# Entwurfsbeschreibung des Prüfungsmoduls

Projektgruppe GR-07-1

21. Mai 2007

## Inhaltsverzeichnis


**1 Allgemeines** **2**
- 1.1 Charakterisierung . . . 2
- 1.2 Systemvoraussetzungen . . . 2
- 1.3 Abgrenzung . . . 2
- 1.4 Allgemein . . . 3

**2 Grundsätzliches Designentscheidungen** **4**
- 2.1 Fachkonzepte . . . 4
  - 2.1.1 MVC . . . 4
  - 2.1.2 Velocity . . . 4
  - 2.1.3 J2EE . . . 5
  - 2.1.4 Tomcat . . . 5
  - 2.1.5 XML . . . 5
- 2.2 Rollenmanagement in der Prüfungsextension . . . 6
- 2.3 Darstellung ausgewählter Szenarien . . . 6

**3 Paket- und Klassenstruktur** **9**
- 3.1 Erweiterungskonzept und dessen Realisierung . . . 9
- 3.2 Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem . . . 11
- 3.3 Grundsätzliche Struktur- und Entwurfsprinzipien einzelner Pakete . . . 12

# 1 Allgemeines

## 1.1 Charakterisierung

Das Prüfungsmodul realisiert die Prüfungsverwaltung, wie sie zur Zeit an der Universität Leipzig, am Institut für Informatik durchgeführt wird. Die Benutzer die dieses System nutzen möchten, können sich an diesem registrieren und anmelden. Es besteht die Möglichkeit sich als Student, Prüfer, oder als Prüfungsamt zu registrieren. Für jede dieser Rollen gibt es verschiedene Funktionalitäten (gesichert über spezielle Rechte für diese Rollen). Zum Beispiel kann ein Prüfer ein Prüfungsangebot erstellen, welches dann vom Prüfungsamt genehmigt werden muss. Ein Student hat nun die Möglichkeit sich zu einer freigeschalteten Prüfung anzumelden oder auch wieder abzumelden. Das Prüfungsamt hingegen hat die meisten Zugriffsrechte neben dem Administrator des Systems. Es wird strikt zwischen schriftlichen und mündlichen Prüfungen getrennt.

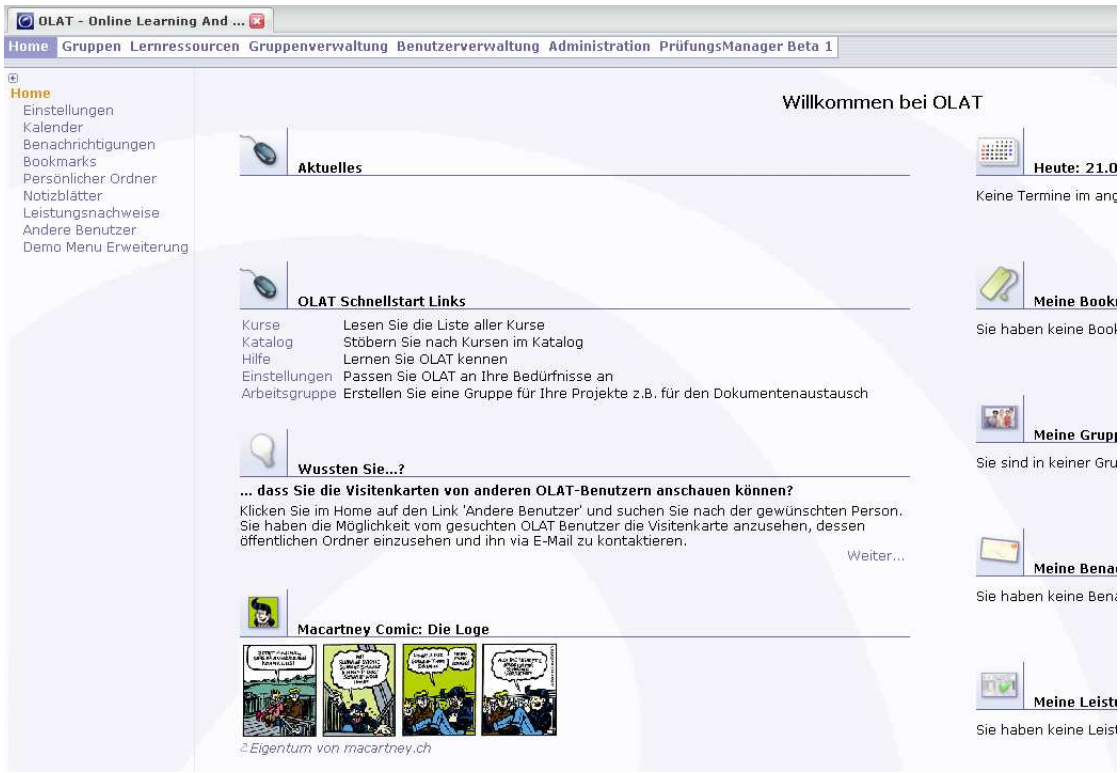
## 1.2 Systemvoraussetzungen

Da das Modul vollständig in Java Implementiert wird und zur Persistierung eine MySQL Datenbank sowie XML-Dateien verwendet werden, kann das Modul (und OLAT) auf einem Window-, Unix-, Linux- oder Mac-Server laufen. Auf diesem Server (leistungsfähige Workstation oder Computer der Serverklasse) sollten alle benötigten Softwarepakete wie z.B. MySQL, JRE, Tomcat, ... installiert sein. Des Weiteren sollte der Server über einen schnellen Breitbandanschluss zum Internet verfügen, damit die geforderten Reaktionszeiten an das Systems auch eingehalten werden können. Clientseitig ist nur ein Webbrowser (wie Firefox, Internet Explorer, ...) und eine Internetverbindung von Nöten.

