1. Begriffe

1.1. Begriffe, die mit der Erstellung des Softwareproduktes in Verbindung stehen

Diese Begriffe wurden ausgewählt, da sie aus der Aufgabenstellung hervor gehen und unmittelbar mit dem Projekt Übungsmanager in Verbindung stehen.

- **Web-Applikation:** Internetprogramme, auch Webanwendungen genannt, sind Anwendungen, die auf einem Webserver ausgeführt werden und dann in einem Browser dargestellt werden. Dabei kann der Browser irgendwo aus dem Netzwerk heraus die Webanwendung aufrufen. Die Sprache für die Oberfläche, die in dem Browser angezeigt wird, ist hauptsächlich HTML. Die Webanwendungen werden im Browser über eine URL aufgerufen.
- **Instanzen (UebManager-Instanzen):** Ist eine laufende Kopie von einer Anwendung, welche aber nicht in unmittelbaren Zusammenhang mit anderen laufenden Instanzen steht.
- **Kooperationsplattform (Plattform):** Ist eine Anwendung, welche auf einem Server läuft. Sie ermöglicht die Verwaltung von Daten, Terminen und Benutzern.
- **Webservice:** Ein Webservice ist ein Dienst, der mit Hilfe von XML auf der Basis von Internet-Netzwerkprotokollen erbracht wird. Web Services sind nicht für menschliche Benutzer gedacht, sondern für Softwaresysteme, die automatisiert Daten austauschen und/oder Funktionen auf entfernten Rechnern aufrufen.
- **XML:** XML (Extensible Markup Language) ist ein Standard zur Erstellung strukturierter, maschinen- und menschenlesbarer Dateien. XML definiert dabei den grundsätzlichen Aufbau solcher Dateien. Für die konkreten Anwendungsfälle müssen die Details des Dateiaufbaus jedoch weiter spezifiziert werden. XML ist damit ein Standard zur Definition von beliebigen, in ihrer Grundstruktur jedoch stark verwandter Auszeichnungssprachen. XML ist eine vereinfachte Teilmenge von SGML.
- **Schnittstelle:** Eine Schnittstelle (englisch interface) ist ein Teil eines Systems, das dem Austausch von Informationen, Energie oder Materie mit anderen Systemen dient. Eine Schnittstelle wird durch eine Menge von Regeln beschrieben, der Schnittstellenbeschreibung. Neben der Beschreibung, welche Funktionen vorhanden sind und wie sie benutzt werden gehört zu der Schnittstellenbeschreibung auch ein so genannter Kontrakt, der die Semantik der einzelnen Funktionen beschreibt.
- **Standard-Webbrowser:** Benutzerschnittstelle für den Zugriff auf das Internet. Die wichtigsten Webbrowser sind derzeit der Internet Explorer, Netscape Navigator, Opera und Mozilla.
- **Servlet:** Servlets sind Java-Programme, die in einem Web- oder Applikationsserver ablaufen und dynamische Webseiten generieren.
- **MVC:** Beim MVC-Modell handelt es sich um ein Entwurfsmuster zur Trennung bestimmter Programmeigenschaften. Die Idee des Musters ist die Trennung des Programms in die drei Einheiten Datenmodell (Model), Präsentation (View) und Programmsteuerung (Controller). Ziel des Modells ist ein flexibles Programmdesign, um u. a. eine spätere Änderung oder Erweiterung einfach zu halten und die Widerverwendbarkeit der einzelnen Komponenten zu ermöglichen.
- **Struts:** ein Open Source – Framework für die Präsentationsschicht von Java Web-Anwendungen. Es ist eines der bekanntesten Jakarta-Projekte und beschleunigt die Erstellung der Web-Applikationen wesentlich, indem es einen standardisierten Prozess zu verarbeiten von HTTP-Anfragen vornimmt. Dabei bedient es sich standardisierter Technologien wie JavaServlets, Java Beans, Ressource Boundles und XML sowie verschiedene Jakarta Commons Pakete.
- **Tomcat:** ein auf Java basierender frei verfügbarer Webserver, der u.a. speziell programmierte JSP und Servlet ausführen kann.
- **Java Beans:** wieder verwendbare Softwarekomponenten, die in Java realisiert werden. Der Einbau dieser Komponenten in eine Applikation oder in ein Applet geschieht mit Hilfe einer grafische JDE.

Java Beans sind darüber hinaus kompatibel zu ActiveX-Controls, sie können in ActiveX-fähigen Umgebung ausgeführt werden.

- **Web-Server:** Server, der Web-Seiten im Internet oder im Intranet über das HTTP-Protokoll versendet. Auch HTTP-Server genannt.

1.2. Nutzungsrechte der verschiedenen Personen

Hier wird beschrieben, welche verschiedenen Rechte die unterschiedlichen Personen haben.

- **Nutzer:** Ist eine Person, die bestehende Accounts von Uebmanager Instanzen übernehmen kann. Der Nutzer kann in einer Rolle als Student, Dozent oder Administrator agieren.
- **Student:** Ist eine bestimmte Rolle, er kann sich für Lehrveranstaltungen, Übungen und Prüfungen einschreiben.
- **Dozent:** Ist ein Mitarbeiter einer Hochschuleinrichtung und kann Einschreibungen für Veranstaltungen löschen, hinzufügen und konfigurieren.

1.3. Begriffe für technische Hilfsmittel

Hier sind Geräte beschrieben, die man benötigt um die Software zu benutzen.

- **Mobiles Endgerät:** Ist ein technisches Gerät, welches die mobile Nutzung von unserem Softwareprodukt ermöglicht. (z.B.: PDA, Laptop, Handy, ...)

1.4. Begriffe für den Umgang mit der Software

Allgemeine Begriffe, die auftauchen können, wenn man mit dieser Software arbeitet

- **Benutzer-Account:** Ein Benutzerkonto (englisch user account) bezeichnet im IT-Bereich einen Zugang zu einem IT-System. Üblicherweise muss man sich vor dem Zugriff mit einem Benutzernamen und Passwort authentifizieren.
- **Listen:** Sind Ergebnisse der Einschreibung, sie werden durch die Plattform über eine Schnittstelle zur Verfügung gestellt.
- **Login:** Als Login bezeichnet man den Vorgang sich in einem Computersystem bei einem speziellen Dienst anzumelden (neudeutsch einzuloggen). Gewöhnlich dient der Vorgang dazu dem System mitzuteilen, dass man nun als Benutzer anwesend ist und stellt den Beginn einer Sitzung dar.
- **Logout:** ist das verlassen oder abmelden von Übmanager.
- **Forum:** dient dem asynchronen Austausch von Information zwischen den Benutzern der Kooperationsplattform. Hier können Meinungen ausgetauscht werden, Fragen gestellt und beantwortet werden.

2. Konzepte

2.1. Struts Framework

Struts ist ein Open Source – Framework für die Präsentationsschicht von Java-Web-Anwendungen, es ist eines der bekanntesten Jakarta-Projekte und beschleunigt die Entwicklung von Web- Applikationen wesentlich, indem es einen standardisierten Prozess zur Verarbeitung von HTTP- Anfragen vornimmt. Dabei bedient es sich standardisierter Technologien wie JavaServlets, Java Beans, Resource Bundles und XML sowie verschiedener Jakarta Commons Pakete.

