Studienjahresarbeit

Thema:
Studie zum Entwurf und zur Implementierung von Online Dienstleistungen auf der Basis der Java 2 Enterprise Edition

Verfasser:
Thorsten Strufe
Studiengang Informatik

Hochschullehrer:
Prof. Dr.-Ing. habil. Dietrich Reschke
Betreuer:
Dipl. Inf. Jörg Deutschmann
Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Institut für Praktische Informatik und Medieninformatik
Fachgebiet Telematik

Inhaltsverzeichnis

Einleitung

Für das Lehr- und Informationssystem (LIS) der TU Ilmenau werden in der vorliegenden Studienarbeit eine Reihe von Online Dienstleistungen entworfen und implementiert.
Als installierbare Anwendung soll eine Lehrveranstaltungsverwaltung entstehen, die sich aus einer Reihe zu realisierender Online Services zusammensetzt. Das Hauptaugenmerk soll zum einen auf eine Nutzerverwaltung gelegt werden, die als Dienstleistungen eine Registrierung im System und eine Verwaltung von Studienergebnissen umfasst. Zum anderen sollen Services zur Verwaltung der Veranstaltungen entwickelt werden. Hierzu gehört eine Oberfläche für die Bereitstellung von Lehrveranstaltungsangeboten durch Mitarbeiter, genauso wie eine Möglichkeit zur Einschreibung in eben diese Veranstaltungen für die Studierenden.
Schwerpunkt der Arbeit soll eine Technologiestudie der Java 2 Enterprise Edition (J2EE) sein. Es geht also vielmehr darum die Möglichkeiten und auftretende Probleme dieser Technologie auszuloten, als eine schön gestaltete Applikation für die Nutzung zu programmieren.

Im ersten Teil wird erläutert, um welche Anwendungsumgebung es bei den gewählten Online Dienstleistungen geht und wie sich diese in das LIS einordnen lassen.
Der zweite Teil der Arbeit befasst sich mit einer näheren Betrachtung der J2EE als Technologie, sowie einer Begründung, warum diese als Basis für die vorliegende Anwendung gewählt wurde.
Zum Schluss wird der komplette Ablauf des Arbeitsprozesses dargestellt. Zu Beginn steht die Ermittlung der Dienstleistungsbedürfnisse für die potenziellen Nutzer. Daraus erwächst die Erstellung funktionaler Kriterien, aus denen sich der Entwurf der notwendigen Komponenten ableitet.
Bei der Umsetzung des Entwurfes in eine lauffähige Applikation kam es zu unvorhergesehenen Schwierigkeiten. Es war nicht möglich die Komponenten in den gegebenen Servern zu installieren und in einer gemeinsamen Applikation zum Laufen zu bringen. Diese Probleme ließen sich nur durch eine Vereinfachung der Applikation oder die Wahl eines anderen Applikationsservers umgehen, nicht aber beseitigen.

Das Lehr- und Informationssystem

Das LIS ist ein Projekt an der Technischen Universität Ilmenau. Zielsetzung ist die Entwicklung eines aus mehreren Komponenten bestehenden Systems. Dieses soll zunächst die administrativen Vorgänge einer Bildungs- und Lehranstalt online und multimedial unterstützen. Im Umfeld der Universitäten und Lehrinstitutionen gibt es bereits eine Reihe unterschiedlicher Projekte, die unter dem Begriff „Virtuelle Universität“ zusammengefasst werden können.

Die Virtuelle Universität

Zu dem Begriff „Virtuelle Universität“ gibt es keine eindeutige Definition. Die vorhandenen Projekte lassen sich allerdings recht gut in zwei erkennbare Entwicklungsrichtungen einordnen. Die eine versteht darunter Internetgestützte Lehr- und Lernsysteme, die vor allem dazu genutzt werden sollen, Lehrinhalte auch Online zu präsentieren (siehe Abb. 1). Hier geht es bei kleineren Projekten in erster Linie darum, einzelne Teile aus Lehrveranstaltungen über das Internet verfügbar zu machen und kleine Übungssequenzen zur Vertiefung des Lernstoffes, in Form von Texten, Grafiken oder Applets anzubieten.

In größeren Projekten ist der Versuch erkennbar, das an Lehrinstituten vorhandene Know-how einer breiteren Basis von Kunden, in diesem Fall Lernenden anzubieten. Der zentrale Antrieb bei diesen ist es, in naher Zukunft weltweit Telelearning Kurse verkaufen zu können und so zusätzliche Geldquellen zu erschließen. (Der Aus- und Weiterbildungsmarkt wird bei der in allen Bereichen immer schneller werdenden Entwicklung, und der daraus resultierenden Suche nach immer spezialisierteren und kompetenteren Mitarbeitern in absehbarer Zeit zu einem der größten Märkte überhaupt werden)

Kritisch betrachtet macht dieser Ansatz allerdings erst dann ernsthaft Sinn, wenn zwei Umstände zusammentreffen. Zum einen bedarf es eines relativ großen Umfanges an Inhalten, die über längere Zeit modularisiert und standardisiert zur Verfügung stehen – und die über diesen Zeitraum nicht veralten dürfen. Zum anderen muss eine große Anzahl Interessenten vorhanden sein, die bereit ist für diese Inhalte auch Geld auszugeben.

Diese Einschränkung liegt darin begründet, dass die Bereitstellung der Lehrinhalte immens aufwendig ist, weil sie in einem sinnvollen Format vorliegen müssen. Dieses Format müsste sich der ganzen Möglichkeiten des Internets bedienen, also multimedial, interaktiv und aktuell sein.

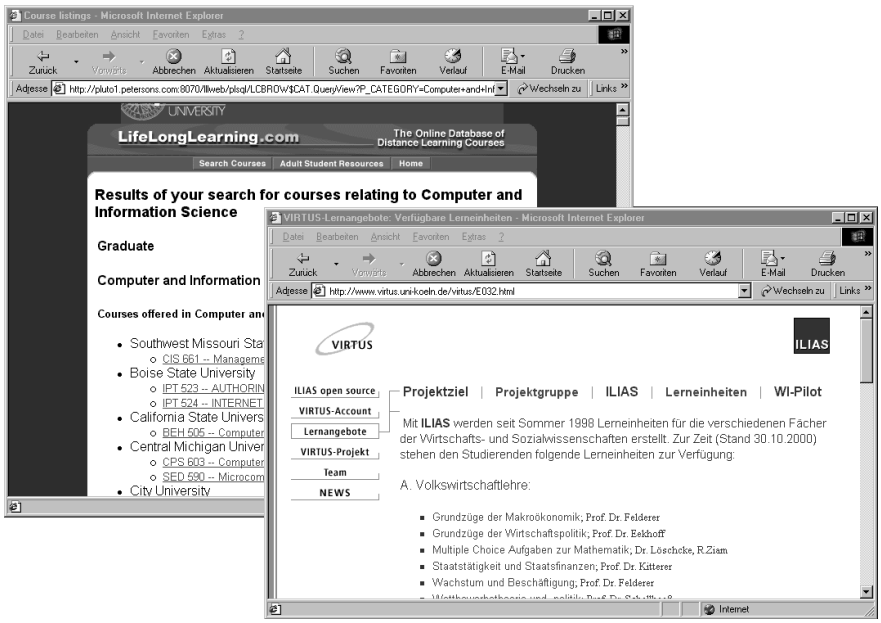


Abbildung 1: Beispiele für „Lehrsysteme“ im Netz[1] (Petersons Distance Learning und Virtus)

So bestehen denn auch die meisten existierenden Systeme aus nicht viel mehr als einigen eingefärbten und mit Bildern versehenen Mitschriften zu Vorlesungen.

Die andere Hauptströmung versteht unter dem Begriff „Virtuelle Universität“ ein für bestehende Institutionen unterstützendes System, welches allgemeine administrative Aufgaben und Datenverarbeitung erleichtern und integrieren soll.
Er steht hier also nicht für eine „Online Universität“, oder eine Lehrinstitution im Internet, sondern soll vielmehr zum Ausdruck bringen, dass die Abläufe hier nicht mehr von Hand auf Papier, sondern digital über ein Rechnernetz organisiert sind. Der Antrieb hinter diesen Projekten ist es also nicht, Lehre im Netz zu betreiben, sondern ein effizientes Informationssystem bereit zu stellen. Dieses soll dazu dienen, dass sich Studierende oder Interessenten über die Universität und ihre Angebote informieren können. Sie sollen sich unter Umständen auch an- oder rückmelden, in Lehrveranstaltungen einschreiben und ihre bisherigen Leistungen einsehen können.

[1] Vgl. Div: /Distance Learning/,
Div: /Virtus/

Mitarbeitern und Professoren soll damit die Möglichkeit gegeben werden, Lehrveranstaltungen anzubieten, zu verwalten und zusätzliche Information dazu bereitzustellen. Und nicht zuletzt sollen die Sekretariate bei der Verwaltung der Personendaten und Leistungen sowie bei der Raum- und Zeitplanung, der Zeugnis-, Schein- und Ergebnislistenerstellung und ähnlichen Aufgaben unterstützt werden.

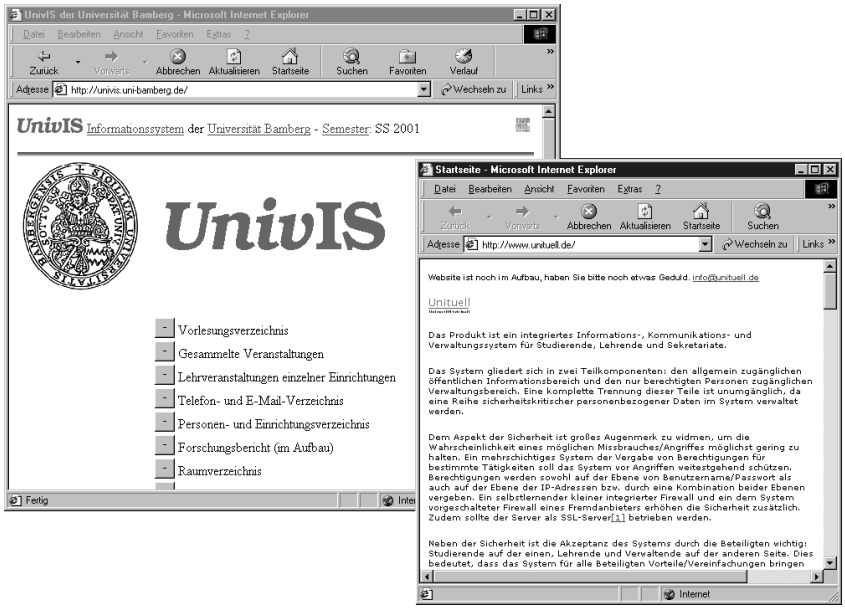


Abbildung 2: Beispiele für Systeme zur administrativen Unterstützung[2] (UnivIS und UniTuell)

Genau betrachtet hat es also eine eher unterstützende Funktion, die Kommunikationswege vereinfachen sowie Medienbrüche vermeiden helfen soll.