## 1.3 Abgrenzung

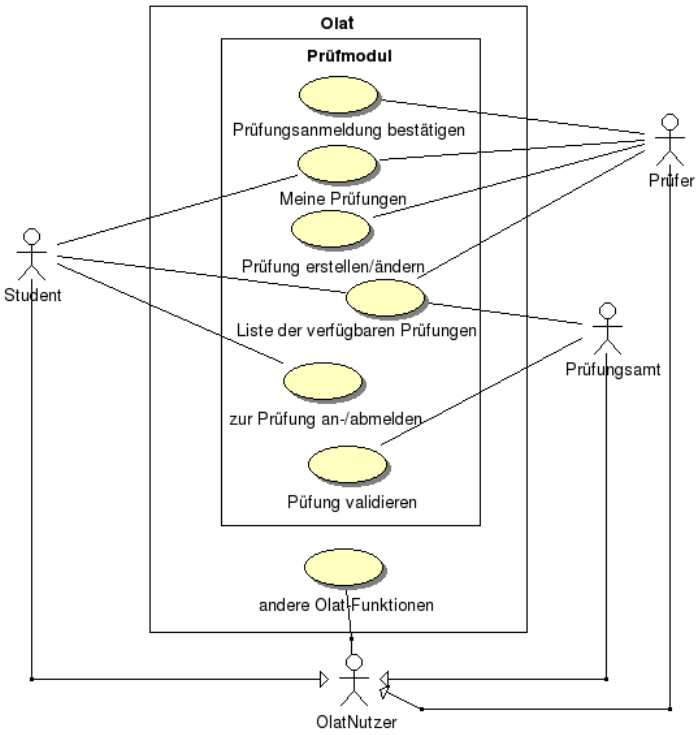
Die gesamte Sicherheit der Rechte- und Prüfungsverwaltung in der Erweiterung ist von Olat abhängig, d.h. es wird keine Bemühungen geben, Prüfungsamt- und Prüferaccounts zu sichern, oder Prüfungen und dazugehörige angemeldete Studenten bzw. Daten zu verschlüsseln.

## 1.4 Allgemein




Das Prüfungsmodul wird als eine Erweiterung in OLAT integriert. D.h. alle Funktionen des Prüfmoduls werden jedem am Olat-System angemeldeten Nutzer zur Verfügung gestellt. Die OLAT-Erweiterung wird zwischen 3 Rollen unterscheiden, die sich in ihren Rechten und Funktionsmöglichkeiten unterscheiden. Die Rolle Student nimmt jeder normale User ein, die Rollen Prüfungsamt und Prüfer ergeben sich dann mittels eines Rechtekonzepts als Spezialfall aus den normalen Usern. Auf diese Weise wird keine neue Rolle im Gesamtsystem Olat nötig sein.

- **Student** Der normale am System angemeldete Nutzer nimmt die Rolle Student ein und erhält damit die Möglichkeiten sich Prüfungsinformationen zu allen veröffentlichen und insbesondere zu den eigenen Prüfungen anzeigen zu lassen. Als weitere wichtige Funktion wird er außerdem in der Lage sein, sich zu Prüfungen an - und abzumelden.
- **Prüfer** Mittels eines Rechtemanagements werden ausgewählten Nutzern die Funktionen eines Prüfers ermöglicht, d.h. neben den allgemeinen Info-Funkionen, die der normale Student auch besitzt, kann ein Prüfer Prüfungen veröffentlichen und Anmeldungen für validierte Prüfungen erstellen und ändern.
- **Prüfungsamt** Die Rolle Prüfungsamt besitzt die Möglichkeit, von Prüfern erstellte Prüfungen zu validieren, d.h. für alle Nutzer des Prüfungsmoduls zu veröffentlichen. Ein Prüfer wird außerdem in der Lage sein, die Anmeldung von Studenten zu bestätigen oder abzulehnen.




# 2 Grundsätzliche Designentscheidungen

## 2.1 Fachkonzepte

Es werden folgende Fachkonzepte bzw. Technologien eingesetzt:

### 2.1.1 MVC

Das MVC-Paradigma ist ein grundlegendes Architekturkonzept, das Modell, Controller und View (Darstellung) klar voneinander trennt. Dabei können die einzelnen Komponenten nicht beliebig miteinander kommunizieren, sondern gehorchen dem MVC-Modell, so dass die einzelnen Teile möglichst unabhängig voneinander sind. Diese Trennung macht zum einen nahezu paralleles Arbeiten an diesen Teilen möglich, unterstützt eine hohe Wiederverwertbarkeit sowie Wartbarkeit und macht es möglich, relativ leicht, Änderungen und Erweiterungen vorzunehmen.
Das ganze Prüfmodul wird stark MVC-orientiert entwickelt, was schon in der Aufteilung der Pakete eine große Rolle spielt.
(siehe Paketdiagramm weiter unten)

### 2.1.2 Velocity

Velocity ist ein Open-Source-Projekt und entstammt der Apache Jakarta-Projektgruppe. Es handelt sich hierbei um ein Java-basierte Template-Engine, die es ermöglicht Java-Objekte in verschiedenste andere Formate, wie Text, XML, SQL, Post Script oder HTML einzubinden. Da diese streng nach dem MVC-Konzept konzipiert wurde und nicht sonderlich schwer zu erlernen ist, erlaubt Velocity quasi paralleles Arbeiten an funktionalem Code und einem ansprechenden Äußeren. Die Hauptanwendungen von Velocity sind XML-Transformationen, das Erstellen von dynamischen

Webseiten und Webanwendungen. OLAT nutzt Velocity aufbauend auf der Präsentationslogik, so dass der Client die erhaltenen Daten nur noch darstellen muss.