Für den Entwickler bedeutet das, dass viele applikationsrelevante Funktionen bereits implementiert und einsatzbereit sind. Struts wurde von versierten Entwicklern geschaffen und wird bereits in hunderten Applikation eingesetzt. Das garantiert ein solides Framework mit den besten existierenden

Softwareentwicklungsstrategien. Struts wurde von Craig McClanahan (http://blogs.sun.com/roller/page/craigmcc) im Jahr 2000 entwickelt. Seitdem arbeitet eine ständig wachsende Entwicklergemeinde an der Verbesserung des Frameworks.

Funktionsweise: Struts basiert auf bereits vorhandenen und in der Praxis bewährten Javatechnologien wie Java Server Pages, Java Servets, XML und Resource Bundles. Besonderns die letztere Technologie erleichtert die Implementierung von Mehrsprachigkeit in Webanwendungen. Man spricht in diesem Zusammenhang auch von Internationalisierung. Das Framework wurde im Jahre 2000 der Apache Software Foundation hinzugefügt und untersteht dem Jakarta Projekt.

Über das weit verbreitete Dispatcher-Konzept wird das MVC Pattern implementiert. Es trennt die Präsentationsschicht (View) von den Daten (Model) und verbindet es über die Logik (Controller). Struts ist auch für größere Projekte sinnvoll, solange man das gesamte Framework als reines Frontend betrachtet und die eigentliche Arbeit in tiefere Schichten verschiebt.

Mittlerweile gibt es das Framework in der Version 1.1 und bietet ein paar Vorteile gegenüber seiner Vorgängerversion. So besteht in der aktuellen Version 1.1 die Möglichkeit Tiles zu definieren. Diese bieten die Möglichkeit erweiterter Templates mit denen eine weitere Modularisierung von Webanwendungen und deren komfortable Konfiguration über XML möglich ist.

2.2. Servlets

Java Servlets stellen den Ansatz Suns dar, Java als Ersatz für CGI-Skripte zu etablieren. Sun selbst definiert das so: "A servlet can almost be thought of as an applet that runs on the server side - without a face."

Es handelt sich also um eine Technologie, mit der in Java geschriebene Programme Anfragen von Clients über HTTP empfangen, verarbeiten und beantworten können. Die Ausführung der Java-Programme und die Bereitstellung der Verbindung zum Client etc. erfolgt dabei durch einen so genannten Servlet-Container, und weitgehend transparent für den Programmierer. Vorraussetzung ist dabei, dass selbst programmierte Servlets immer Unterklassen einer der von Sun bereitgestellten Servletklassen sein müssen, hier HttpServlet.

Die Vorteile gegenüber anderen Ansätzen sind dabei klar: Zugriff auf die komplette Java API und viele Java-Bibliotheken von Drittherstellern, die weitgehende Plattformunabhängigkeit und die Existenz von ausgereiften Entwicklungsumgebungen.

2.3. Model-View-Control

Model-View-Control (MVC) Architektur beschreibt die Modularisierung einer Software, die mit einem Bediener im Dialog steht. Ein Dialog-Programm besteht demnach aus einem Modell (model), in dem die aufgabenspezifischen Objekte mit ihrer Funktionalität und ihren Daten definiert sind, dem Bildschirm (view) und einem Kontroll-Modul (control). Diese steuert die Kommunikation zwischen dem Bildschirm (view) und diesen Objekten (model). Diese Trennung von Programm und Benutzer-Oberfläche erlaubt es, die Mensch-Maschine Schnittstelle völlig individuell und damit unabhängig von der Funktionalität eines Programms zu gestalten.

MVC trennt die Ebenen in drei selbständige Einheiten. Der Begriff Model wird für die Applikation ohne Benutzungsoberflächen (Fachkonzept), also für die interne Datenverarbeitung verwendet. Die einzelnen Views sind für die aktuelle Darstellung der Eingangs- und Ausgangsdaten in den jeweiligen Anzeigeobjekten (z.B.: Weboberfläche) verantwortlich. Sie werden dabei von der Client unterstützt.

Der Controller überwacht alle Eingabedaten. Diese werden an die zuständigen Klassen weitergeleitet. Änderungen der Modelldaten werden also von Controllerklassen (z.B.: mit ActionServlet) eingeleitet. View und Controller bilden zusammen die Benutzungsoberfläche.

Diese Beziehungen bestimmen den möglichen Fluss der Botschaften und der Daten. Das besondere Kennzeichen der Architektur ist die Tatsache, dass das Model weder die Views noch den Controller kennt. Praktisch heißt das, dass in den Datenklassen eine View– oder Controllerelemente aufgerufen werden dürfen! Dies bedeutet, dass die interne Datenverarbeitung von der Benutzungsoberfläche

gänzlich abgekoppelt wird. Änderungen in der Benutzungsoberfläche haben also keine Auswirkung auf die interne Verarbeitung der Daten und der Datenstruktur. Jedoch wird auch damit die Möglichkeit eröffnet, gleichzeitig die Daten mehrfach unterschiedlich darstellen zu können.

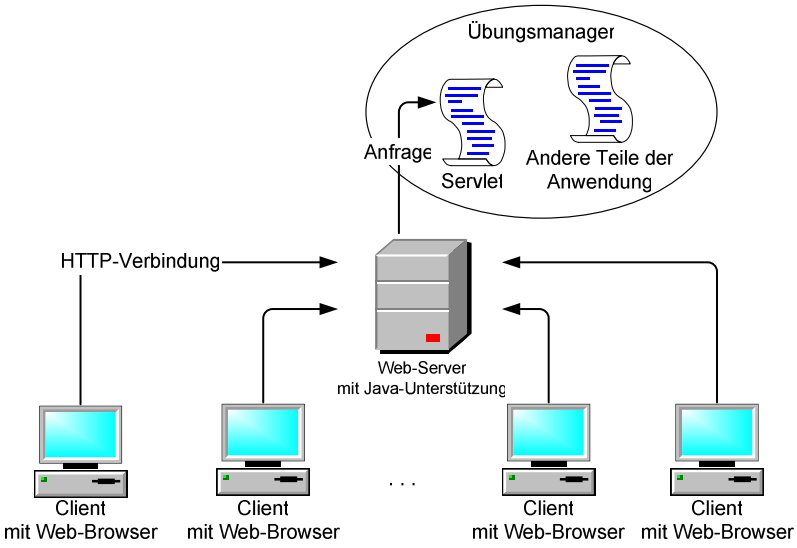
3. Beschreibung der Rahmenapplikation

3.1. Funktionsweise

Der **UebManager** ist ein web gestütztes Informationssystem zur Begleitung des Übungsbetriebes an einem Lehrstuhl einer Universität. Die Implementierung basiert auf der Servlet-Spezifikation der J2EE.