Das Lehr- und Informationssystem der Technischen Universität Ilmenau ist zunächst der zweiten Klasse von Systemen zuzurechnen.

Als ein erster Versuch an der TU Ilmenau in dieser Richtung etwas zu entwickeln kann ein System der technischen Informatik angesehen werden. Hier wurde mit PHP Scripten eine relativ einfache Anwendung entwickelt, mittels derer sich Studenten online in Praktika einschreiben können. Allerdings ist diese Applikation auf genau diese Praktikumseinschreibung spezialisiert und es wäre wohl eher schwierig, sie für umfangreichere Aufgaben einzusetzen.

[2] Vgl : Div : /UnivIS/,
Div : /UniTuell/

Die Lehrveranstaltungs-Verwaltung

Als Resultat der vorliegenden Arbeit soll es Mitarbeitern und Studenten ermöglicht werden, einen Großteil der beim Studium anfallenden organisatorischen Tätigkeiten digital erledigen zu können. So sollen vor allem die Bereitstellung, Einschreibung, Bearbeitung, Benotung von Lehrveranstaltungen und das Nachschlagen von erreichten Ergebnissen unterstützt werden.

Die vornehmlich bereitzustellenden Dienstleistungen können hier in zwei übergeordnete Kategorien eingeteilt werden. Zum einen die Sicht der wissenschaftlichen Mitarbeiter, zum anderen die der Studenten.

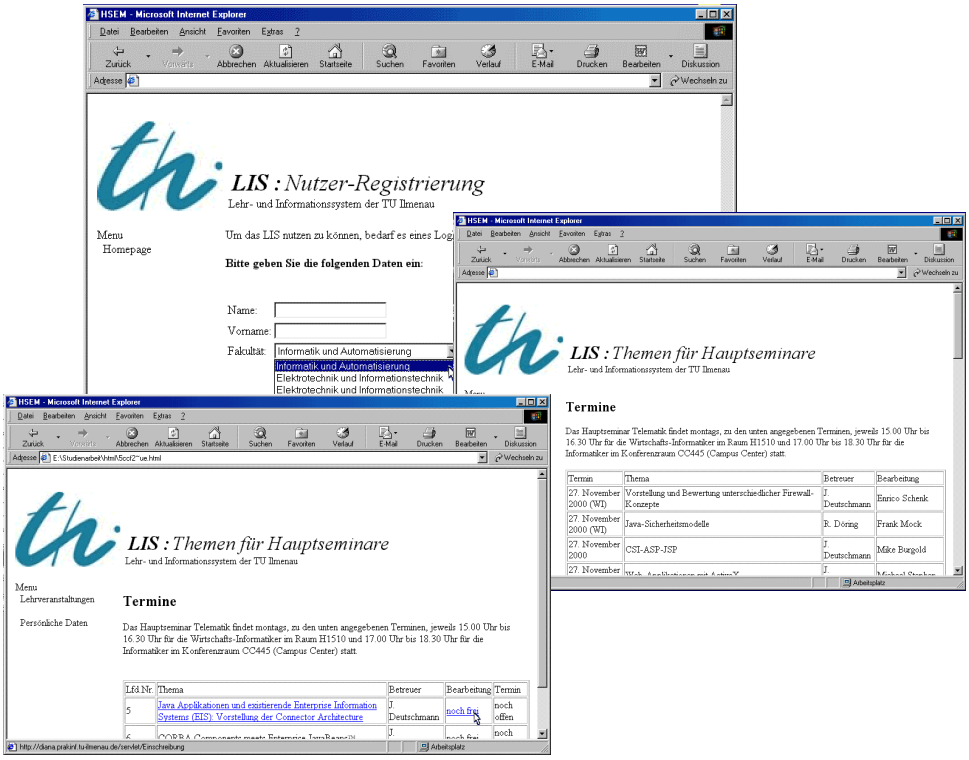


Abbildung 3: Oberfläche der Lehrveranstaltungs Verwaltung

So ist es für den Mitarbeiter interessant, die Möglichkeit zu erhalten, Themen zu Projekten nicht nur auszuhängen, sondern darüber hinaus auch einfach ins Netz zu stellen. Weiterhin soll es dem Mitarbeiter möglich sein nachzuschlagen, welcher Studierende sich zu seinen Projekten eingeschrieben hat. Schlussendlich muss nach abgeschlossener Leistung, etwa eines Hauptseminars oder einer Studienarbeit, diese benotet werden, was im vorliegenden Fall ebenfalls online geschehen soll.

Aus Sicht eines Studenten ist es zunächst interessant, einen Überblick über die angebotenen Themen für die von ihm zu erbringenden Leistungen sowie die Lehrveranstaltungen zu erhalten. Hat er ein Thema gefunden, welches ihn interessiert – und das noch frei ist, so soll er sich dafür einschreiben können, um das Thema für sich zu beanspruchen (oder im Falle von Vorlesungen und Praktika sich für diese anzumelden). Nachdem eine Reihe Leistungen erbracht und ihre Benotung erfolgt ist, muss es dem Studenten natürlich auch möglich sein, diese nachschlagen zu können.
Über diese anwendungsbezogenen Dienstleistungen hinaus wird eine globale Authentifizierung von Nöten sein, um sicher zu stellen, dass jeder Student nur seine eigenen Ergebnisse zu Gesicht bekommt – und nur Mitarbeiter eines Fachgebietes Themen für dieses eintragen – und nur die von ihnen betreuten Leistungen benoten können. Hieraus entsteht der Bedarf nach einer Nutzerverwaltung, die das Registrieren für das System, Einsehen – und Ändern von nutzerbezogenen Daten ermöglichen soll.

Java 2 Enterprise Edition

Um Online Services wie das LIS im Netz bereitzustellen, also rein statische HTML Seiten interaktiv zu machen gibt es eine Reihe vorhandener Lösungsmöglichkeiten

Warum Java?

Alternative Implementierungstechnologien

Ein Grossteil der produktiven Online Dienstleister verlässt sich auf Technologien wie PHP, Perl, Server Side Includes, mit denen sich schnell relativ einfache Lösungen realisieren lassen. Die Bearbeitung von Nutzeranfragen, Datenbankzugriffe etc. sind hiermit zweifellos relativ effizient möglich. Kommt es allerdings zur Entwicklung eines umfangreicheren oder komplexeren Systems wird die Administration und Anpassung bei der Nutzung dieser Scriptsprachen sehr schnell sehr unübersichtlich. Spätestens, wenn über eine Wiederverwendbarkeit von Systemkomponenten oder eine Integration mit bestehenden Enterprise Information Systemen (EIS) nachgedacht wird, bietet es sich an eine objektorientierte Programmiersprache mit unter Umständen benötigter Middleware-Software zu nutzen.

In der Vergangenheit erschien das eher schwierig. Um über einen Webserver auf hochsprachige Komponenten zugreifen zu können, bedurfte es des Common Gateway Interfaces (CGI), einer ineffizienten Schnittstelle. Bei jeder neuen Anfrage über das CGI wird auf dem Server ein extra Prozess gestartet, der sie bearbeitet und eine Antwort an den Webserver zur Ausgabe zurückgibt. Beim CGI können Datenbankverbindungen nicht über mehrere Anfragen hinaus aufrecht erhalten werden, eine Nutung von Verbindungspools ist dementsprechend auch nicht möglich. Wird die Tatsache berücksichtigt, dass Verbindungsaufbau und –abbau über 90% der Zeit einer Anfrage in Anspruch nehmen, rücken die Nachteile noch deutlicher in den Vordergrund.

Über diese Ineffizienz hinaus musste sich der Programmierer um die Kommunikation der entwickelten Komponenten selber kümmern, was ein grundlegendes Verständnis von Remote Procedure Calls (RPC), Remote Method Invocation (RMI), Corba oder vergleichbaren Architekturen voraussetzte oder nur völlig rudimentär umgesetzt werden konnte. Bei der am Markt herrschenden Knappheit an qualifizierten Anwendungsentwicklern kam es so zu einer verstärkten Nutzung der einfacheren Technologien wie Perl oder PHP, für die es Server Plugins und ein schon breites Angebot fertiger Skripte gibt. Firmen, die Webanwendungen entwickelten, brauchten so zusätzlich zu den Webdesignern nur noch jemanden, der einige Erfahrung mit diesen Sprachen hatte.

Dabei liegen die Vorteile der Nutzung einer Programmiersprache auf der Hand. So gibt es für diese eine große Anzahl Entwicklungsumgebungen (IDE's), was nicht nur die Entwicklung, sondern auch die Erweiterung und Wartung deutlich vereinfacht. Darüber hinaus sind bei dem Einsatz von Scripten Präsentationslogik und Anwendungslogik nicht zu trennen. Bei einfachen Änderungen an einer der beiden Welten bedarf es einer kompletten Neuentwicklung des Services. Anderenfalls besteht das Risiko dass schon durch kleine Änderungen an der Darstellung die Applikationslogik in Mitleidenschaft gezogen wird.