### 2.1.3 J2EE

Die Java (2) Platform, Enterprise Edition ist eine Entwicklungs- und Laufzeitumgebung von verteilten Anwendungen, die auf einer n-Schichten Architektur beruhen. Hierfür stellt die Plattform eine Vielzahl von APIs zur Verfügung, um etwa Servlets, EJBs oder RMIs etc. erzeugen zu können. Desweiteren beinhaltet die J(2)EE Spezifikationen und Standards, damit die entickelten Programme auf allen möglichen Applicationservern ausführbar bleiben. J2EE stellt also auch die Standards und die meisten Werkzeuge für OLAT und somit für dieses Projekt zur Verfügung.

### 2.1.4 Tomcat

Tomcat bezeichnet einen Open-Source Webserver, welcher den Servlet-Container Catalina beinhaltet. Catalina ist also eine Laufzeitumgebung für Servlets, die bei Apache innerhalb des Jakarta-Projektes entwickelt wird. Er wurde vorallem für JavaServer Pages konzipiert und nutzt hierzu den JSP-Compiler Jasper. Tomcat bietet neben dem Servlet-Container Catalina und einem HTTP-Server auch den Connector Coyote an, um die Kommunikation mit anderen Servern zu ermöglichen. Da Tomcat Open-Source ist, also nichts kostet und recht benutzerfreundlich ist, hat er sich als Applicationserver etabliert.

### 2.1.5 XML

Bei XML handelt es sich um eine Auszeichnungssprache („Extensible Markup Language“) für Datenrepräsentation in Textform. Dabei stellt XML nur eine Metasprache dar, aus der mittels Schemasprachen wie DTD die eigentliche Sprache gewonnen wird. Zu den auf XML basierenden Sprachen gehören u.a. XHTML, WML und RSS. Die repräsentierten Daten sind relativ leicht durch Maschinen zu verarbeiten, bleiben aber auch für Menschen les- und änderbar. In diesem Projekt wird XML vorwiegend zur Sicherung der Prüfungsdaten etc. genutzt, wobei wieder möglichst auf Nutzung von bereitgestellten Olat-Schnittstellen geachtet wird

## 2.2 Rollenmanagement in der Prüfungsextension

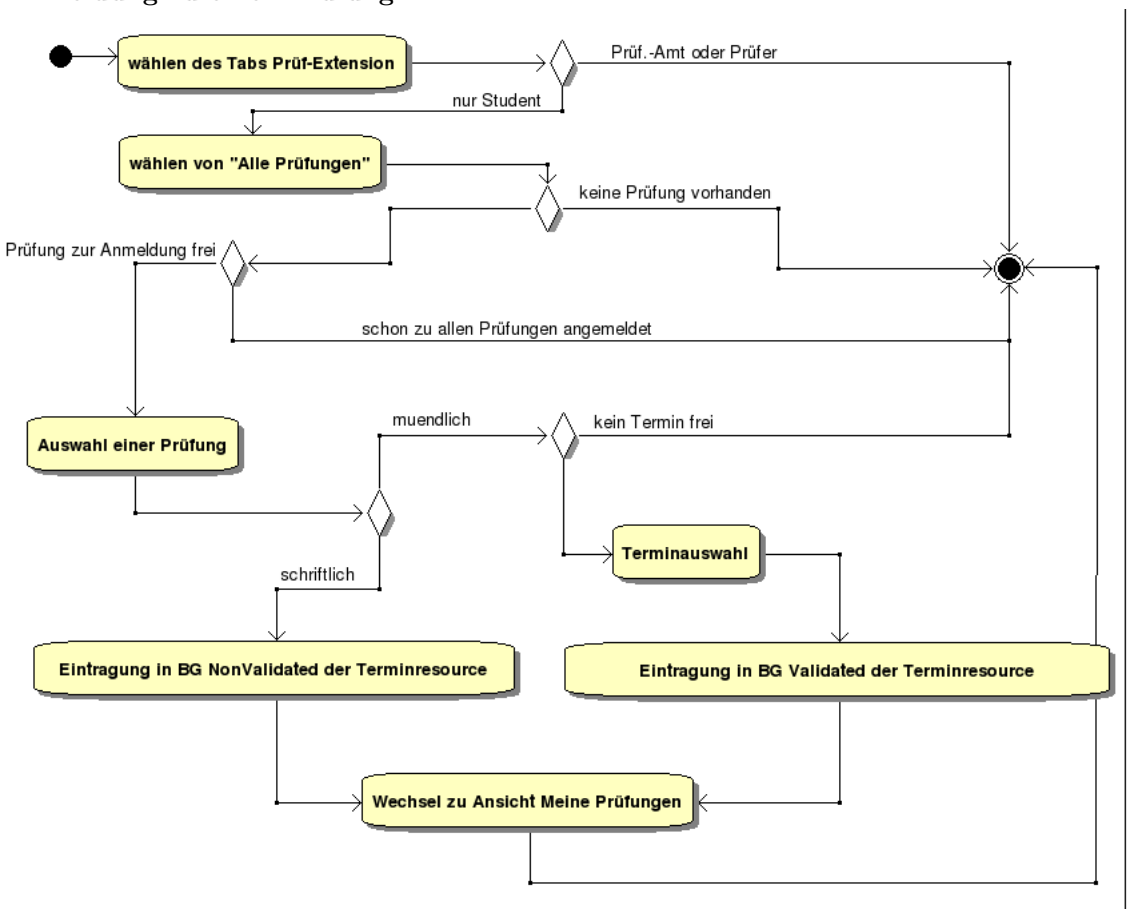
Es wird grundsätzlich zwei systemweite SecurityGroups geben, eine für Prüfer und eine für das Prüfungsamt. Die Rollen Prüfer/Prüfungsamt werden vergeben, in dem der Olat-Nutzer in diese SecurityGroups aufgenommen wird.

