysteme** (VSS)" . Sommersemester 2002
http://swt-www.informatik.uni-hamburg.de/teaching

[Hall01] Marty **Hall** , *Core Servlets und JavaServerPages,* Markt und Technik Verlag

[Horstmann & Cornell] Cay S. **Horstmann**, Gary **Cornell**, *Core Java 2.* Band 2 –Expertenwissen , 2000, ISBN 3-8272-9566-1

[Züllighoven98] Heinz **Züllighoven,** *Das objektorientierte Konstruktionshandbuch* Dpunkt Verlag

Links

1. www.apache.org (Tomcat und ANT)
2. www.java.sun.com (JDK)
3. www.mysql.com (Datenbank)
4. www.cvs.org (CVS)

8. Anhang

Im Anhang finden Sie zusätzliche Informationen, die ich nicht zwingend notwendig, aber hilfreich sind. Hier kann man eine ausführliche Installation unserer Anwendung nachlesen, falls Interesse vorhanden sein sollte dies auch mal auszuprobieren.

Anschließend finden Sie Konfigurationshinweise zu RMI, die als eine Art Hilfestellung zu betrachten ist.

8.1 Installationsanleitung

Um die Lösung unserer Gruppe zu testen, wird in diesem Abschnitt eine ausführlich Installation beschrieben. Leider kann zum Zeitpunkt dieser Baccalaureatsarbeit die Installation nur in Form einer Anleitung gegeben werden.

Nach dem die Programme sich auf Ihrer Festplatte befinden, installieren Sie das **JDK**, danach installieren Sie **MySQL**, dann entpacken Sie die ZIP-Datei mit dem **TomCat**.
Jetzt erstellen Sie im Windows-Explorer eine neues Verzeichnis C:\ant.
In dieses Verzeichnis entpacken sie das **Ant**. Nach dem es entpackt ist gehen sie in das Verzeichnis `C:\ant\jakarta-ant-1.4.1` und Kopieren die beiden Verzeichnisse bin und lib in den Verzeichnis `c:\ant`.
Als nächstes Klicken sie
`START>SYSTEMSTEUERUNG>SYSTEM>ERWITERT>UMGEBUNGSVARIABLE`

Jetzt definieren Sie neue Systemvariablen:
`ANT_HOME=C:\ant`
`JAVA_HOME= C:\j2sdk1.4.0_01`
Zusätzlich zum `PATH=.....; C:\j2sdk1.4.0_01\bin; c:\ant\bin`
Jetzt können alle Eingaben Übernehmen und Ok. Klicken.
Als nächstes kopieren (entpacken) Sie Das JaZe Verzeichnis(ZIP) von der CD-ROM auf die Festplatte. Starten Sie die DOS-Eingabeaufforderung und gehen Sie in das Verzeichnis `c:\Jaze.`
Dann tippen Sie das Kommando `C:\Jaze> ant.` Jetzt wird alles kompiliert.

Als nächstes Kopieren Sie die Dateien `dropall_mysql` und `create_mysql` aus dem Verzeichnis
`C:\JaZe\source\sql` ins `c:\` , dann gehen Sie in die DOS-Eingabeaufforderung in das Verzeichnis `C:\mysql\bin` und geben Sie das kommando- `mysql.exe` ein. Um die MySQL-Konsole zu starten.
In der MySQL-Konsole führen Sie den Befehl `create database oose` aus. Dann melden sie sich ab mit dem Kommando exit.
Als nächstes geben sie in der Kommandozeile folgende Zeilen ein:

```
mysql < dropall_mysql.sql          (Enter)
mysql < create_mysql               (Enter)
```

Wenn keine Fehlermeldung zu sehen war, dann war alles richtig.

Jetzt müssen Sie noch die Datei `jaze.war`, die sich im Verzeichnis `C:\JaZe\release` befindet in das Verzeichnis `C:\jakarta-tomcat-4.0.3-LE-jdk14\webapps` kopieren.
Die Datei xmlParserAPIs, die sich im Verzeichnis `C:\JaZe\lib` befindet müssen sie ins Verzeichnis `C:\jakarta-tomcat-4.0.3-LE-jdk14\lib` kopieren.