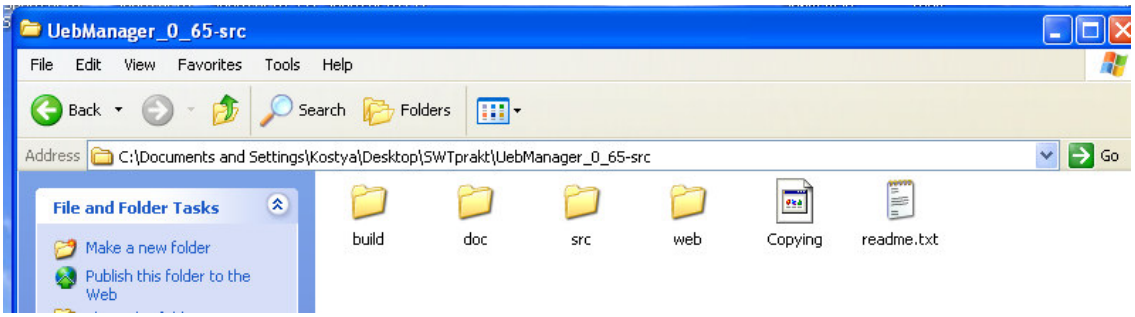
UebManager ist eine **Web-Anwendung**, d.h.:

- Sie funktioniert nach dem „Client-Server“ Prinzip
- Clients sind **Browser** der **Endbenutzer**, d.h. es gibt kein echtes **Client-Teil** der Anwendung.
- Das Server-Teil läuft nicht selbstständig, sondern benötigt einen **Web-Server** mit einem Servlet-Container, der die Servlet-API 2.3 unterstützt, wie z.B. Tomcat (http://jakarta.apache.org/tomcat/index.html) ab Version 4.
- Alle Daten werden auf dem Server gespeichert. Alle Applikationen laufen auf dem Server:

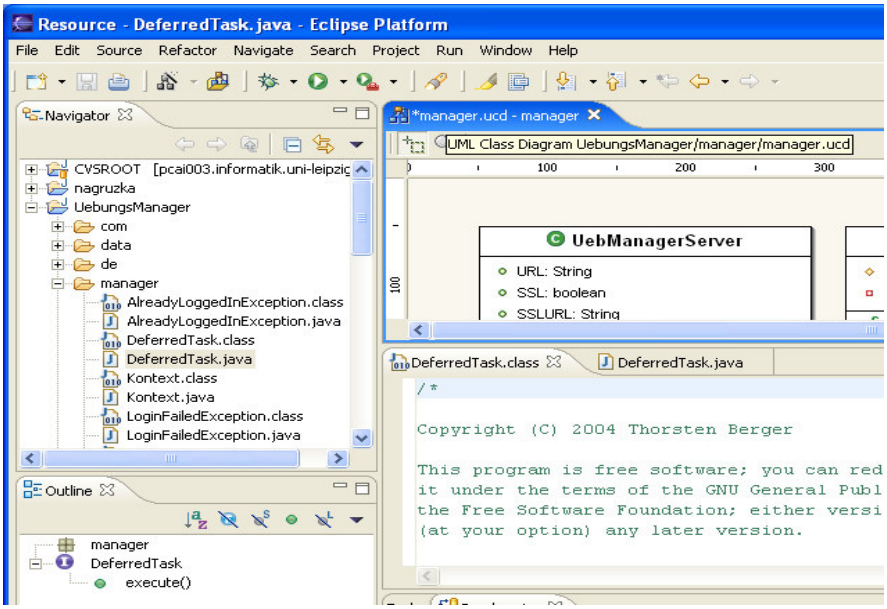


3.2. Quelle

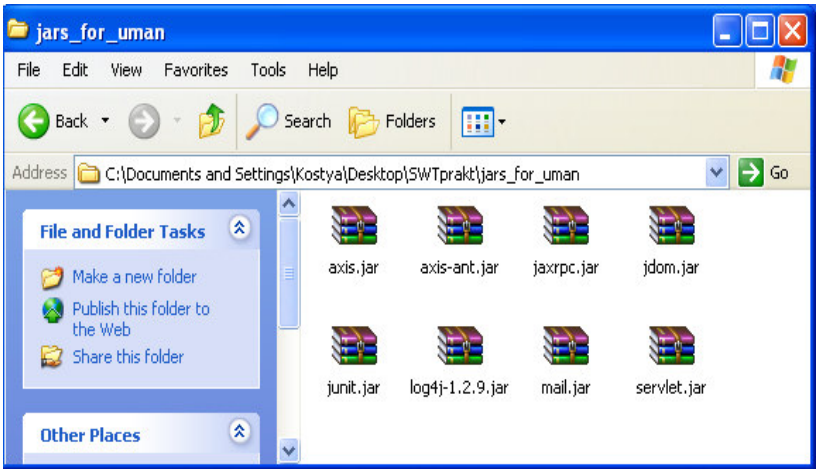
Der **UebManager** ist als OpenSource-Projekt unter folgender URL verfügbar: http://sourceforge.net/projects/uebman. Auf dieser Seite ist eine Package mit allen Java-Quelltexten frei zum Download verfügbar. Inhalt des Packages:



Im Verzeichnis **“src”** liegen alle Quelltexte in Java. Dieses Verzeichnis lässt sich in IDE Eclipse einfach importieren:



Danach ist automatische Kompilation bzw. Build möglich. Dabei benötigt Eclipse manche zusätzliche **.jar** – Bibliotheken, die man auch problemlos im Internet finden kann:



Nach dem Build ist die Anwendung bereit, in Web-Server exportiert zu werden. Die Autoren von UebManager empfehlen, Apache Tomcat als Web-Server zu verwenden. Nach der Installation von Tomcat hat man folgendes zu erledigen:

- Im Verzeichnis ***<Tomcat installation dir>*\webapps** ein neues Verzeichnis **„UebManager“** erstellen
- Den Inhalt des Verzeichnisses **„web“** aus dem UebManager-Package nach ***<Tomcat installation dir>*\webapps\UebManager** kopieren
- Im Verzeichnis ***<Tomcat installation dir>*\webapps\UebManager\WEB-INF** ein neues Verzeichnis **„classes“** erstellen
- Den Inhalt des Projekts aus dem Eclipse-Workspace Verzeichnis nach **„classes“** kopieren.

Danach muss Tomcat neu gestartet werden und nach dem Neustart ist das Ergebnis im Browser unter http://localhost:8080/UebManager/manager.UebManager sichtbar.

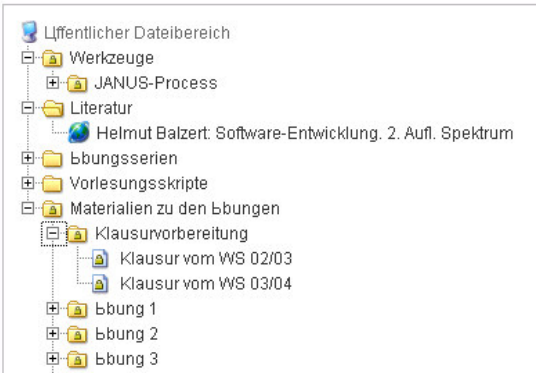
- Weitere Verwaltungen sind mit Übungsmanager selbst möglich. Die Logindaten für Administrierung sind in der Datei readme.txt aus dem Download-Paket enthalten.