Desweiteren gibt es für jede Prüfung wiederum genau zwei weitere SecurityGroups. Die eine bezeichnet die Besitzer der Prüfung, und damit die "Prüfer der Prüfung", die andere die Korrektoren der Prüfung, die Ergebnisse eintragen dürfen.

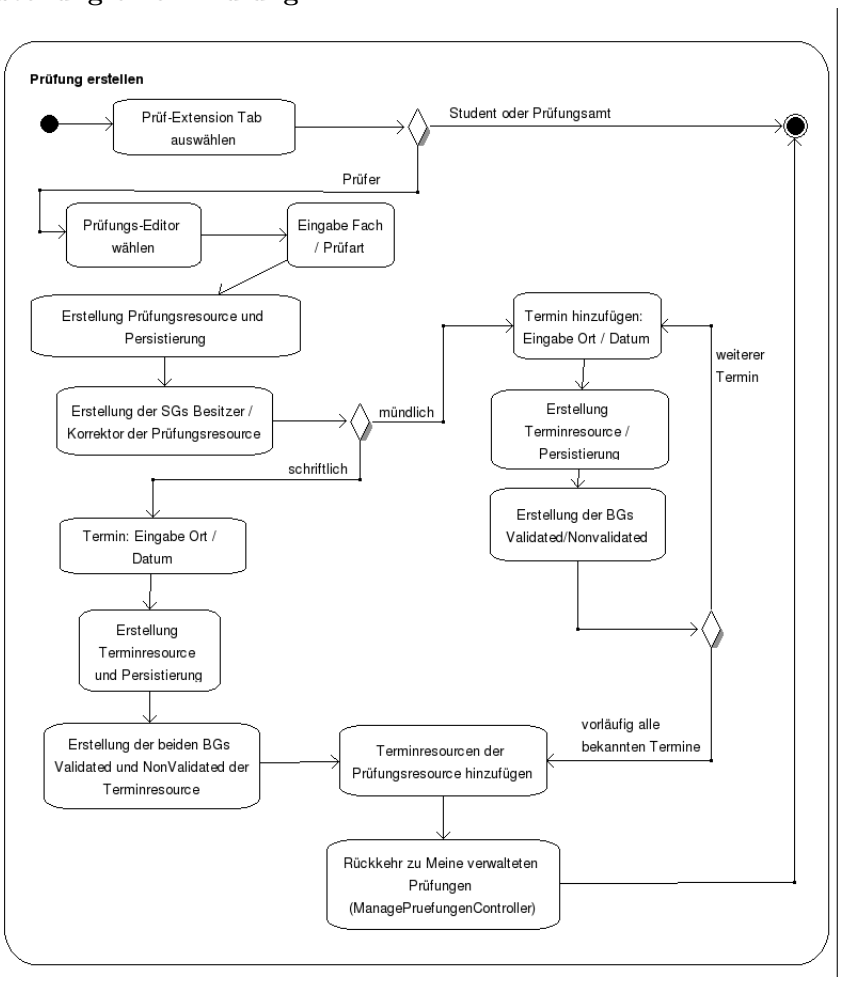
Wie im Prototypen werden je einer Prüfung Termine zugeordnet. Dabei werden die eingeschriebenen Studenten nunmehr nicht im XML gespeichert, sondern werden in jeweils einer der beiden, für jeden Termin spezifisch vorhandenen, BusinessGroups "Validated" bzw "NonValidated"aufgenommen. Diese symbolisieren zum einen die Studenten die sich angemeldet haben und den Nachweis erbracht haben die schriftliche Prüfung schreiben zu dürfen, bzw vom Prüfer freigeschaltet wurden, bzw alle Teilnehmer einer mündlichen Prüfung. Zum anderen die Studenten die noch auf eine Freischaltung für die Prüfung durch den Prüfer warten.