Alles was Sie noch machen müssen:
Reihenfolge ist Wichtig
Die MySQL Datenbank starten mit c:\mysql\bin\mysqld-max -standalone in der DOS-Eingabeaufforderung.
Dann die Datei runwithdatabase.bat die im Verzeichnis `C:\JaZe\release` liegt.
Als nächstes den TomCat Server starten mit der Datei startup.bat die im Verzeichnis
`C:\jakarta-tomcat-4.0.3-LE-jdk14\bin` liegt und zuletzt den Internetexplorer starten und in die URL http://localhost:8080/jaze eingeben.
Danach sollte der Login von JaZe zusehen sein.
Nach dem Sie sich Eingeloggt haben, können Sie mit ihre Arbeit anfangen.

8.2 RMI „Step by Step"

Der Remote Zugriff wurde bereits im Abschnitt 6. „Remote Method Invocation" beschrieben.
Hier möchten ich mich mit diesem Thema ausführlicher befassen und eine Art Hilfestellung anbieten.

Man beginnt damit ein **Interface** zu definieren, das vom Interface ***java.rmi.Remote*** abgeleitet ist. Dieses Interface definiert die exportierten Methoden. Die das entfernte Objekt implementiert (also der Client). In der Deklaration jeder Methode in diesem Interface muss ***java.rmi.RemotException*** , als mögliche Ausnahme deklarieren, weil beim Aufruf entfernter Methoden über das Netzwerk eine ganze Menge schief gehen kann.

Definieren einer Subklasse von ***java.rmi.server.UnicastRemoteObject*** die ihr eigenes Remote-Interface implementiert. Diese Klasse repräsentiert das entfernte Objekt (Server-Objekt)

Ein Programm schreiben, das eine Instanz des entfernten Objekts erzeugt. Dieses Objekt wird exportiert, also für den Client verfügbar gemacht, indem es bei einem Registrierungsdienst angemeldet wird. Dies macht man mit der Klasse ***java.rmi.Naming*** und dem Programm *rmiregistry*.

Dann wird das Server-Programm kompiliert und mit *rmic* wird der *Stub- und Skelett-Code* für das entfernte Objekt generiert. Im RMI-Modell kommunizieren Client und der Server **nicht direkt**. Auf der Client-Seite wird eine Instanz einer Stub-Klasse verwendet, die für das entfernt Objekt steht. Wenn der Client eine entfernte Methode aufruft, dann wird in Wirklichkeit eine Methode dieses Stub-Objekts aufgerufen. Dieses leitet die Anfrage an den Skeleton des Servers weiter, welches die Anfrage übersetzt...

Jetzt wird ein Client-Programm geschrieben, dass das vom Server exportierte entfernte Objekt benutzt. Dazu muss der Client sich zunächst eine Referenz auf das entfernte Objekt besorgen (***lookup***). Dies geschieht mit der Klasse Naming, mit der ein Objekt anhand seines Namens ermittelt werden kann. Der Name ist

normalerweise eine URL mit dem Suffix rmi: Die zurückgegebene Referenz ist eine Instanz des Remote-Interface für das Objekt(oder genauer ein Stub-Objekt für das entfernte Objekt). Wenn der Client sich dieses entfernte Objekt besorgt hat, kann er daran genauso Methoden aufrufen, wie er das an einem lokalen Objekt tun würde.

RMI-Client-Anwendungen müssen einen Sicherheitsmanager einsetzen (siehe unten „Sicherheitsmanager"). Das ist ein Sicherheitsmechanismus, der das Programm gegen Viren im Stub-Code schützt. Die Klasse ***RMISecuritymanager*** ist dafür geeignet.

8.2.1 Serverobjekte ausfindig machen

Um auf ein Remote-Objekt zuzugreifen, das auf dem Server existiert, braucht der Client ein lokales Stub-Objekt. Das gebräuchlichste Verfahren ist es, eine Remote-Methode eines anderen Serverobjekts aufzurufen und ein Stub-Objekt als Rückgabewert zu erhalten.