Wird darüber hinaus noch eine auf die Verteilung spezialisierte Architektur verwendet, so kommt hinzu, dass Aufgaben, welche auf Systemlevel anfallen, in der Regel schon vom Anbieter dieser Technologien implementiert sind. Connection Pooling zur effizienten Datenbankanbindung, Sicherheitsroutinen und die Kommunikation der Komponenten über Netzwerke können so genutzt – und müssen nicht selbst umgesetzt werden.

Erste Schritte in die Richtung des Einsatzes von objektorientierten Programmiersprachen für Webapplikationen hat es von Microsoft mit der Einführung der Active Server Pages (ASP) und ISAPI (Internet Server Application Programming Interface) sowie von Sun mit den Java Server Pages (JSP) und Servlets gegeben.
ASP und JSP als seitenorientierte Technologien ermöglichen die Einbettung von Visual Basic- respektive Java- Programmstücken in ansonsten traditionell geschriebene HTML-Seiten.
Beim ISAPI werden einzelne Webapplikationen in C++ geschrieben und als DLL kompiliert. Der Webserver ruft dann schlicht deren Funktionen mit Formularwerten als Parameter auf, was zu einer sehr hohen Ausführungsgeschwindigkeit führt. Beide Technologien von Microsoft sind allerdings proprietäre und plattformgebundene Lösungen.

Java als Technologie

Anders als bei den ISAPI-DLL, deren Funktionen auf der Serverhardware direkt kompiliert werden, laufen Servlets in eigenständigen Java Virtual Machines. Diese Architektur führt zu Nachteilen bei der Performance, da der Code ständig interpretiert werden muss, und der Stabilität, da die zur Verfügung stehenden kostenlosen Server (etwa der Tomcat) unter sehr hoher Nutzerlast schon mal zusammenbrechen. Sie hat jedoch den Vorteil, dass die Servlets auf allen Plattformen, für die eine Virtual Machine existiert (also quasi allen vorhandenen) laufen und daher leicht portierbar sind. Es existieren auch durchaus stabile kommerzielle Server. Darüber hinaus spielt die vergleichsweise leicht zu erlernende und zur Zeit hohe Popularität Sprache Java sicher ebenfalls eine große Rolle bei der schnellen Verbreitung dieser Technologien. Unterstützend wirkt sich der Umstand aus, dass sowohl Serversoftware, als auch Entwicklungsumgebungen und Dokumentation frei verfügbar sind
Alle soweit angeführten Technologien haben den Nachteil dass sich die Anwendungsentwickler selbst um die Kommunikation zwischen dem Applikations-server, den Datenbanken und anderen Komponenten kümmern.
Vorhandene Middleware-Konzepte, die dieses ermöglichen, umfassen die RPC-Architektur, Corba, Java RMI oder COM+. Allerdings sind all diese Möglichkeiten kompliziert und setzen wiederum den Einsatz erfahrener Entwickler voraus.

Sowohl Sun als auch Microsoft bieten nun einfachere Lösungsmöglichkeiten, den kommerziellen MTS (Microsoft Transaction Server), oder aber die von Sun bereitgestellte frei verfügbare J2EE Spezifikation.

Die Enterprise Edition

Bei der Java 2 Enterprise Edition handelt es sich nicht um ein vermarktetes Produkt, sondern vielmehr um eine Spezifikation, nach der sich Anbieter von Applikations-Servern bei der Entwicklung ihrer Produkte richten können. Sun stellt zudem eine Referenz Implementierung bereit, auf der Entwickler ihre Komponenten bezüglich der Kompatibilität zur Spezifikation testen können.
Die Spezifikation umfasst neben der unten beschriebenen Enterprise JavaBeans als zentraler Komponententechnologie eine Reihe anderer Java Technologien (siehe Abb. 4) und verpflichtet den Application Server Provider[3] dazu, bestimmte Aufgaben bereits im Server zu realisieren. Hierzu gehören die Kommunikation zwischen Komponenten mittels RMI oder Corba IIOP, das Auffinden gesuchter Komponenten mittels JNDI und eines Verzeichnisdienstes, das Bereitstellen von Transaktionen etc.

Die für die vorliegende Arbeit benötigten Technologien aus der J2EE sollen hier kurz vorgestellt werden.

Enterprise JavaBeans

Enterprise JavaBeans (EJB) sind der zentrale Bestandteil der J2EE. Sie[4] sind alleinstehend installierbare, serverseitige Softwarekomponenten, mit denen in der Regel die Aufgaben der Applikationsschicht in einer verteilten mehrschichtigen („Multitier“) Anwendung implementiert werden sollen. Als Softwarekomponenten müssen sie nicht aus einzelnen Objekten bestehen, sondern können mehrere Objekte enthalten. Jede EJB implementiert zwei Interfaces, die vom Entwickler definiert werden müssen. Es handelt sich hierbei um ein Home- und ein Remote- Interface. Das Home- Interface umfasst alle Methoden, die über den Container aufgerufen werden, also in etwa Finder- oder create()-Methoden.

[3] Vgl. Matena /EJBSpec/
[4] Vgl. Roman /Mastering/ 51-59

Das Remote- Interface hingegen enthält alle Methoden, welche die Applikationslogik der Bean ausmachen.
Der Client einer Enterprise Bean kann nur über diese Interfaces mit ihr kommunizieren und erhält niemals direkten Zugriff auf ihre Methoden, wobei als Client nicht nur außerhalb des EJB Containers befindliche Applets, Servlets oder Corba Komponenten in Frage kommen, sondern auch andere Beans des gleichen Containers (oder eines anderen).

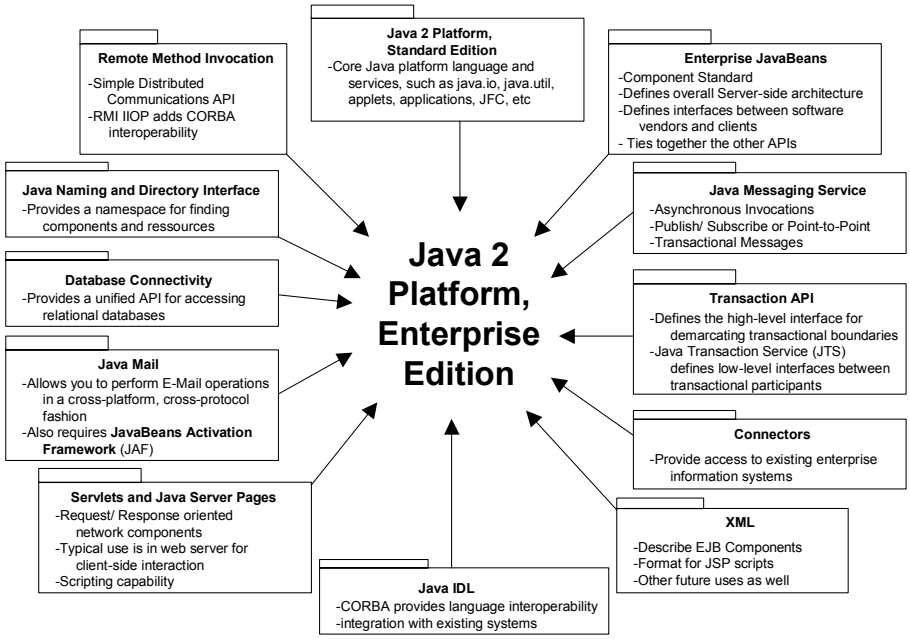


Abbildung 4: die Java2 Enterprise Edition (nach Perrone und Chaganti: /Enterprise Systems/)

Zu allen Enterprise Beans muss der Entwickler über die beiden Interfaces hinaus einen „Deployment Descriptor“ erstellen. Hierbei handelt es sich um eine XML Datei, in der alle Installationsbedingungen, wie Pfade zu anderen Komponenten, JNDI Namen, Sicherheitskriterien und Ähnliches beschrieben werden, um die Bean bei der Installation in die vorliegende Architektur einbinden zu können.
Nun könnte der Eindruck entstehen, dass bei dieser Vielzahl an Dateien die für jede Komponente erstellt werden muss, die Entwicklung sehr umständlich würde. Remote- und Home-Interface werden allerdings bei der Nutzung eines Entwicklungswerkzeuges zumeist automatisch erstellt. Darüber hinaus bieten die meisten J2EE Server ein

Installationswerkzeug an, welches die Erstellung des Deployment Descriptors automatisiert.
Es gibt (in der aktuellen Spezifikation 1.1) grundsätzlich zwei unterschiedliche Arten Enterprise JavaBeans: die Session- und die Entity Beans. In der Version 2.0 kommen MessageDriven Beans hinzu, die über das JMS asynchrone Kommunikation ermöglichen sollen[5] .

SessionBeans stellen Geschäftsprozesse dar. Im Zusammenspiel bilden sie die gesamte Applikationslogik. Dabei können sie zwar auf unterliegende Daten zugreifen, diese aber nicht repräsentieren. Die Lebensdauer einer Session Bean ist, wie der Name andeutet, auf eine Nutzersession begrenzt. Es gibt keine Möglichkeit, sie über diese Dauer hinaus persistent zu machen. Bei Session Beans wird bezüglich der Lebensdauer zwischen zwei Arten unterschieden.
Zum einen gibt es *Stateless Session Beans*, die keinen Zustand annehmen können. Sie führen, einmal vom Client aktiviert, nur einen einzigen Prozess aus, beenden die Verbindung und werden vom Container zurückgesetzt oder deaktiviert.
Zum Zweiten gibt es *Stateful Session Beans*, die einen Zustand annehmen können. Mit ihnen können Prozesse realisiert werden, die mehrerer Methodenaufrufe oder Transaktionen bedürfen, da sie mit dem Client in Kontakt bleiben und durch ihren Zustand ein beschränktes Vorwissen haben.