## 2.3 Darstellung ausgewählter Szenarien

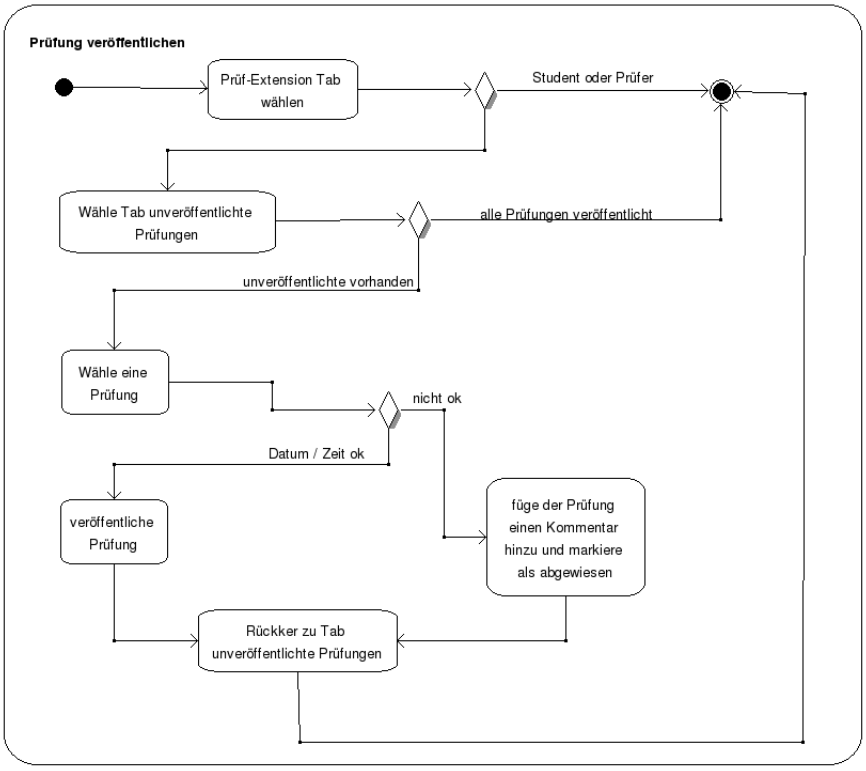
### Anmeldung zu einer Prüfung



### Erstellung einer Prüfung



### Veröffentlichung einer Prüfung



# 3 Paket- und Klassenstruktur

## 3.1 Erweiterungskonzept und dessen Realisierung

Man hat sich dazu entschlossen unser Prüfungsmodul als Erweiterung (Extension) in OLAT zu integrieren. Dies ist die sauberste Lösung, um OLAT bezüglich neuer Funktionalitäten zu erweitern. Dazu wird an den OLAT-Sourcen nichts geändert. Im Folgenden erhalten Sie einen Überblick über die Vorgehensweise.

Zunächst wird das Prüfungsmodul implementiert. Dabei leitet man sich eine neue Klasse von DefaultController ab, welcher die Anwendungslogik steuert. Dazu gehört unter anderem das Klicken auf eine Schaltfläche, das Auswählen einer Option in einer Auswahlbox usw. Des weiteren kann ein PackageTranslator eingefügt werden, welcher die Internationalisierung realisiert. Um die Daten (Model) in HTML-Elemente (View) zu transformieren benötigen wir noch einen Velocity-Container. Mit

```
myContent =  new VelocityContainer("helloworldvc",
                                   VELOCITY_ROOT + "/helloworld.html",
                                   translator,
                                   this);
```

ist es möglich, eine neue Seite zu erzeugen. Die fertig gerenderte Seite ist dann helloworld.html. Den Pfad zu den Velocity Templates erhält man mit:

```
private static final String VELOCITY_ROOT =
  Util.getPackageVelocityRoot(HelloWorldController.class);
```

Nun können die Daten und das Template gemerged werden:

```
myContent.contextPut("myContentVariable", myString );
```

Das Projekt muss folgende Struktur aufweisen:

```
de.softadapt.pruefmodul
 controller
 _content
 _i18n
 _static
 PruefungsManagerExtension.java
```

Im Ordner _i18n befinden sich die Daten für die oben genannte Internationalisierung. Zum Schluss wird das Projekt noch als .jar-Archiv gepackt und kann mit wenig Aufwand in OLAT integriert werden. Dazu aktualisieren wir folgende Datei:

```
<OLAT_DIR>/webapp/WEB-INF/lib/olat_extensions.xml
```