Ein Serverprogramm registriert Objekte mit den Bootstrap-Registrierungsdienst und der Client ruft Stubs zu diesen Objekten ab. Ein Serverobjekt registriert man, indem man dem Bootstrap-Registrierungsdienst eine Referenz auf das Objekt und einen Namen übergibt. Der Name ist eine Zeichenfolge, die eindeutig ist:

```
//Server
JaZEImpl obj = JaZeImpl(“JaZe Server“);
Naming.bind(“server“,obj);
```

Der Client-Code holt einen Stub, um auf das Serverobjekt zuzugreifen, indemer den Servernamen und den Objektnamen in der folgenden Weise spezifiziert:

```
//Client
ProjectService       projectService       =       (ProjectService)
Naming.lookup(“rmi://localhost/” + "ps");
```

8.2.2 Server starten

Um den Server unter einer Windows-Umgebung starten zu können, muss folgende Anweisung

Start rmiregistry

an einer DOS-Eingabeaufforedrung auszuführen

Jetzt kann man den Server mit

Start java JaZeServer

8.2.3 Sicherheitsmanager

Client Programme, die RMI verwenden, sollten einen Sicherheitsmanager installieren, um die Aktivitäten der dynamisch geladenen Stubs zu kontrollieren. RMISecurityManager ist ein derartiger Sicherheitsmanager, den man mit der Anweisung

System.setSecurityManager(new RMISecurityManager());

installiert.

Beim **SecurityManager** handelt es sich um eine Klasse, die kontrolliert, ob eine bestimmte Operation erlaubt ist.

Die Richtliniendateien (dort werden die Berechtigungen festgelegt und vom SecurityManager abgefragt bzw. geprüft) sind an den zwei folgenden Standorten wieder zu finden:

- die Datei *java.policy* im Basisverzeichnis der Java-Plattform(***lib**Security**java.policy***)**
- die Datei **.**java.policy im Basisverzeichnis des Users(***user.home*** Verzeichnis)

Falls die Richtliniendateien explizit angegeben werden sollen, kann man die Application folgendermaßen starten :

a) java –Djava.security.policy=Application.policy Application

Dieser Aufruf ist aber nur eine Ergänzung zur eigentlichen Richtliniendatei!!!

b) java –Djava.security.policy==Application.policy Application

Es wird hier nicht wie bei a) die Datei *Application.policy* zu den anderen Richtliniendateien hinzugefügt, sondern hier verwendet die Anwendung ausschließlich die angegebene Richtliniendatei und ignoriert die Standartrichtliniendateien.

c) java –Djava.security.manager –Djava.security.policy=Application.policy

Hier wird beim starten der Anwendung ein eigen vorgegebene policy-Datei festgelegt, wo dann noch explizit ein Security-Manager installiert wird, falls dies nicht schon im Quellcode gemacht wurde.

Falls die Richtliniendatei implizit angegeben werden soll, installiert man mit ***System.setSecurityManager(new SecurityManager());*** den Manager, der dann mit der policy-Datei den jeweiligen Schutz bietet.

POLICY-Datei kann man am besten mit dem **policytool.exe** unter jdk erstellen.

8.2.3.1 Policy-Einträge:

- **Policy** File: Angabe der URL, wo die policy-Datei abgelegt werden soll (standardmäßig C:\Windows\.java.policy)
- **Keystore**: mit der URL der policy-Datei ein Schlüsselring zugeordnet, aus dem die Zertifikate geholt werden, die zum Überprüfen der digitalen Signaturen von Java-Code notwendig sind *(**file:/D:/Test/mykeystore, JKS**)*
- **CodeBase**: ist ein optionaler URL-Eintrag, der die Aufenthalt des zu generierenden Code angibt (***file:/D:/Apllication/code/)***

- **SignedBy:** ist ein optionaler Eintrag, der den „alias-Namen“ des Schlüsselring spezifiziert, um den Urheber zu erwähnen, dessen Private Key zum unterzeichnen des Codes genutzt wurde.

Bemerkung zu URL-Angaben:

a) „/“ alle Klassen in diesem Verzeichnis

b) „/*“ alle Dateien in diesem Verzeichnis

c) **c)** „/-„ alle Dateien in diesem und drunter liegenden Verzeichnissen.

In der „Service Architecture“ wurden bereits die einzelnen Schichten der Architektur erläutert. Der hier ausführlich beschriebene Remote Zugriff erfolgt im Interaktionsservice, der es ermöglicht die Dienste des JaZe auf dem Server zu nutzen. In diesem Fall sind es die Fachlichen Services, die unser fachliches Wissen des Anwendungsbereichs zusammenfassen und den Benutzer über die Benutzungsschnittstelle nach außen angeboten werden.

9. Index