3.3. Beschreibung der Leistungsmerkmale

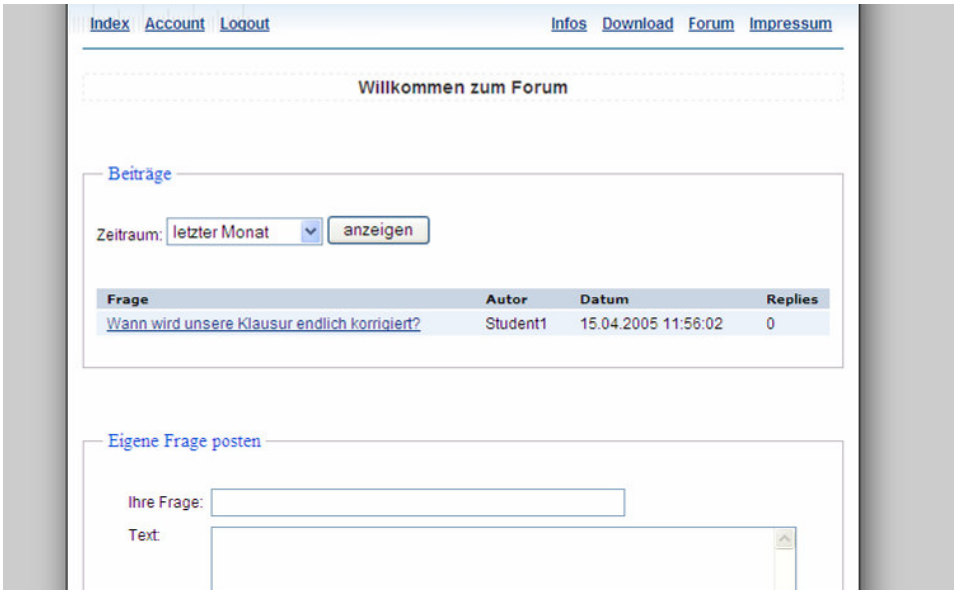
Übungsmanager verwaltet alle relevanten Daten für den Übungsbetrieb, z.B. an einer Universität. Die im Moment bestehende Version ist auf genau ein Fach eingeschränkt. Benutzer des Systems sind Studenten, Tutoren (Hilfsassistenten) und Dozenten.

Grundlegende Funktionen des Programms sind:

- **Benutzerverwaltung (Accounts).** Benutzer mit Rollen „Administrator“ und „Dozent“ können Benutzer-Accounts erstellen, ändern und löschen. Außerdem ist es möglich, dass sich die Studenten zur Veranstaltung selbst einschreiben. Insgesamt gibt es vier Benutzerrollen: „Student“, „Tutor“, „Dozent“ und „Admin“. Selbstverständlich haben Benutzer mit unterschiedlichen Rollen auch unterschiedliche Rechte im System. Das Passwort kann von jedem User beliebig geändert werden, jedoch nur, wenn man Kenntnis über sein altes Passwort hat.
- **Gruppenverwaltung.** Dozenten (evtl. auch Tutoren) können Gruppen erstellen. Der Begriff der Gruppe entspricht dem üblichen Begriff der Übungsgruppe an der Universität: jeder Student ist genau einer Gruppe zugeordnet, die Übung in der Gruppe findet in einem bestimmten Raum, mit einer Platzbegrenzung für die Teilnehmer, zu einer bestimmten Zeit statt. Jede Gruppe hat einen bestimmten Leiter. Alle diese Daten (Zeit, Ort, Leiter, max. Anzahl von Teilnehmern, usw.) werden im Übungsmanager gespeichert. Die Studenten können sich selbst in die Gruppe einschreiben oder die Gruppe wechseln, solange es in der gewünschten Gruppe noch freie Plätze vorhanden sind.
- **File-Manager.** Die Benutzer können aus dem Übungsmanager, genauso wie aus einer normalen Web-Seite, Dateien herunterladen. Es gibt sowohl die Dateien, die nur für Benutzer des Übungsmanagers zugänglich sind, als auch die Dateien, die keine Autorisierung benötigen, d.h. jeder kann sie herunterladen. Um die Navigation zu erleichtern, gibt es auf der Download-Seite des Übungsmanagers eine baumförmige Struktur.



- **Forum.** Die registrierten Benutzer haben mit dem Forum die Möglichkeit ihre Übungsaufgaben und sonstigen Probleme online zu besprechen. Jeder User kann die Beiträge, die in diesem Forum stehen, anschauen. Die Einträge können nach speziellen Themen gruppiert werden.



- **Übungsaufgaben und ihre Korrektur.** Selbstverständlich verwaltet Übungsmanager auch Übungsaufgaben. Es gibt grundsätzlich zwei Typen von Übungsaufgaben: „normale“ Übungsaufgaben, die Dozenten und Tutoren erstellen und korrigieren müssen und „multiple choice“ -Aufgaben*, die automatisch erstellt und korrigiert werden. Jeder Student kann auf der Index-Seite des Übungsmanagers einen Überblick über die aktuellen und fertigen

Übungsaufgaben machen. Die Aufgaben werden „gesammelt“, indem jeder Student seine Lösung als Datei uploadet. Der Korrektor liest dann die Arbeit durch und trägt Korrekturdaten ein. Die „multiple-choice“ -Aufgaben werden automatisch aus den vorhandenen Schablonen erstellt und automatisch korrigiert. Die Statistik der erreichten Punktzahl aller Studenten in den Übungen wird automatisch erstellt und kann von einem registrierten User eingesehen werden. [* diese Funktion kennen wir aus dem Fach Softwaretechnik 3. Semester. Im Übungsmanager 0.65 haben wir aber bisher diese Funktion nicht gefunden. Vielleicht existiert ein Plugin?]

- **Einschreibung zur Klausur.** Der Dozent meldet die Prüfung (Klausur) an der Plattform an: er veröffentlicht den Termin und formuliert die Bedingungen der Zulassung. Beispielsweise erreichte Mindestpunktzahl in den Übungen. Die Studenten können sich dann für die Prüfung (Klausur) einschreiben. Die Einschreibung ist befristet. Nach der Kontrolle der Prüfung kann der Student seine erreichte Punktezahl im UebManager einsehen.
- **Infos.** Der Dozent hat in diesem Menü die Möglichkeit, allgemeine Informationen über die Veranstaltung in Form eines HTML-Textes zu erstellen. Jeder Student kann diese Informationen lesen, aber nicht bearbeiten. Zusätzlich kann der Dozent unter der Rubrik „News“ Hinweise, Änderungen oder aktuelle Termine veröffentlichen.
- **Webservice** ist ein Dienst für Softwaresysteme, der mit Hilfe von XML auf der Basis von Internet-Netzwerkprotokollen erbracht wird. Er kann automatisch Daten austauschen und Funktionen auf entfernten Rechnern aufrufen. Es existiert im Übungsmanager ein Menüpunkt „Webservices“, mit dem die oben genannten Funktionen ausgeführt werden können, jedoch wurde dies in der derzeitigen Version 0.65 noch nicht implementiert.

3.4. Einsatzgebiet

Grundsätzlich ist UebManager für den Hochschulbetrieb konzipiert und dient zur Verwaltung des Lehrbetriebs. Aber er könnte in allen anderen Schulformen (Berufsschule, Gymnasium, Mittelschule) eingesetzt werden. Hierbei handelt es sich nicht um eine fachspezifische Software, sondern sie kann in vielen Lehrstühlen verwendet werden. Die Zielgruppe dieses Produktes sind Studenten, Dozenten, Mitarbeiter einer Hochschuleinrichtung.