EntityBeans repräsentieren alle im System vorkommenden „real-world-“ Objekte[6] und sind für die Persistierung der Attribute derselben zuständig. Sie bilden eine Objektsicht auf die Geschäftsdaten einer unterliegenden Datenbank. Wichtig ist allerdings, dass sie nicht als eigenständige Kopien von Objekten aus der unterliegenden Datenschicht angesehen werden, sondern eher als „Sicht“ auf dieselben, da der Container das Updaten bei Änderungen der Attribute übernimmt.
Bei der Speicherung der Attribute von Entity Beans gibt es zwei Implementierungsmöglichkeiten. Einerseits kann sich der Application Component Provider[7] dazu entscheiden, die benötigten SQL Statements zur Erzeugung und Abfrage der Tabellen direkt in der Bean zu implementieren. Diese Möglichkeit wird *Bean*

[5] Vgl. Haefel /EJB 2.0/
[6] Vgl. Haefel /EJB/ 23-24
[7] Vgl. Matena & Hapner /EJBSpec/

Managed Persistence genannt und wird bis zum aktuellen Zeitpunkt als zuverlässiger eingeschätzt.

Andererseits besteht die Möglichkeit, dem Server die Generierung und Durchführung der Datenbankanfragen zu überlassen. Die Statements werden dann bei der Installation der Bean in den Server durch den Application Deployer angegeben und können so direkt an das unterliegende Datenbanksystem angepasst werden. Dieses Verfahren wird als *Container Managed Persistence* bezeichnet.[8]

Um die zur EntityBean gehörende Reihe in der Datenbank finden zu können, werden in einer Primarykey-Klasse die Schlüsselattribute (und bei Bedarf gesonderte Hash- und Vergleichsmethoden) beschrieben.

Naming and Directory Interface

Beim Java Naming and Directory Interface (JNDI) handelt es sich um eine Schnittstelle, über die es mit Java möglich ist, auf eine Vielzahl von Namens- und Verzeichnisdiensten zuzugreifen. Die zentrale Aufgabe der JNDI in der Entwicklung von Enterprise Applikationen besteht darin, die einzelnen Komponenten in einer verteilten Architektur referenzierbar zu machen. So kann JNDI zum Beispiel dazu verwendet werden, Rechner, Dienste oder Komponenten in einem Netzwerk aufzufinden.

Für Enterprise JavaBeans bedeutet das insbesondere, dass alle implementierten Komponenten bei der Installation in den J2EE Server einen JNDI Namen bekommen, mittels dessen sie von anderen Komponenten oder Klienten aufgefunden und instanziiert werden. Auch die von den Enterprise Beans genutzten Ressourcen wie Datenbanken sollen über ihren JNDI Namen, und nicht über ihren direkten Pfad angesprochen werden. Außerdem besteht die Möglichkeit mittels JNDI das Passwort oder bestimmte Konfigurationen eines Nutzers zu sichern und netzweit verfügbar zu machen.

[8] Für eine ausführliche Beschreibung der Enterprise JavaBeans siehe Dittmar: /Asynchrone Kommunikation/

Database Connectivity

JDBC, sprich Database Connectivity (laut Sun steht die Abkürzung tatsächlich *nicht* für *Java* Database Connectivity), ist ein weiterer Bestandteil der J2EE. Es handelt sich bei JDBC um eine an Microsofts ODBC (**O**pen **D**ata**b**ase **C**onnectivity) angelehnte Schnittstelle, mittels derer Java Klassen auf fast alle relationalen Datenbanken zugreifen können.

Wird JDBC genutzt, übergibt der Anwendungsentwickler als Strings angegebene SQL Statements und erhält das Resultat als Aufzählung oder Kollektion von Datensätzen zurück.

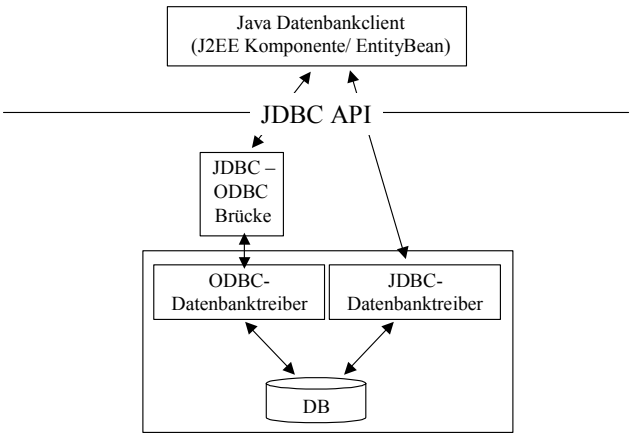


Abbildung 5: JDBC Architektur – zwei Zugriffsmöglichkeiten auf Datenbanken

Mit JDBC kann auf zwei unterschiedliche Weisen auf unterliegende Datenbanken zugegriffen werden (siehe Abb. 5). Zunächst kann auf alle Datenbanken, für die ein Java Connector existiert, direkt zugegriffen werden. Bei Java Connectoren handelt es sich um spezielle Datenbanktreiber, die für die meisten Produkte am Markt entweder durch den Hersteller, oder aber die Entwicklergemeinde bereitgestellt werden. Um auf die wenigen ODBC fähigen Datenbanksysteme zugreifen zu können, für die tatsächlich kein Connector existiert, gibt es als zweite Möglichkeit eine JDBC-ODBC-Brücke. Diese verpackt die JDBC Schnittstelle in ODBC und ermöglicht so, nicht ganz so performant, die Nutzung dieser Datenbanken.

Servlets

Als gebräuchliche Komponenten für die Präsentation enthält die J2EE Spezifikation die Servlets und die Java Server Pages (JSP). Bei Servlets handelt es sich um kommunikationsorientierte Präsentationskomponenten, die in erweiterten Webservern

(wie etwa Tomcat) installiert werden. Sie warten auf http-Anfragen (GET oder POST-Methoden), bearbeiten diese und reagieren mit einer http-Antwort. Während Servlets als richtige Java Klassen programmiert werden müssen, ist es dem Entwickler bei JSP möglich, kurze Java Programmstücke in seine herkömmlich entwickelten HTML Seiten zu integrieren, ohne ganze Klassen für die Seitengenerierung schreiben zu müssen (s.o.). Intern werden JSP aber ebenfalls zu Servlets kompiliert und im Webserver ausgeführt. Sowohl mit Servlets als auch mit JSP können einfache eigenständige Web-Applikationen entwickelt werden. Ihre Rolle in der J2EE ist es aber primär als Schnittstelle zwischen dem Nutzer und den Applikations-Komponenten zu dienen.

Bewertung der J2EE

Die Java 2 Enterprise Edition Spezifikation sichert dem Entwickler alle Möglichkeiten für eine effektive Entwicklung von komplexen, verteilten Anwendungen zu, die dieser benötigt. Durch die offene Architektur wird eine Integration der existierenden proprietären und freien Systeme immens erleichtert und ist für den größten Teil der etablierten Software am Markt schon realisiert. So ist es durch die Existenz der Connectoren und die wohldefinierten Schnittstellen möglich, eine einzelne EJB Komponente oder eine ganze J2EE Applikation horizontal an Corba-, RMI-, DCE-, oder über den Java Messaging Service (JMS) sogar an messageorientierte Middleware-Systeme anzubinden. Vertikal existiert für praktisch jedes erhältliche Datenbank-System auch ein Connector.

Nur die Anbindung an die Microsoft-Welt erwies sich in der Vergangenheit als eher schwierig. Ein Problem welches durch die breite Entwicklergemeinde allerdings bis heute als nahezu behoben angesehen werden kann.

Weitere Vorteile der J2EE sind vor allem bei der Wiederverwendbarkeit der entwickelten Software zu finden. Es besteht die Möglichkeit, eine Komponente oder sogar ein vollständiges System, welches nach der Spezifikation entwickelt wurde, in jedem beliebigen EJB Container zu installieren. System oder Komponente sind somit ohne Änderung auf praktisch allen Plattformen einsetzbar. Fertige Komponenten sollen darüber hinaus in Zukunft als Superklassen in einem Vererbungsprozess dienen können. Dadurch werden sie leicht an eigene Anforderungen anpassbar und müssen nicht immer aufs Neue programmiert werden.

Der Einsatz des Deployment Descriptors macht den Entwickler vollkommen unabhängig von der späteren Architektur des Systems. Rechnernamen, Sicherheitsrollen, Pfade zu anderen Komponenten und so fort brauchen erst zum Zeitpunkt der Installation festgelegt zu werden.
Wie zu erwarten, hat sich im Laufe der letzten zwei Jahre ein reger Markt um die Entwicklung von einerseits J2EE Servern und andererseits fertigen käuflichen und frei verfügbaren EJB Komponenten entwickelt. So können nun ernsthaft „Make- or Buy-" Entscheidung getroffen werden, da die Komponenten, anders als in der Vergangenheit, tatsächlich wiederverwendbar sind.

Entwurf und Implementierung

Entwurfsphase

Ermittlung der benötigten Dienstleistungen

Aus der Beschreibung der zu implementierenden Online Dienstleistungen heraus ergeben sich eine Reihe von Funktionalitäten, die das fertige System aufweisen soll. Zunächst wird ein kurzer Überblick über diese gegeben, bevor es zur genauen Beschreibung der technischen Details kommt.
Geordnet nach den Nutzergruppen, für die sie von Interesse sind, haben die benötigten Funktionalitäten den folgenden Umfang:

Zunächst die für die Mitarbeiter relevanten Dienstleistungen:

Themen eintragen Um überhaupt zu einer Auswahl an belegbaren Themen im System zu kommen, soll es jedem Mitarbeiter möglich sein, Lehrveranstaltungen bzw. Hauptseminare zu eigenen Projekten in seinem Fachgebiet einzutragen. Es ist darauf zu achten, dass die Integrität erhalten bleibt – und die Themen dem richtigen Fachgebiet zugeordnet werden.

Leistungen benoten Der betreuende Mitarbeiter soll nach Abschluss einer Lehrveranstaltung oder eines Projektes dieses auch im System terminieren und die evtl. erlangte Note eintragen. Nach dieser Terminierung soll das Thema aus dem System ausgetragen und nur noch als „Schein" bei Studierendem und Mitarbeiter vorliegen.

Eine mögliche Erweiterung wäre die Nutzung der Mailapp[9] des LIS, um Mitarbeiter nach abgelaufenem „Abschlusstermin“ einer Leistung auf die durchzuführende Terminierung hinzuweisen.