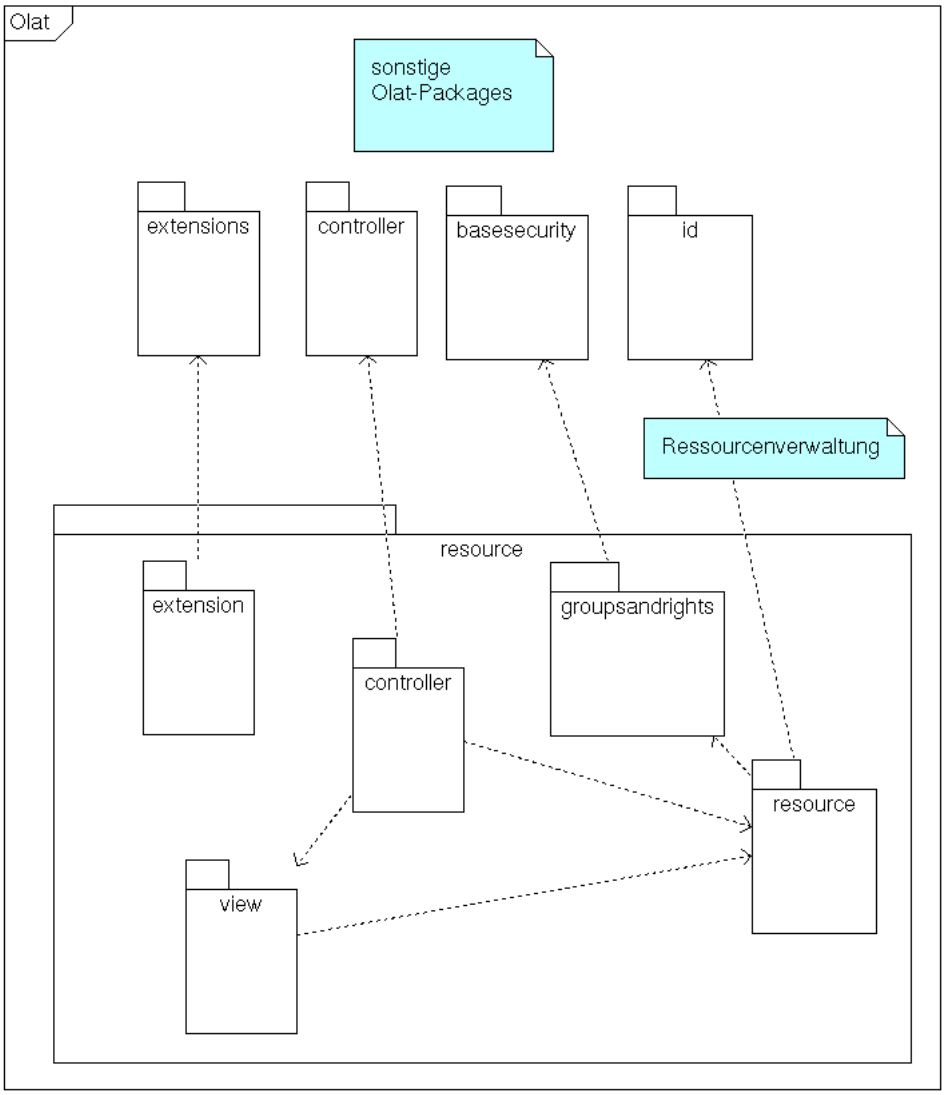
Hier ein Ausschnitt inklusive der Änderung:

```
...

<!-- generic OLAT extensions -->
<bean id="extmanager" class="org.olat.core.extensions.ExtManager" singleton="true">
<property name="extensions">
```

```
<list>
<!-- classes implementing the Extension interface -->
<bean id="pruefungsmanagerextension"
  class="de.softadapt.pruefmodul.extension.PruefmodulExtension" singleton="true" />
<!-- <bean id="simpleExtension"
  class="com.xyz.demoextension.HelloWorldExtension" singleton="true" /> -->
<!-- <bean id="combineExtension"
  class="ch.unizh.campusmgnt.SAPCampusMgntExtension" singleton="true" /> -->
</list>
</property>
</bean>

...
```

In diesem speziellen Beispiel wurde **de.softadapt.pruefmodul.extension.PruefmodulExtension** freigeschalten.

## 3.2 Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem



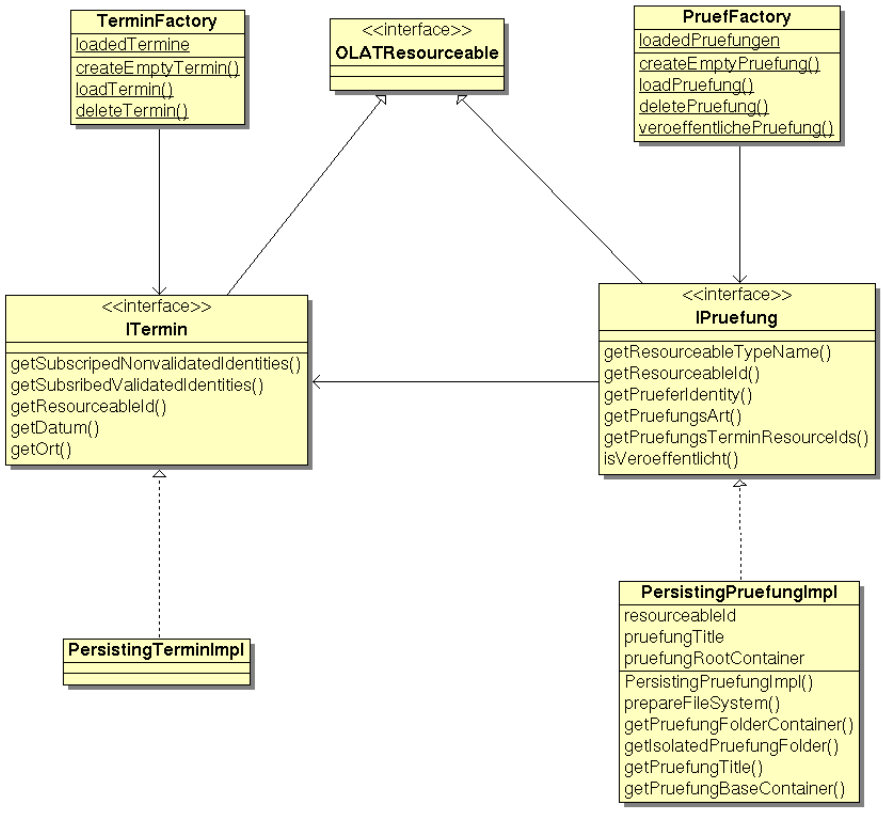

Dieses Diagramm zeigt alle Pakete der Erweiterung und die wichtigsten Schnittstellen in Olat. Die Pakete **de.softadapt.pruefmodul.extension** und **de.softadapt.pruefmodul.controller** realisieren bzw. erweitern die entsprechende Olat-Schnittstelle, um überhaupt eine Anbindung an das System zu ermöglichen.
Das Paket **de.softadapt.pruefmodul.groupsandrights** wird sich mit Hilfe von **org.olat.basesecurity** um die Rechteverteilung und die verschiedenen Rollen kümmern.
Als weitere wichtige Schnittstelle dient **org.olat.core.id**, um eine Resourcenverwaltung möglich zu machen, die hier für Prüfungen und Termine genutzt wird.
Diese Resourcen werden alle im Paket **de.softadapt.pruefmodul.resources** zusammengefasst.
Als letztes Paket in Prüfmodul wird sich **de.softadapt.pruefmodul.view** um die Darstellung und Ausgabe der Daten kümmern.
Die Klassen aus den Paketen **de.softadapt.pruefmodul.groupsandrights** und **de.softadapt.pruefmodul.resources** stellen im MVC-Konzept das Modell dar,

während **de.softadapt.pruefmodul.controller** und **de.softadapt.pruefmodul.view** die Aufgaben des Contollers bzw. des Views übernehmen. Durch die MVC-Konforme Einteilung soll ein möglichst modularer Aufbau erreicht werden, um Änderungen und Erweiterungen unkompliziert vornehmen zu können.