3.5. Leistungsparameter

- Die Daten (Benutzerdaten, Termine, Informationen über Lehrveranstaltungen, Gruppen, usw.) werden in XML-Dateien gespeichert. Die Dateien, die die Studenten up – bzw. downloaden, werden auf dem Server gespeichert.
- UebManager ist auf allen gängigsten Web-Bowsern darstellbar.
- Die Antwortzeit der Anwendung ist nicht durch das Programm selbst begrenzt, sondern durch die Hardware-Ressourcen (Server) und Internet-Geschwindigkeit.
- Genauere Angaben zu den Einschränkungen des Programms (beispielsweise Nutzermaximum, Datenmaximum usw.) kann man nicht machen, jedoch lief der UebManager von Softwaretechnik im WS2004-05 ohne große Komplikationen. Es ist aber bekannt, dass viele Studenten das System aktiv benutzt haben.

3.6. Anwendungsfälle des UebManager

Anwendungsfall: Anmelden
[evtl. Akteur]: Student
Beschreibung: Student meldet sich an
Ergebnis: Student erhält Account

Anwendungsfall: Einloggen
[evtl. Akteur]: Student, Tutor, Dozent, Admin
Beschreibung: Student, Tutor, Dozent, Admin meldet sich an
Ergebnis: angemeldeter Student, Tutor, Dozent, Admin

Anwendungsfall: Benutzerverwaltung
[evtl. Akteur]: Dozent, Admin
Beschreibung: Dozent, Admin verwaltet Benutzerdaten, d.h. erstellt, ändert und löscht diese
Ergebnis: Erstellte, geänderte und gelöschte Benutzerdaten

Anwendungsfall: Gruppenverwaltung
[evtl. Akteur]: Dozent, Admin
Beschreibung: Dozent, Admin verwaltet Gruppen, d.h. erstellt, ändert und löscht diese
Ergebnis: Erstellte, geänderte und gelöschte Gruppen

Anwendungsfall: Dateiverwaltung
[evtl. Akteur]: Dozent, Admin
Beschreibung: Dozent, Admin verwaltet Dateien oder Ordner, d.h. erstellt, ändert, löscht, beschreibt diese
Ergebnis: Erstellte, geänderte, gelöschte, beschriebene Datei oder Ordner

Anwendungsfall: Konfiguration
[evtl. Akteur]: Dozent, Admin
Beschreibung: Dozent, Admin konfiguriert den Übungsmanager selbst.
Ergebnis: Konfiguriert den Übungsmanager

Anwendungsfall: Übungsserienverwaltung
[evtl. Akteur]: Dozent
Beschreibung: Dozent verwaltet Übungsserien, d.h. lädt diese hoch und schränkt sie nach Abgabedatum und Punktezahl ein
Ergebnis: Zur Verfügung gestellte Übungsserien

Anwendungsfall: Übungsserienabgabe
[evtl. Akteur]: Student
Beschreibung: Student lädt bearbeitete Übungsserien hoch
Ergebnis: Zur Korrektur gestellte Übungsserien

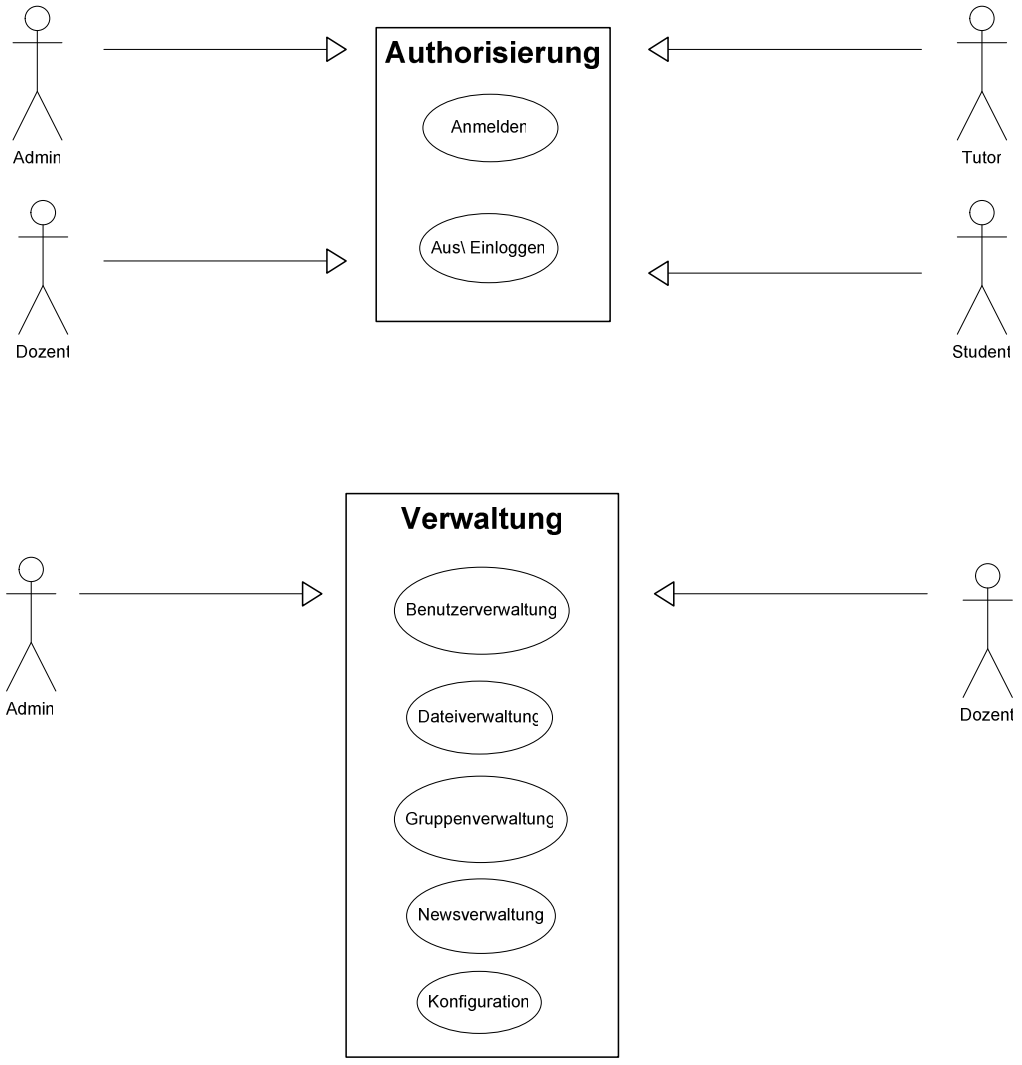
Anwendungsfall: Korrektureinsicht und Bewertung
[evtl. Akteur]: Dozent, Tutor
Beschreibung: Dozent, Tutor sieht in die Korrekturen und Statistik ein
Ergebnis: Bewertete Übungsserien und eingesehene Statistik