Von Mitarbeitern wie von Studenten genutzte Dienstleistungen umfassen:

Themen durchsuchen Der Interessent soll auf einfache Art und Weise die Themen der Lehrveranstaltungen oder Projekte durchstöbern können. Hierbei ist zu beachten, dass nur die Themen eines einzelnen Fachgebietes zur Anzeige kommen, um zu verhindern dass es zu einer unüberschaubaren Flut an Informationen auf der Seite kommt. Zu diesem Punkt gehört somit auch die Möglichkeit, das anzuzeigende Fachgebiet zu wählen.

Ergebnisnachschlag Der Studierende soll die Möglichkeit haben, seine abgeschlossenen Leistungen zur Anzeige zu bringen. Ebenso soll es dem Mitarbeiter möglich sein eine Liste der Ergebnisse von ihm betreuter Leistungen einzusehen. Die Möglichkeit der nachträglichen Änderung von Ergebnissen ist nicht vorgesehen.

Vom System benötigte Dienstleistungen:

Registrierung Studierende, die noch keine eingetragenen Nutzer des Systems sind, sollen die Möglichkeit haben, sich selbständig als Nutzer zu registrieren. Hierfür werden außer dem Namen eine Reihe persönlicher Daten wie Matrikel, Matrikelnummer, Seminargruppe, Email Adresse etc. aufgenommen, sowie ein Passwort festgelegt und gespeichert.

Die Registrierung der Mitarbeiter soll nicht selbstständig im Netz stattfinden, sondern über einen Administrator. Es gibt keine gute Möglichkeit, die Identität eines Mitarbeiters online zu verifizieren, weshalb diese eher umständliche Lösung preferiert wird.

Login Um den vollen Umfang an Dienstleistungen der Anwendung nutzen zu können, muss sich der Interessent am System anmelden und authentifiziert werden.

[9] Vgl. Dittmar /Asynchrone Kommunikation/ 39

Nutzerprofil einsehen/ ändern Es soll dem Nutzer möglich sein, seine eigenen persönlichen Daten einzusehen und einzelne Felder zu ändern.

Für die Studenten von Interesse ist gegebenenfalls:
Themen zu **belegen** Hat ein Studierender ein Thema gefunden, welches verfügbar ist und ihn interessiert, soll er sich durch einfaches Anklicken dafür einschreiben können. Wünschenswert wäre eine Kontrolle, ob der Studierende schon eine große Anzahl Themen belegt, woraufhin er unter Umständen aufgefordert werden könnte, erst einmal diese zu beenden oder zumindest über die Tatsache informiert werden sollte.

(**Nutzersuche**) Es wäre vorstellbar eine Nutzersuche zu implementieren, bei der etwa ein Studierender die Telefonnummer oder Email Adresse seines betreuenden Mitarbeiters oder, anders herum, ein Mitarbeiter die Email Adresse seines betreuten Studenten nachschlagen könnte.

Beschreibung der Architektur

Die Implementierung kann in drei Schichten unterteilt werden: die Präsentationsschicht, die Applikationsschicht („Business Layer“ [10]) und die Datenschicht.
Die Datenschicht besteht aus einer relationalen Datenbank und einem Verzeichnisdienst. Als Verzeichnisdienst wurde ein LDAP Server (**L**ightweight **D**irectory **A**ccess **P**rotocol) gewählt. Die relationale Datenbank wird mit dem im J2EE Paket enthaltenen Cloudscape realisiert.
Die Applikationsschicht besteht aus einer Reihe von Enterprise JavaBeans, die im J2EE Server in einen EJB Container installiert werden. Hier kommen sowohl SessionBeans, welche die Abläufe realisieren, als auch EntityBeans als Datenobjekte zur Anwendung.
Die Präsentationsschicht besteht aus einer Gruppe von Servlets, die im J2EE Server in einem Web Container, oder auf einem eigenständigen Webserver (Tomcat o.ä.) laufen und die (u.U. über das Netz) auf die Applikationsschicht zugreifen.

Bei der Entwicklung wurde mit Bedacht eine Architektur gewählt, in der der Klient lediglich aus einem im Netz befindlichen Webbrowser besteht. Es wurde also bewusst

[10] Vgl. Roman /Mastering/ 18

darauf verzichtet, ein extra Applet zu schreiben. Um im Anwendungsfall eine möglichst große Akzeptanz des Systems zu erreichen, ist es ausgesprochen wichtig, Studenten und Mitarbeitern ein Werkzeug zur Arbeitserleichterung anzubieten, welches diese sofort nutzen – und auf wohlbekannte Art und Weise bedienen können, ohne erst Client Software aus dem Netz laden – und womöglich installieren zu müssen.
Die wohlbekannte Problematik der schlechten Kompatibilität zwischen unterschiedlichen Java Versionen ist ein weiterer Grund, sich bei der Client Seite auf das stabile, etablierte und in seinen Kernfunktionen gut unterstützte HTML zu verlassen.

Implementierung

Die Implementierung erfolgte in fünf Teilen. Zuerst wurde, nur mit Session- und Entity-Beans, prototypisch eine lauffähige Version der Applikationsschicht entwickelt, welche zur Persistierung lediglich an die relationale Cloudscape Datenbank angebunden war. Die einzelnen Komponenten mussten noch mit passenden Kommandozeilenklienten angesprochen werden, da keine Präsentationsschicht vorgesehen war.
Darauf folgte in vier Schritten die Entwicklung der vollständigen Anwendung. Die erste Aufgabe bestand in der Realisierung der Datenschicht mittels Einrichtung der Cloudscape Datenbank und des OpenLDAP Servers.
Als nächstes wurden die drei EntityBeans implementiert, welche auf die Datenschicht zugreifen. Dieser Schritt erwies sich überraschend als schwierig, da die Integration eines beliebigen LDAP Servers mit der zur Nutzung geforderten Referenz-Implementation von Sun unmöglich zu sein scheint.
Als dritter Schritt folgte die Programmierung der SessionBeans, die wiederum die Applikationslogik enthalten und die EntityBeans als Mittel zur Persistierung der Daten nutzen.
Als letztes blieb das Schreiben der Servlets zur Präsentation und Interaktion mit dem Nutzer.
Für einen guten und detaillierten Überblick zur Applikationsentwicklung mit J2EE im allgemeinen und mit Enterprise JavaBeans im besonderen sei auf [Dittmar: /Asynchrone Kommunikation/], [Pawlan: /Writing Enterprise Applications/] oder auf
[Roman: /Mastering/] verwiesen.

Die Datenschicht

Zur Speicherung der angebotenen Themen, sowie der Nutzer des Systems soll ein Verzeichnisdienst zum Einsatz kommen. Die Gründe hierfür sind vielfältig. So bieten[11] Verzeichnisdienste eine sehr viel flexiblere Unterstützung zur Teilstring Suche als relationale Datenbanken, was sich gerade bei der Suche nach Stichworten, die in Themen oder Vorlesungen vorkommen gut einsetzen lässt. Darüber hinaus sind Verzeichnisdienste bei Such- und Leseanfragen sehr viel schneller als relationale Datenbanken, leiden dafür allerdings unter einer relativ schlechten Schreibperformance. Tatsächlich werden Verzeichnisdienste vor allem dort eingesetzt, wo der Schwerpunkt auf Retrieval- und nicht auf Updateaufgaben liegt. Um genau so eine Umgebung handelt es sich bei der Verwaltung der Lehrveranstaltungen und Nutzer. Hauptsächlich wird hier zur Information nach Themen, oder zur Authentisierung nach Nutzern gesucht. Schreibvorgänge werden nur bei der Nutzerregistrierung und dem Eintragen neuer Lehrveranstaltungen, also verhältnismäßig selten, vorkommen.

Da bei dem kommentierten Vorlesungsverzeichnis[12] an der TU Ilmenau ein LDAP Server zum Einsatz kommt – und die vorliegende Anwendung mit diesem zumindest integrierbar sein soll, wurde hier mit OpenLDAP ebenfalls ein LDAP Server gewählt.

Die endgültige Struktur des im Vorlesungsverzeichnis genutzten Baumes stand zu diesem Zeitpunkt allerdings noch nicht fest, weshalb eine beispielhafte und stark eingeschränkte eigene Struktur entwickelt wurde (siehe Abb. 6). Eine zentrale Überlegung dabei war, dass die Struktur des Baumes leicht zu verstehen, ausgewogen und realitätsnah sein sollte. Eine Anlehnung an den Aufbau der Universität war naheliegend – blieb die Entscheidung zu fällen, in welchem Detaillierungsgrad diese nachzubilden sei. Eine sehr ausführliche Nachbildung, mit unter Umständen vorzunehmender Untergliederung in einen Teilbaum für Lehrveranstaltungen und einen für Personen, hätte bei einer sehr großen Anzahl von Einträgen zu einem deutlich besseren Antwortverhalten geführt. Andererseits sprachen Überlegungen wie die Übersichtlichkeit (und damit Wartbarkeit) des Baumes, die Wiederverwendbarkeit der geschriebenen Programmteile, sowie die Verringerung redundanter Daten demgegenüber für einen möglichst kompakten und überschaubaren Entwurf.

Die letztendlich gewählte Struktur sieht vor, die Veranstaltungen und Personen in einem gemeinsamen Baum abzulegen (siehe Abb. 6). Sie entspricht zunächst der Unterteilung

[11] Vgl. Weltman & Dahbura /LDAP with Java/ 6
[12] Vgl. Steiert /Entwurf von Verzeichnisstrukturen/

der Uni in Fakultäten. Hierunter gibt es für jede Fakultät einen Zweig, in dem die dort eingeschriebenen Studenten registriert werden. Darunter wird zudem in Fachgebiete unterteilt, um das Ergebnis einer Suche nach Personen und Veranstaltungen besser einschränken zu können – und einzelne Zweige nicht mit Einträgen zu überfüllen.