## 3.3 Grundsätzliche Struktur- und Entwurfsprinzipien einzelner Pakete

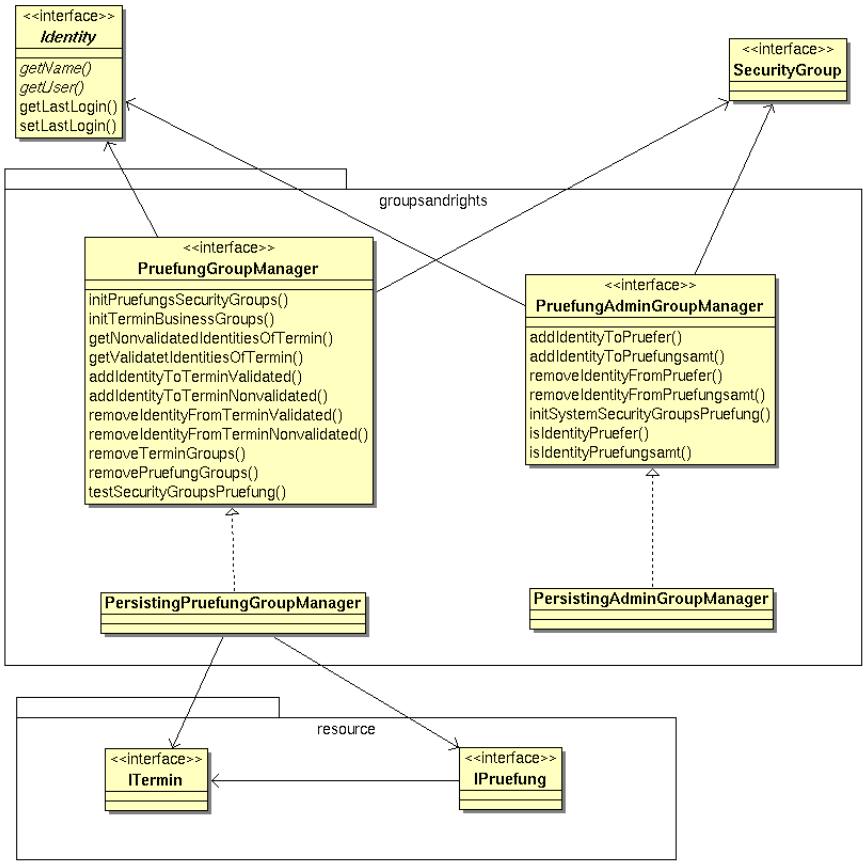
Das erste Klassen-Diagramm ist eng an die Resourcen im Olat angelehnt. Als Vorlage diente uns die Handhabe der Kurs-Resourcen. Mit der Factory lässt sich jeweils eine neue Instanz der Klasse entweder komplett neu erstellen oder eine vorhandene nach der Resourcen-ID laden. Dabei ist das Interface streng von der Implementierung getrennt, die je nach Persistence-Schicht variieren kann. Es wird jedoch bei uns auch eine XML-Dateisystem-Struktur dahinter liegen, genau wie es bisher im Olat gehandhabt wird.