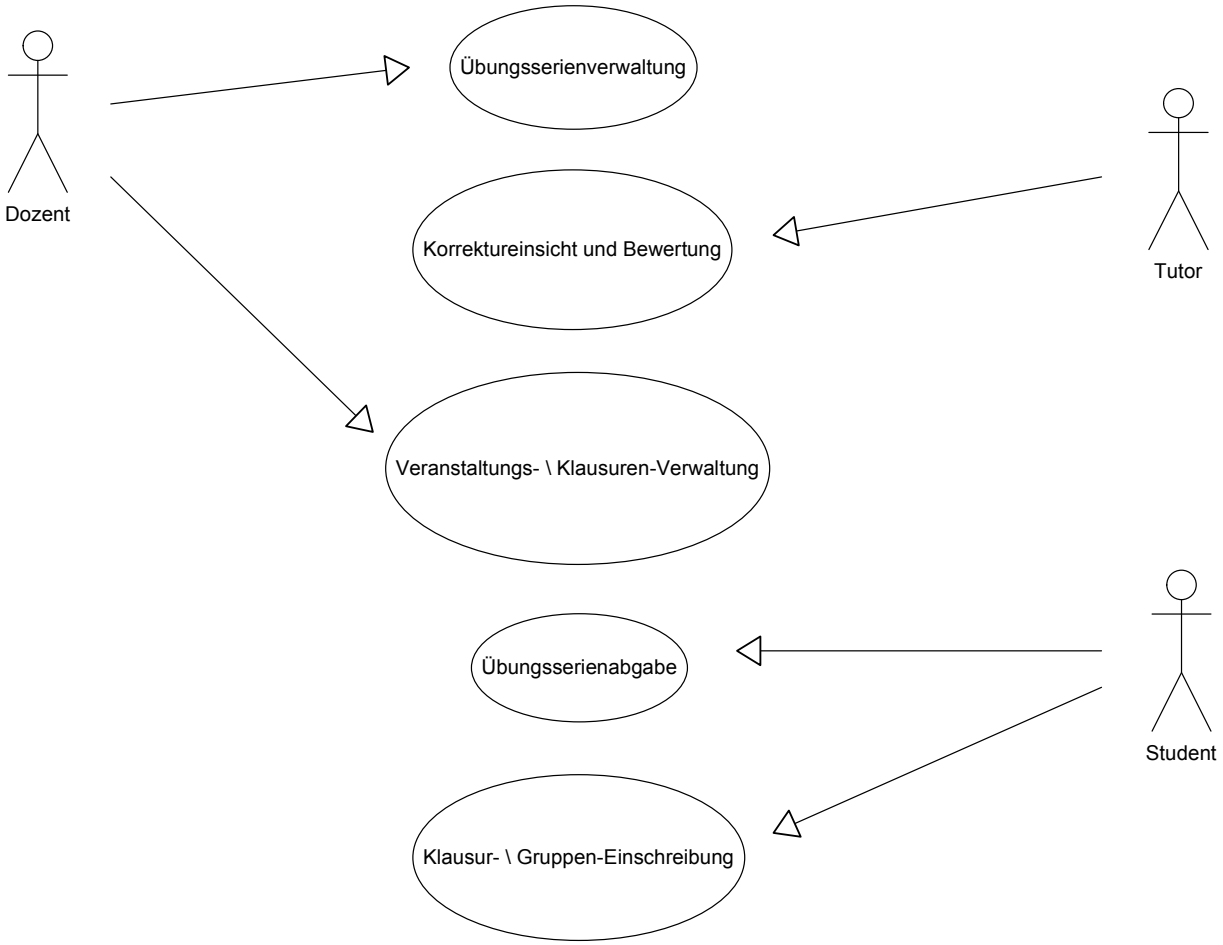
Anwendungsfall: Newsverwaltung
[evtl. Akteur]: Dozent
Beschreibung: Dozent verwaltet News, d.h. erstellt, ändert, löscht diese
Ergebnis: Erstellte, geänderte und gelöschte News

Anwendungsfall: Veranstaltungs- \ Klausuren-Verwaltung
[evtl. Akteur]: Dozent
Beschreibung: Dozent verwaltet Veranstaltungen oder Klausuren, d.h. erstellt, ändert, löscht, beschreibt diese
Ergebnis: Erstellte (zur Verfügung gestellte), geänderte und gelöschte Veranstaltungen oder Klausuren

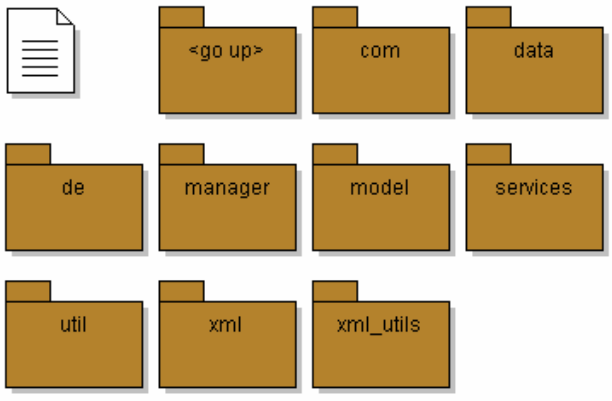
Anwendungsfall: Klausur- \ Gruppen-Einschreibung
[evtl. Akteur]: Student
Beschreibung: Student schreibt sich in die Gruppe ein/ meldet sich für die Klausur an
Ergebnis: Eingeschriebener \ angemeldeter Student