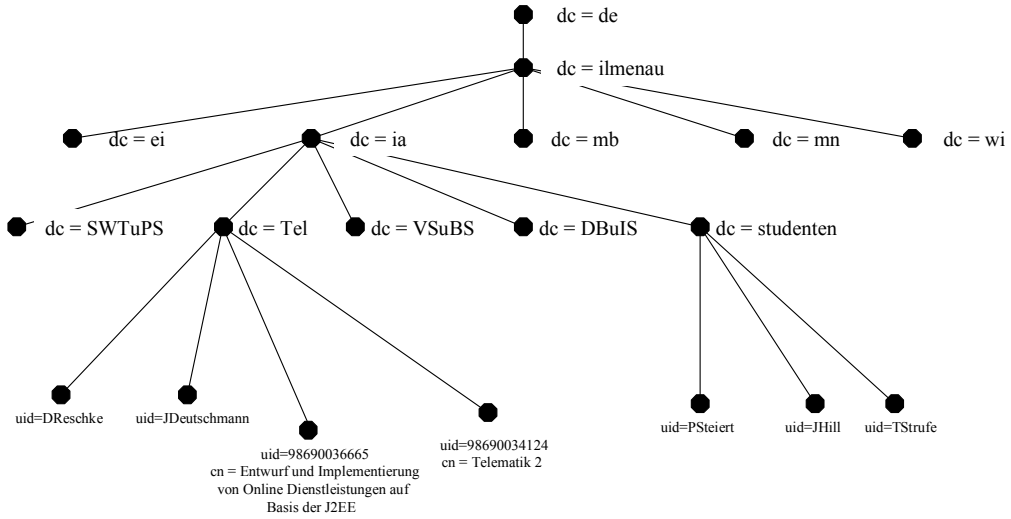


Abbildung 6: Struktur des LDAP Baums (Lehrveranstaltungen und Nutzer)

Als Schlüssel für Personen sollte eine selbstzuwählende Nutzerkennung dienen, wobei die Verantwortlichkeit für die Sicherstellung der Eineindeutigkeit dieser Kennung auf die Applikationsschicht verlagert wurde. Für den Schlüssel der Lehrveranstaltungen wurde ein Feld des Typs String vorgesehen, ohne nähere Vorgaben über dessen Inhalt zu machen. Der Aufbau der einzelnen Einträge ist beispielhaft im Anhang dargestellt.

Um Leistungen, unter denen hier belegte und abgeschlossene Lehrveranstaltungen verstanden werden zu speichern, wird eine relationale Datenbank genutzt. Zum Einsatz kommt das bei der J2EE mitgelieferte Cloudscape. Die Datenbank umfasst lediglich eine Tabelle (siehe Anhang). Belegt ein Nutzer eine Lehrveranstaltung, so wird in dieser Tabelle eine neue Zeile angelegt, in welcher die Einschreibungsinformationen gespeichert werden. Zusätzlich wird der Eintrag im Verzeichnis als belegt markiert. Nach Abschluss dieser Leistung wird die Information vom Betreuer respektive dem Lesenden um Note, evtl. Punkte, Art des Abschlusses und weitere benötigte Daten ergänzt. Handelt es sich um eine Projektveranstaltung wie ein Hauptseminar, eine Studien- oder eine Diplomarbeit wird gleichzeitig das zugehörige Blatt aus dem LDAP Baum entfernt.

Die Applikationsschicht

Die Applikationsschicht besteht aus drei EntityBeans, die eine Lehrveranstaltung, eine Person und einen (erweiterten) Schein darstellen, sowie einer Reihe SessionBeans, durch welche die Systemlogik implementiert wird. Einen Überblick über das Zusammenspiel dieser Komponenten gibt Abbildung 7.

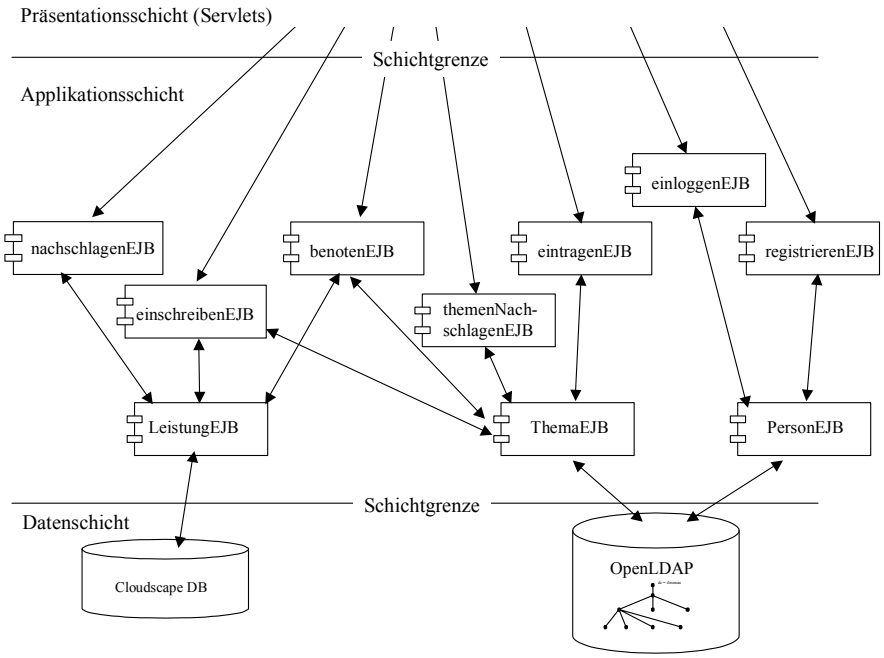


Abbildung 7: Komponentendiagramm der Applikationsschicht (vereinfacht)

Die für die Speicherung der Einschreibungen und erreichten Noten zuständige **LeistungEJB** wird zu dem Zeitpunkt, da sich ein Student für eine Veranstaltung einschreibt, durch die *einschreibenEJB* kreiert. Sie enthält zunächst nichts weiter als die Kennung des Studenten und Grundinformationen zu der belegten Veranstaltung. Hat der Student die belegte Veranstaltung abgeschlossen, werden die restlichen für die Erstellung eines Scheines relevanten Daten durch den betreuenden Mitarbeiter eingegeben, und von der *benotenEJB* in die EntityBean eingetragen. Die *nachschlagenEJB* greift auf ihre Methoden zur Auffindung zu, wenn ein Student sich seine abgeschlossenen oder offenen Leistungen anschauen will. Genauso ist es dem Mitarbeiter möglich, die von ihm betreuten Leistungen, seien sie schon benotet oder noch in der Bearbeitungsphase, mit Hilfe der *nachschlagenEJB* zur Anzeige zu bringen.

Die LeistungEJB enthält zunächst eine Reihe Methoden, mit denen Daten aus der unterliegenden Tabelle ausgelesen, oder in diese gespeichert werden. Darüber hinaus umfasst sie lediglich die im Home Interface vorkommenden Methoden, mittels derer es etwa möglich ist, ein Objekt zu finden, eine Bean zu kreieren, zu aktivieren, passivieren etc.
Die Implementierung der LeistungEJB erwies sich alles völlig unkritisch, genau wie die Integration mit der Cloudscape Datenbank.

Als Sicht auf die gespeicherten Veranstaltungen wurde die **ThemaEJB** entwickelt. Sie wird nach dem vollständigen Eintragen aller zu einem Thema gehörigen Daten vom betreuenden Mitarbeiter durch die *eintragenEJB* kreiert - und legt dem Entwurf nach einen neuen Eintrag im LDAP Verzeichnis an. Der eineindeutige Schlüssel für den Eintrag wird zum Zeitpunkt der Erstellung vom System generiert. Im Prototypen wurde sich hier schlicht der zum String konvertierten Systemzeit in Millisekunden bedient.

Auch die Nutzer des Systems sollten, durch die **PersonEJB** repräsentiert, im Verzeichnisdienst abgelegt werden. Des Entwurfs nach werden sie von der *registrierenEJB* kreiert, wobei sichergestellt wird, dass die gewählte Nutzerkennung noch nicht im System vorhanden ist, um spätere Uneindeutigkeiten auszuschließen.

Bei der Integration des Verzeichnisdienstes mit dem J2EE Server kam es zu unerwarteten Schwierigkeiten. So ist es laut Spezifikation einer EntityBean nicht möglich, einen I/O-Port zu öffnen und darauf auf eine Anfrage zu warten. Sehr wohl wäre es laut Spezifikation möglich, als Client einer Socket Verbindung aufzutreten, also einen Port zu öffnen und darauf eine Anfrage zu stellen. Die Idee hinter dieser Einschränkung liegt in der Philosophie, dass EntityBeans lediglich eine Sicht auf die unterliegende Datenbank darstellen, SessionBeans nur Workflow modellieren, keine der beiden aber als eigenständiger Server fungieren soll, begründet.
Mit der Referenz Implementation von Sun sind nun, überraschenderweise, beide Arten, einen Port zu öffnen unmöglich. Um diese Einschränkung zu umgehen, war geplant, einen selbst implementierten Port Manager zu nutzen.

Zur Implementierung eines Port Managers wurde nach dem Singleton-Pattern[13] eine nur einmalig zu instanzierende Klasse geschrieben. Die EntityBeans sollten diese Klasse über deren JNDI Namen referenzieren und daraufhin auf ihre Methoden zugreifen, welche als Resultat die Antwort Sets des LDAP Servers enthalten.
Leider erwies es sich sogar als schwierig, von den vorhandenen EntityBeans auf den implementierten Port Manager zuzugreifen. Nach eingehender Recherche der Online-Ressourcen und Foren stellte sich heraus, dass es bei dem Versuch von EntityBeans die in der Sun RI installiert sind, anders als über die JDBC Schnittstelle auf Daten oder andere Klassen zuzugreifen, des häufigeren zu Problemen kommt. Bei einem Umstieg auf einen beliebigen anderen Applikationsserver würden demnach diese Probleme nicht auftreten.
Eine andere Lösung, außer der Speicherung der gesammelten Daten in einer relationalen Datenbank, scheint es bei der Nutzung der Sun J2EE 1.2.1 Referenz Implementation allerdings auch nicht zu geben.