**de.softadapt.pruefmodul.resources**



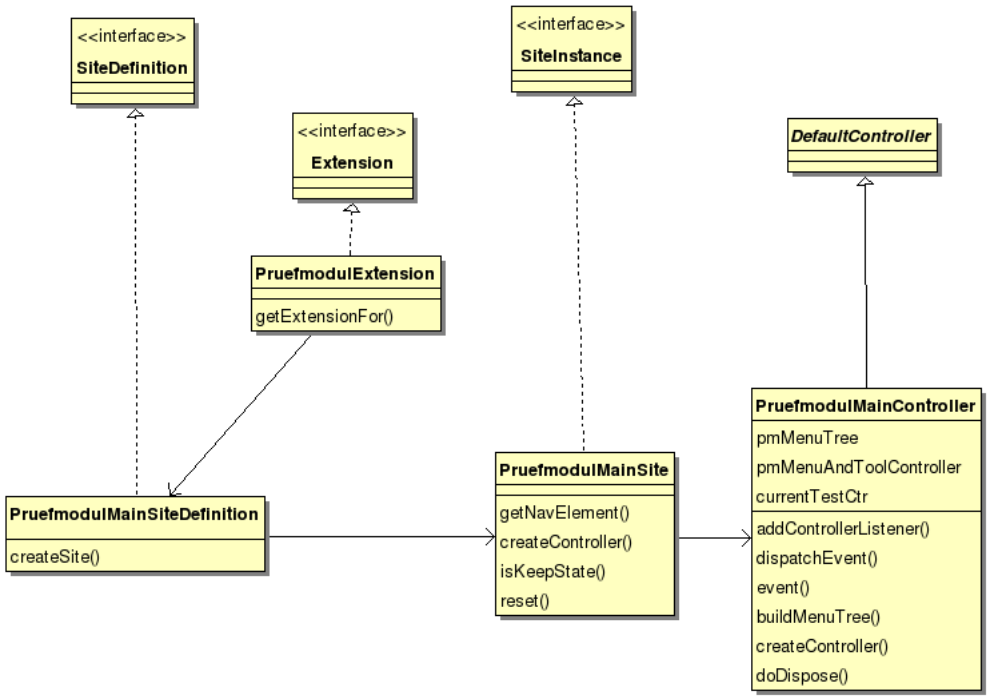

Die folgenden Klassen sollen für das Management der Security-Groups der Prüfungs-Resourcen und der Business-Groups der Termin-Resourcen zuständig sein. Somit sind alle Operationen die sich auf das Rechte und Gruppenmanagement beziehen, an zentraler Stelle vereint und können leicht koordiniert werden.

**de.softadapt.pruefmodul.groupsandrights**




Das folgende Diagramm liefert einen zentralen Überblick über die Extension und deren Funktion. Alle GUI-Events innerhalb des MenuTree der Prüfungs-Extension werden vom Maincontroller abgefangen und an die entsprechenden spezifischen Controller weitergereicht. Somit stellt der Main-Controller die zentrale Steuereinheit dar.

**de.softadapt.pruefmodul.extension**




Nun werden alle wichtigen Controller vorgestellt die sich bisher in unserem Model befinden. Jeder Controller ist in etwa für einen Geschäftsprozess zuständig, und somit auch zum Großteil einer speziellen Rolle vorbehalten. Man kann dies wie folgt aufschlüsseln:

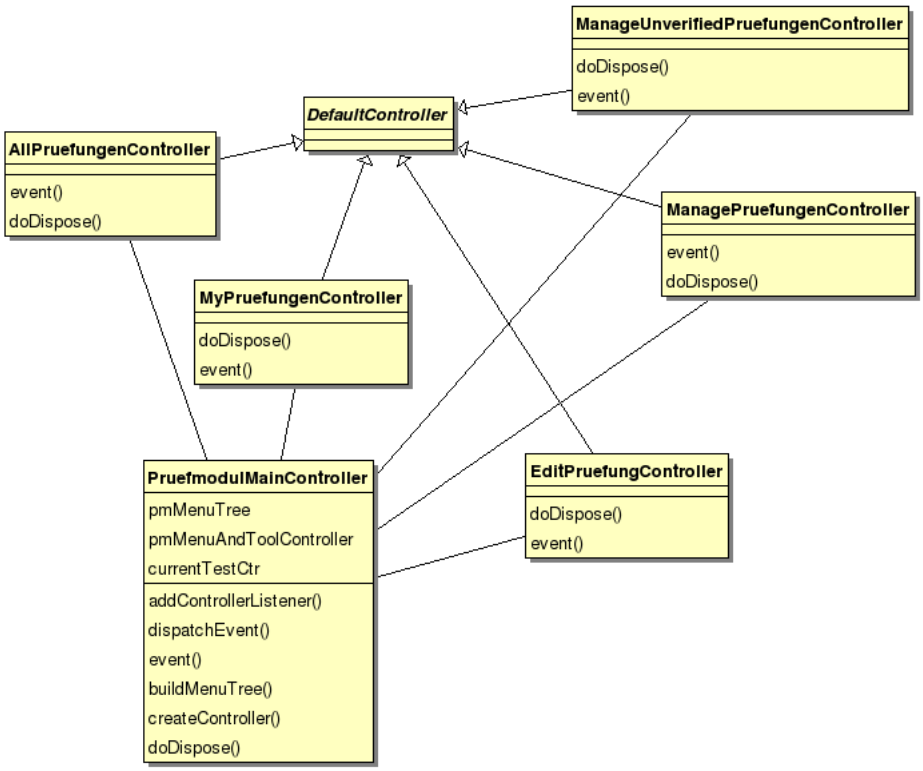
**AllPruefungenController**:
**Alle Nutzer:** Ansicht aller im Olat erstellten UND freigegebenen Prüfungen
**Student(kein Prüfer / Prüfungsamt):** Auswahl einer Prüfung zur Einschreibung

**MyPruefungenController**
**Student(kein Prüfer / Prüfungsamt):** Ansicht der Prüfungen für die sich der Student eingeschrieben hat, sowie Austragung aus diesen

**EditPruefungControlle**
**Prüfer:** Erstellungs- / Bearbeitungsformular für Prüfungen, Anmeldebestätigung für Studenten ( Freischaltung)

**ManagePruefungenController**
**Prüfer:** Ansicht der eigenen erstellten Prüfungen

**ManageUnverifiedPruefungenController**
**Prüfungsamt:** Ansicht aller unveröffentlichten Prüfungen mit der Möglichkeit Prüfungen zu verifizieren / veröffentlichen bzw den Termin abzulehnen und dabei mit einer Begründung zu versehen


ManageUnverifiedPruefungenController
doDispose()
event()
DefaultController
AllPruefungenController
event()
doDispose()
ManagePruefungenController
event()
doDispose()
MyPruefungenController
doDispose()
event()
PruefmodulMainController
pmMenuTree
pmMenuAndToolController
currentTestCtr
addControllerListener()
dispatchEvent()
event()
buildMenuTree()
createController()
doDispose()
EditPruefungController
doDispose()
event()