3.7. Use Case Diagramme



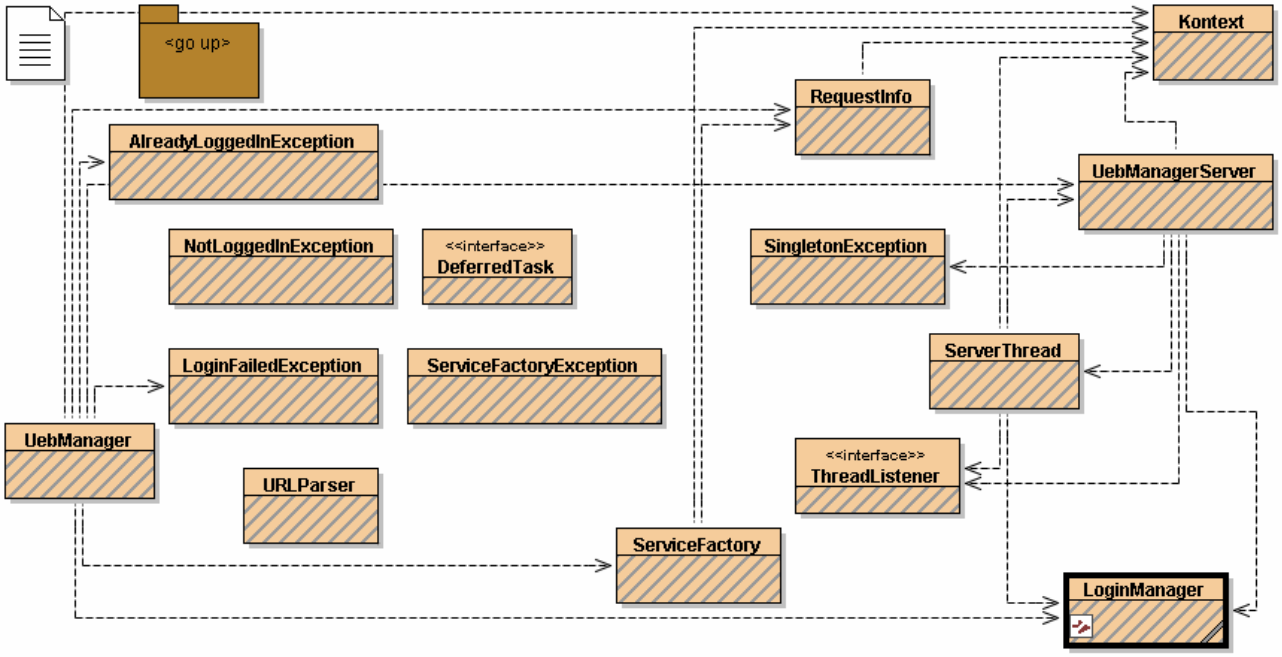


3.8. Allgemeine Struktur des UebManagers



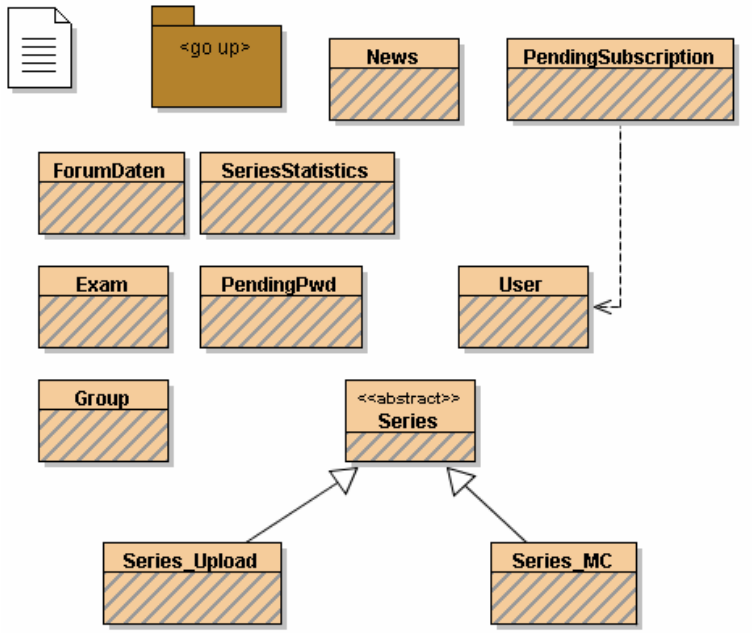
Hauptklassen:

- Manager



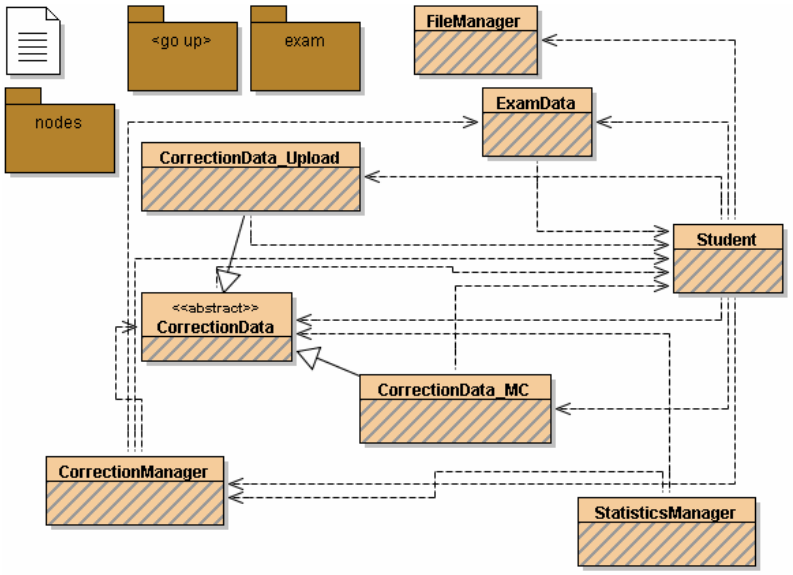
Class Summary	
Kontext	Alle Informationen zur Identifizierung eines Benutzers innerhalb einer Session über mehrere Requests hinweg werden hier gespeichert.
LoginManager	Diese Klasse verwaltet alle eingeloggten User (inkl. login/logout), überprüft die Authentifizierung des Benutzers gegen die XML-Benutzer-DB und legt ein entsprechendes Kontext-Objekt an.
RequestInfo	Diese Klasse enthält Informationen, die mit genau einem Request in Zusammenhang stehen
ServerThread	Thread für administrative Aufgaben, u.a. idle User ausloggen, verfallene Einschreibungen löschen und Garbage Collection.
ServiceFactory	Hier wird das Factory-Pattern implementiert, das aufgrund der Auswertung von Request- Methode, einer bereits erfolgten Authentifizierung, der Rolle des eingeloggten Benutzers und des Inhalts der action-URL-Variable den konkreten Service an den UebManager liefert.
UebManager	Dieses Servlet ist die Basis unserer gesamten Übungsgruppenverwaltung.
UebManagerServer	Zentrale Server-Instanz.
URLParser	Experimenteller Parser für URL-Rewriting.

- Model



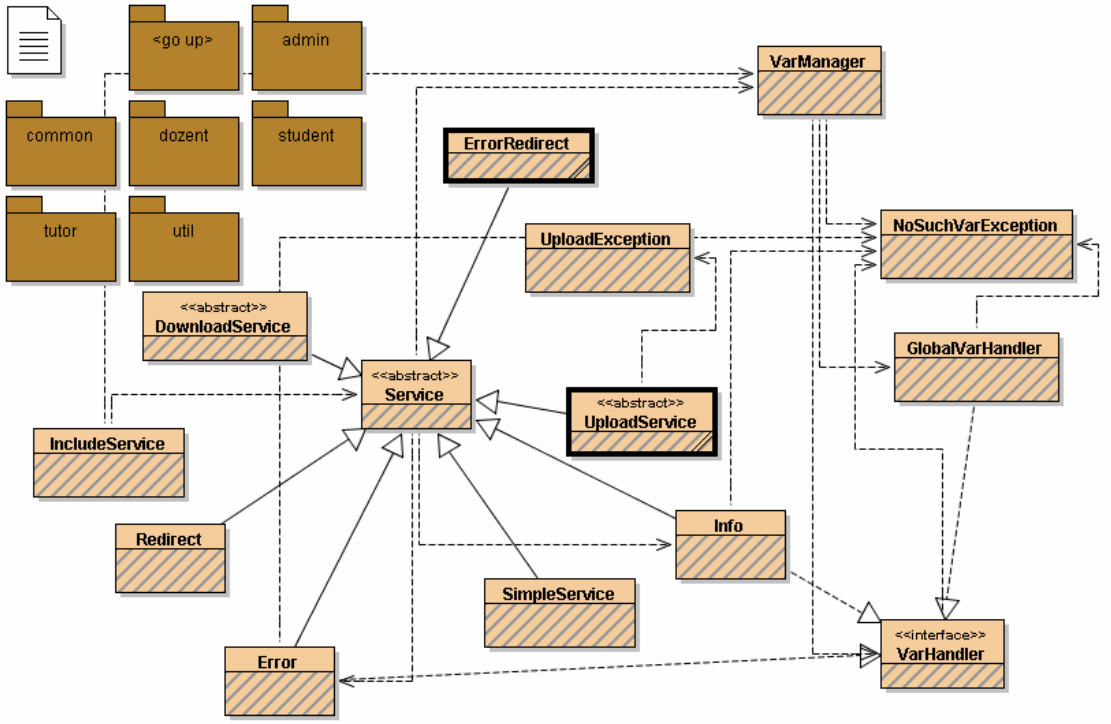
Class Summary	
Exam	Repräsentiert eine Klausur.
Group	Repräsentiert Daten einer Übungsgruppe.
News	Repräsentiert News-Beitrag.
PendingPwd	Repräsentiert Passwort-Änderung.
PendingSubscription	Repräsentiert Einschreibungs-Daten.
Series	Repräsentiert Übungsserie.
SeriesStatistics	Repräsentiert die Korrektur-Statistik einer Serie
User	Einweg-Klasse zum Austausch von User-Infos