Die Präsentationsschicht

Aus Gründen der Akzeptanz sollen als Client Software der Veranstaltungsverwaltungs-Applikation lediglich einfache Webbrowser vorausgesetzt werden. Da im universitären Umfeld die Nutzer wissenschaftliche Mitarbeiter und Studenten sein werden, die aller Wahrscheinlichkeit nach überdurchschnittlich sicherheitsbewusst sind, muss davon ausgegangen werden, dass bei der Mehrheit der Browser die JavaScript Unterstützung nicht aktiviert, und das Hinterlegen von Cookies nicht möglich ist. In der Anwendung werden allerdings personenbezogene Daten gespeichert und Noten vergeben, wodurch der konsequenten Zugriffssicherung eine zentrale Rolle zukommt. Eine einfache Authentifikation, die nach einer Passwortabfrage den vollen Zugriff auf alle Daten gestattete, reicht schon aufgrund der unterschiedlichen Nutzergruppen nicht aus. Das System muss also nach einer personenbezogenen Authentifizierung beim Login eine ständige Verbindung zu den Nutzern halten. In der Applikationsschicht erscheint das relativ einfach, beim Login wird die PersonEJB des Nutzers instanziiert, zu ihm assoziiert und während der gesamten Verbindung im Speicher gehalten. Auf der Ebene der Präsentationsschicht gestaltet sich das etwas schwieriger. Das genutzte http-Protokoll ist verbindungslos orientiert, was eine eigene Nutzerverfolgung nötig macht.

[13] Vgl. Grand /Patterns in Java/ 127

Diese Nutzerverfolgung wird im allgemeinen Session Management genannt, die Aktionen des Nutzers werden für die Dauer seiner Sitzung verfolgt, wodurch eine ständige erneute Authentisierung vermieden werden kann.

Die Java Servlet API bietet von sich aus zwei Möglichkeiten, ein Session Management zu realisieren. Die erste und gebräuchliche Variante speichert einen Nutzerschlüssel auf dem Rechner des Nutzers in einer eigenen Datei (als Cookie) ab, um diesen bei allen http-Anfragen auslesen und damit den Nutzer identifizieren zu können. Diese Möglichkeit setzt voraus, dass die Browser der Nutzer offen konfiguriert sind – und das Speichern von Cookies erlauben, eine Annahme die in der vorliegenden Applikation aus oben genannten Gründen nicht gemacht werden soll.

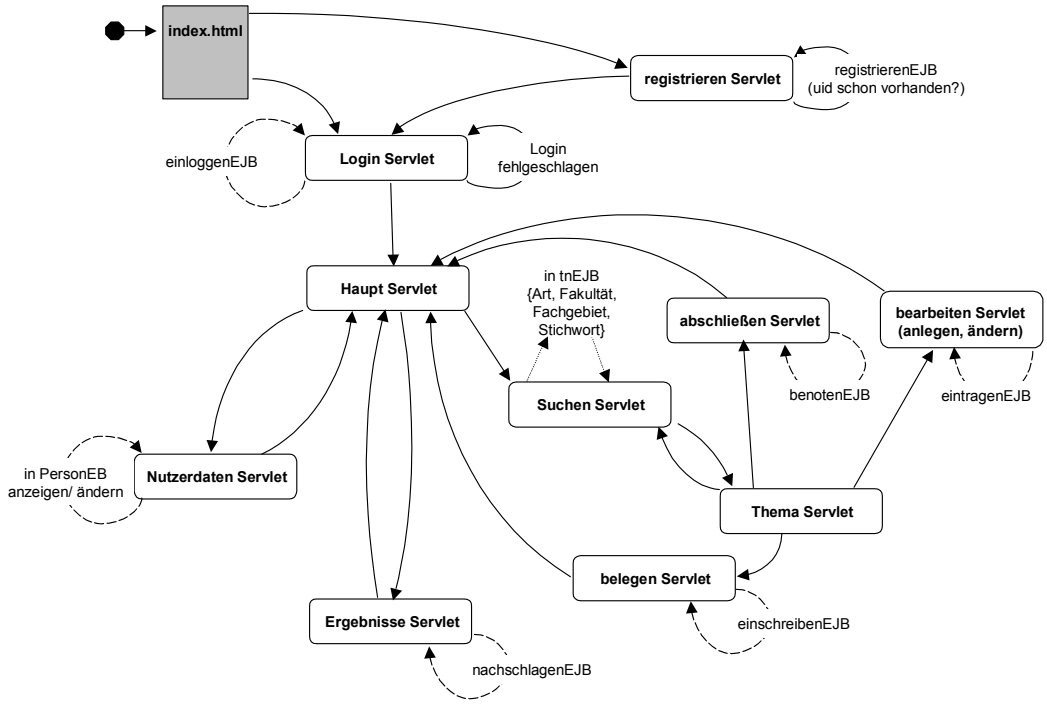


Abbildung 8: Komponentendiagramm der Präsentationsschicht (vereinfacht)

Die zweite Variante, welche in der Applikation zur Anwendung kommen soll, bedient sich eines Tricks, der sich seit längerem bei kleineren, auf Scripten beruhenden Online-Applikationen eingebürgert hat, des „URL- Rewritings“. Am Anfang einer Sitzung wird dem Client als Schlüssel etwa eine große zufällige Zahl oder vergleichbares zugewiesen, der beim Aufbau der Web Seiten in jeden Link eingebunden wird. So ist es dem Webserver möglich, einen Client über die Dauer seiner Sitzung anhand seines Schlüssels zu identifizieren. Voraussetzung dafür ist natürlich, dass dieser Schlüssel in

alle Links jeder neu generierten Seite geparst wird, da bei der Nutzung eines nicht angepassten Links der Schlüssel – und damit die Sitzung – verloren wäre.
Die Realisierung dieser beiden Varianten kann bei der Programmierung von Java-Servlets im Großen und Ganzen dem Webserver überlassen werden. Dieser kümmert sich um die serverseitige Sicherung aller zur Sitzung abgelegten Daten und die Generierung sowie das Ablegen des Schlüssels auf dem Client Rechner. Als Standard der Webserver kommen Cookies zum Einsatz. Will man sich des URL-Rewritings bedienen, so muss das bei der Eröffnung der Sitzung dem Server mitgeteilt werden. Die Integration des auch hier vom Server generierten Schlüssels wird über das enkodieren aller auf den Seiten vorkommenden Links mittels einer Java Methode erreicht.
Um nicht nur einer möglichen lokalen Verteilung des Servlet Containers und des EJB Containers, sondern auch einer sauberen Trennung zwischen Präsentations- und Applikationsschicht Rechnung zu tragen, ist die Architektur der Servlets nur auf Nutzerführung und Präsentation ausgerichtet. Jedes Servlet soll auf nur eine Enterprise Bean zugreifen, die dann die Anwendungslogik übernimmt und unter Umständen wiederum auf mehrere unterliegende Persistierungsmechanismen zugreift.
Allerdings sollte auch die Belastung des Netzes zwischen den beiden Schichten nicht unberücksichtigt bleiben, weshalb die Servlets zunächst so viele Nutzereingaben wie möglich sammeln, bevor sie mit den Enterprise Beans in Kontakt treten – und das gesammelte Datenpaket austauschen.
Diese Überlegung kommt insbesondere bei den „registrieren-“, „nutzerdaten-“ und „bearbeiten- Servlets“ zum tragen, die zunächst alle Eingaben speichern, daraufhin zum EJB Container und der betreffenden SessionBean Kontakt aufnehmen und die gesammelten Daten als Paket übergeben.
Die geschriebenen Servlets sowie ihr Zusammenspiel sind in Abbildung 8 als Überblick dargestellt.

Zusammenfassung und Ausblick

Die in der umgesetzten Online Dienstleistungen der Lehrveranstaltungsverwaltung gehören zu den Kernbestandteilen eines jeden Informationssystems. In jedem, wie auch immer gearteten Projekt, welches eine „virtuelle Universität“ umsetzen will, wird es eine Nutzerverwaltung und die Möglichkeit geben, das Lehrangebot zu erfassen, zu administrieren und zu belegen. Im Netz angeboten und einzelnen Fachgebieten, den Fakultäten, oder sogar der ganzen Universität zur Verfügung gestellt, würde es helfen, die existierenden Prozesse und Abläufe zu automatisieren – und dadurch immens zu vereinfachen.

In einen anderen frei verfügbaren J2EE Server (wie etwa JBoss oder Enhydra) installiert, sollte es kein Problem sein, die Applikation mit einem Verzeichnisdienst zu integrieren. Allerdings sollte bei dieser Nutzung, ein vollständiges LDAP Schema für die genutzten Einträge und ihre Attribute entwickelt werden, da die Daten bis jetzt nur mit abgeschalteter Schema Überprüfung abgelegt werden können. Nur bei einem voll entwickelten Schema kämen aber weitere LDAP-spezifische Vorteile, wie automatische Integritätsüberprüfungen oder Ähnliches zum Tragen. In der momentanen Konfiguration wird der Server nur zu einer einfachen hierarchischen Datenbank mit guten Lese- und Sucheigenschaften degradiert.

Auch eine Integration mit dem kommentierten Vorlesungsverzeichnis würde in diesem Fall kein Problem darstellen. Es bedürfte lediglich einer weiteren zu erstellenden Komponente, welche, gesetzt den Fall, eine Veranstaltung würde im lokalen Verzeichnisbaum nicht gefunden, eine Verbindung zum Vorlesungsverzeichnis aufnähme und die gesuchten Daten von dort holte. Es sollte dabei natürlich berücksichtigt werden, dass keine Lehrveranstaltung mit versehentlich leicht unterschiedlichen Namen doppelt eingetragen, oder tatsächlich ähnliche Veranstaltungen als gleich interpretiert und ignoriert würden.

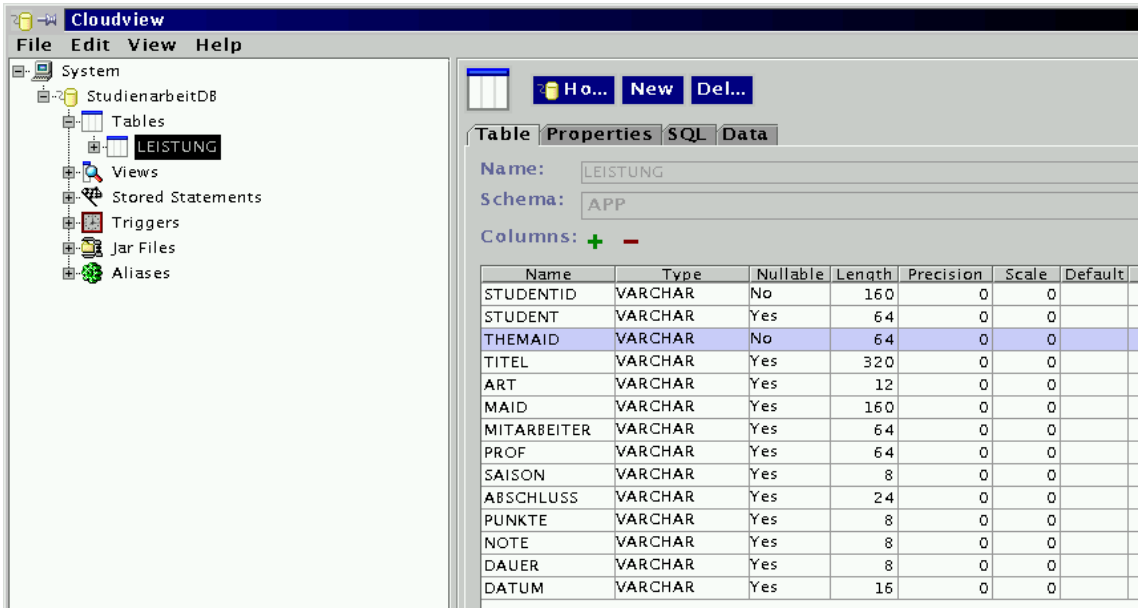
Eine sinnvolle Erweiterung wäre außerdem eine Modellierung von teilnehmer-beschränkten Lehrveranstaltungen wie Praktika. Diese zeichnen sich dadurch aus, dass sie zwar regelmäßig als identische Veranstaltungen stattfinden, allerdings von jedem Studierenden nur einmal – und das zu einem fest vergebenen Termin – besucht werden. Zur Zeit ist die Anwendung noch nicht in der Lage, solche Veranstaltungen nutzerfreundlich anzubieten.

Bei professionellem Einsatz wäre die Anwendung außerdem zur Nutzung von installationsabhängigen „Property-“ oder Konfigurationsfiles zu ändern. In der vorliegenden Version sind alle Pfade und Portnummern noch hart im Code vergeben, was eigentlich nicht der J2EE Philosophie entspricht.
Außerdem wäre zu überlegen, ob nicht aufgrund der dann besseren Performance unter Umständen auf die Nutzung der LeistungEJB zugunsten eines direkten Datenbankzugriffes der SessionBeans verzichtet werden sollte. Des Weiteren wäre es eine Überlegung wert, ob ThemaEJB und PersonEJB zu einer einzelnen EntityBean verallgemeinert werden könnten. Beide Maßnahmen unterlaufen direkt die Grundidee der Spezifikation, würden allerdings deutliche Geschwindigkeitsvorteile erbringen. Die Unterstützung von EntityBeans ist in den vorhandenen Servern zwar relativ gut, die der Architektur (welche ein ständiges updaten der Tabellen und der Beans nötig macht) inhärenten Nachteile sind allerdings bei der Realisierung komplexer Anwendungen noch immer spürbar.
Als Stand-Alone-System wäre die vorliegende Anwendung tatsächlich nur bedingt sinnvoll. Viel effizienter ist die Integration mit dem an der TU Ilmenau in Entwicklung befindlichen Framework[14], welches eine Applikation zur asynchronen Kommunikation und weitere Komponenten für ein sinnvolles Universitäts Informationssystem enthält und hierdurch positiv ergänzt würde.

[14] Vgl. Dittmar /Framework für Online-Dienstleistungen/

Anhang

Aufbau der relationalen Datenbank



Name	Type	Nullable	Length	Precision	Scale	Default
STUDENTID	VARCHAR	No	160	0	0	
STUDENT	VARCHAR	Yes	64	0	0	
THEMAID	VARCHAR	No	64	0	0	
TITEL	VARCHAR	Yes	320	0	0	
ART	VARCHAR	Yes	12	0	0	
MAID	VARCHAR	Yes	160	0	0	
MITARBEITER	VARCHAR	Yes	64	0	0	
PROF	VARCHAR	Yes	64	0	0	
SAISON	VARCHAR	Yes	8	0	0	
ABSCHLUSS	VARCHAR	Yes	24	0	0	
PUNKTE	VARCHAR	Yes	8	0	0	
NOTE	VARCHAR	Yes	8	0	0	
DAUER	VARCHAR	Yes	8	0	0	
DATUM	VARCHAR	Yes	16	0	0	

Aufbau der LDAP-Einträge für Themen

- dn: uid=981462885471,dc=tel,dc=ia,dc=ilmenau,dc=de
  - objectclass: top
  - objectclass: Lehrveranstaltung
  - uid: 981462885471
  - cn: Informatik (ET)
  - art: VL
  - besitzerID: DReschke
  - prof: DReschke
  - termin: Mit
  - Time: 9:00
  - Raum: HS2
  - FinTermin: 17.02.2001
  - FinTime: 19:00
  - FinRoom: GrHs

Aufbau der LDAP-Einträge für Personen

Mitarbeiter:

- dn: uid=MusterProf,dc=tel,dc=ia,dc=ilmenau,dc=de
  - objectclass: top
  - objectclass: person
  - objectclass: organizationalPerson
  - uid: MusterProf
  - cn: Muster Professor
  - sn: Professor
  - vn: Muster
  - title: Prof. Dr. Ing. habil.
  - userPassword: Mpgeheim
  - Raum: CC445
  - Phone: 03677-690
  - eMail: Muster@prakInf.tu-ilmenau.de

Studenten:

- dn: uid=Mmuster,dc=Studenten,dc=ia,dc=ilmenau,dc=de
  - objectclass: top
  - objectclass: person
  - objectclass: student
  - uid: Mmuster
  - cn: Maximilian Mustermann
  - sn: Mustermann
  - vn: Maximilian
  - userpassword: MMgeheim
  - matrikel: 90
  - matrikelNr: 22222
  - email: Mustermann@rz.tu-ilmenau.de
  - studiengang: Informatik
  - nebenfach: Wirtschaftswissenschaften
  - seminargruppe: I1

Danksagung

Wenn es auch bei Schriften des Umfanges der vorliegenden Studienarbeit nicht üblich ist möchte ich dennoch einigen Personen, ohne die diese Studienarbeit nicht das geworden wäre, was sie jetzt ist, gerne meinen Dank aussprechen.

Zunächst gilt mein besonderer Dank meinem Betreuer Jörg Deutschmann für die ausgesprochen produktive und menschliche Betreuung. Darüber hinaus möchte ich mich bei den folgend aufgeführten Personen für technische Tipps, das Korrekturlesen der Arbeit und wertvolle Denkanstöße von unschätzbarem Wert bedanken: Tony Cook, Andrea Heinze, Jan Hill, Cornelia Krannich, Christoph Lühr, Steffen Michael, Daryl Monge, Maryam Moradi, Professor Dietrich Reschke, Peter Steiert, Professor Dirk Stelzer, Sonja van Thiel, Nils Thode und Anke Wiertelorz.

Alle in der Version 1.0 der vorliegenden Arbeit immer noch befindlichen Fehler, inhaltlicher und förmlicher Art, sind voll und ganz von mir zu verantworten.

Quellenverzeichnis

Div /Distance Learning/

http://www.lifelonglearning.com/

Abruf am: 2001-02-19

Div /Virtus/

http://www.virtus.uni-koeln.de/virtus/

Abruf am: 2001-02-19

Div /UnivIS/

http://univis.uni-bamberg.de/

Abruf am: 2001-02-19

Div /Unituell/

http://www.unituell.de/

Abruf am: 2001-02-19

Matena & Hapner /EJBSpec/

Matena, Vlada & Hapner, Mark „Enterprise JavaBeans™ Specification, v1.1, Public Release“, Sun Microsystems, Inc. Oktober 1999.

Haefel /EJB 2.0/

http://www.javaworld.com/javaworld/jw-06-2000/jw-0609-ejb.html

Abruf am: 2001-01-31

Haefel /EJB/

Richard Monson-Haefel: Enterprise JavaBeans

Sebastopol u. a., 1999

Dittmar /Asynchrone Kommunikation/

Jörg Dittmar: „Integration und prototypische Entwicklung von Funktionen zur synchronen Kommunikation zwischen den Benutzern eines plattformunabhängigen Informationssystems“

Ilmenau, 2000

Pawlan: /Writing Enterprise Applications/

Monica Pawlan: “Writing Enterprise Applications with Java™ Platform, Enterprise Edition”, Sun Microsystems, Inc. 1999

Roman /Mastering/

Ed Roman: Mastering Enterprise JavaBeans

Wiley, New York u. a., 1999

Weltman und Dahbura /LDAP with Java/

Rob Weltman und Tony Dahbura: LDAP Programming with Java

Reading u. a., 2000

Perrone und Chaganti /Enterprise Systems/

Paul J. Perrone und Venkata S.R.R. Chaganti: Building Java Enterprise Systems with J2EE

SAMS, Indianapolis, 2000

Steiert /Entwurf von Verzeichnisstrukturen/

Peter Steiert: Entwurf von Verzeichnisstrukturen für das modularisierte Studium an der Technischen Universität Ilmenau.
Diplomarbeit, Ilmenau, 2001

Grand /Patterns in Java/

Mark Grand: Patterns in Java Volume 1
Wiley, New York u. a. 98

Dittmar /Framework für Online Dienstleistungen/

Jörg Dittmar: Ein Framework für Online Dienstleistungen auf der Basis von Enterprise JavaBeans.
Diplomarbeit, Ilmenau, 2001

Abbildungsverzeichnis

Index