- Data



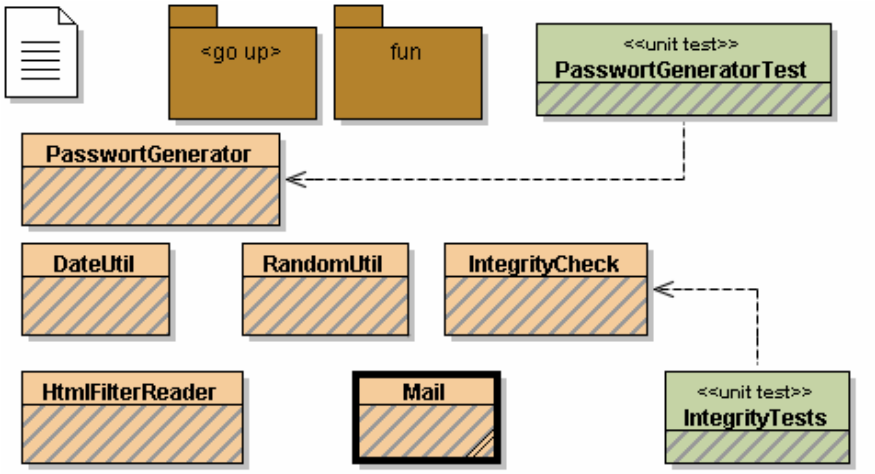
Class Summary	
CorrectionData	Repräsentiert die Korrekturinfos einer Serie eines Studenten.
CorrectionData_MC	Das konkrete CorrectionData Objekt für MultipleChoice-Aufgaben.
CorrectionManager	Der CorrectionManager verwaltet intern Studentenobjekte.
FileManager	FileManager verwaltet Directory- und FileNodes im Data-Verzeichnis.
StatisticsManager	Erstellt Statistiken über die Korrektur der Übungsserien.

- Services



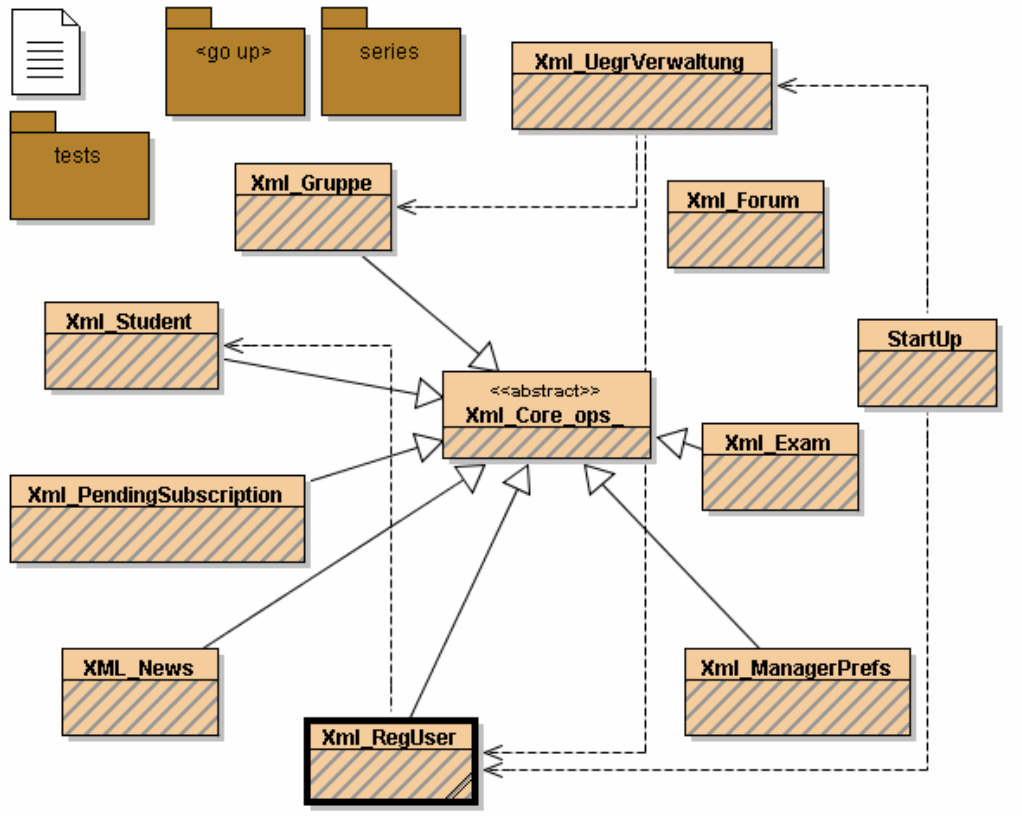
Class Summary	
DownloadService	Abstrakter DownloadService als Oberklasse aller Services, die (binäre) Files handlen.
Error	Service zur Anzeige von Fehlermeldungen
GlobalVarHandler	Expandiert globale Variablen.
IncludeService	Ein IncludeService schreibt nicht in den Out-Stream/Writer, sondern gibt nur einen String zurück.
Info	Ähnlich Error zeigt dieser Service eine Meldung an.
Service	Oberklasse aller konkreten Services.
SimpleService	Dieser Service gibt einfach nur Strings oder ein HTML-Template aus.
UploadService	Abstrakte Oberklasse für Services, die Dateien hochladen wollen.
VarManager	Hier befinden sich Methoden zum Parsen von HTML-Templates und zum Expandieren der darin enthaltenen Variablen.

- Util



Class Summary	
DateUtil	Verschiedene Methoden zum Date-Handling
HtmlFilterReader	HtmlFilterReader Quelle: http://www.java-tutor.com
IntegrityCheck	Diese Klasse enthält Integritätschecks.
Mail	Mails verschicken m.H. der JavaMail-API
PasswortGenerator	Diese Klasse enthält Methoden zur Generierung neuer Passwörter.

- XML



Class Summary	
Xml_Core_ops	stellt die Kern Funktionen im Umgang mit Sax und Dom zur Verfügung.
Xml_Gruppe	realisiert das Verhalten einer einzelnen Übungsgruppe.
Xml_PendingSubscription	Klasse, die zum temporären Abspeichern von noch nicht bestätigten Userdaten, z.B. nach der Anmeldung, warten auf das Confirm der Mail, und zum anderen für die Zwischenspeicherung mgl. neuer Passwörter, die vom lostpwd-service kommen.
Xml_RegUser	Stellt die Methoden zum Auslesen.
Xml_Student	Diese Klasse handelt nur die scores.xml Geschichten
Xml_UegrVerwaltung	Verwaltet alle Übungsgruppen (Einschreibungen, Wechsel, etc).