Entwurfsbeschreibung des Fremdprojektes „eLate“

Gliederung:



## 1) Allgemeines

Das eLate-Portal ist ein webbasierendes Informationssystem, welches hauptsächlich an einem Lehrstuhl einer Universität eingesetzt wird. Es stellt eine Vielzahl an Möglichkeiten zur Verwaltung von Veranstaltungen bereit. Dazu zählt die Übersicht von und Einschreibung in Veranstaltungen, das Bearbeiten von Übungsaufgaben (Online-Bearbeitung und Upload von Aufgabenlösungen) usw..

Das eLate-Portal, ist ähnlich wie bei Jetspeed durch einzelne Portlets erstellt worden, die an einzelne Veranstaltungen modifiziert werden kann, was für eine gute Anpassbarkeit darstellt. Dadurch entsteht, da einzelne Informationen, die nicht benötigt werden, auch nicht eingebaut werden müssen, eine gute Übersichtlichkeit.

Um das eLate-Portal zu nutzen, benötigt man ausschließlich einen Internet-Browser. Mit diesem wird über das Internet eine Verbindung zu dem Server hergestellt, auf dem das Portal installiert ist. Bei dem eLate-Portal handelt es sich um eine Instanz des Jetspeed-2. Jetspeed stellt schon einige Funktionen zu Verfügung, die für das eLate-Portal wichtig sind. Allerdings erweitert das Portal Jetspeed um einige Funktionen. Es handelt sich somit um eine Erweiterung von Jetspeed.
Die Speicherung aller wichtigen Daten wie Benutzername, Matrikelnummer und Passwort erfolgt in XML-Dateien. Auch Rollendaten, die bei der Registrierung des Nutzers vergeben werden, werden von eLate verwaltet und gespeichert. Sie dienen der Rechtevergabe in dem Portal.

## 2) Produktübersicht

Die vom eLate-Portal zur Verfügung gestellten Rollen sind für die Veranstaltungsverwaltung an Universitäten von großer Bedeutung. Unterschieden werden hier die Rollen Administrator, eingeschriebener User und Gast. Jede dieser Rollen hat spezielle zugeschnittene Rechte, welche den Bedienumfang der einzelnen User festlegen.

Im eLate-Portal lassen sich die folgenden Funktionen unterteilen, wenn man noch kein registrierter bzw. eingeloggter User ist:
- registrieren in das Portal
- als registrierter User kann man sich einloggen.
- Falls man bereits registrierter User sein Passwort vergessen hat, so kann man es sich an die bei

der Registrierung angegebenen eMail-Adresse zusenden lassen.

Für bereits registrierte User stehen die folgen Möglichkeiten zur Verfügung:
a) Administrator:
- Bearbeitung von Rechten
- Anlegen von Texten und Veranstaltungen
- Beiträge im Forum erstellen

Aufgrund der unzureichenden Kommentierung des eLate-Quelltextes war es nicht weiter möglich rauszufinden, was dem Administrator noch zur Verfügung steht.

b)eingeschriebene User:

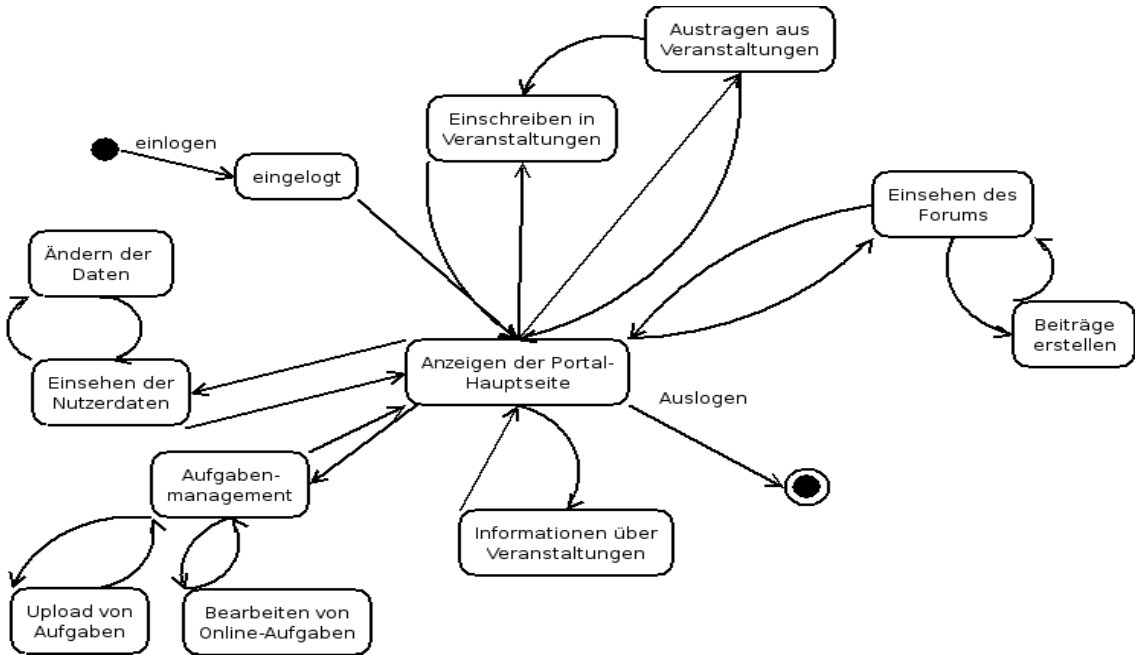
- Unterteilung Dozent:

- Erstellung von Veranstaltungen (Übungen, Vorlesungen, Prüfungen)
- Bereitstellung von Informationen
- Bereitstellung von Materialien, die runtergeladen werden können
- Erstellung von Übungen
- Beiträge im Forum erstellen
- Download der abgegeben Lösungen

- Unterteilung Student:

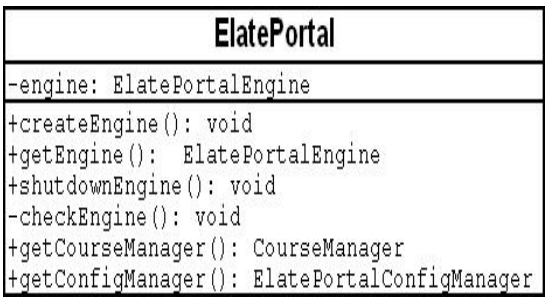
- Einschreibung in Veranstaltungen (Übungen, Vorlesungen, Prüfungen)
- austragen bzw. wechseln der Veranstaltung
- Bearbeitung von Aufgaben
- Upload von Lösungen ?
- Beiträge im Forum erstellen
- Einsehen von Informationen
- Download der bereitgestellten Materialien
- Ändern der User-Daten

Nun wurde das mögliche Vorgehen eines registrierten Users in Form eines Zustandsautomaten dargestellt.




## 3. Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtkonzept

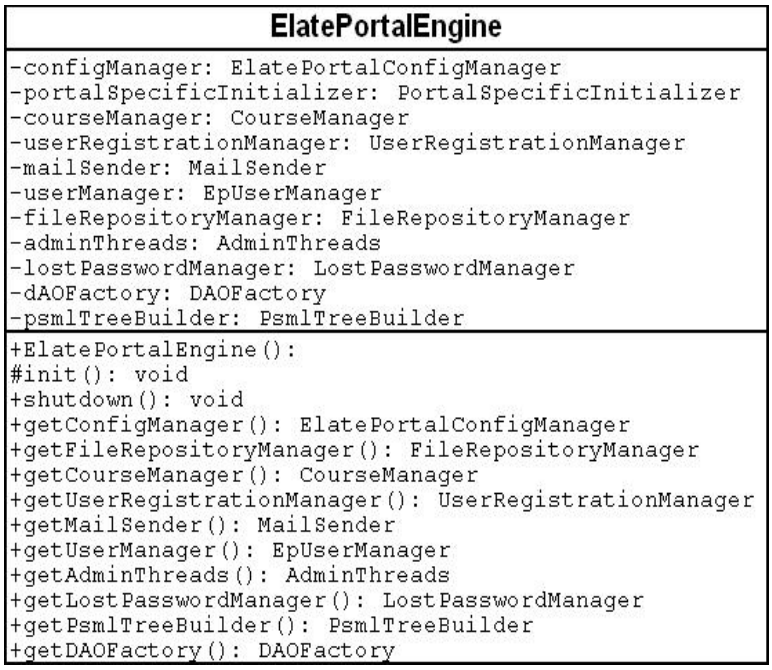
Das Elate-Portal ist ein Portal, welches von seiner Portaltechnik hochgradig unabhängig ist, d. h. es ist möglich dieses Portal in verschiedene Portaltechniken einzubinden. Im aktuellen Fall ist dies Jetspeed 2.0. Um dies zu bewerkstelligen, ist eine sehr hohe Abstraktion vom Elate nötig. Umgesetzt wurde dies in der Startklasse ElatePortal. Diese Klasse enthält eigentlich nur eine statische Variable, die die Engine (ElatePortalEngine) repräsentiert. Außerdem hat sie einige Funktionen um diese Engine zu starten (createEngine) und zu beenden (shutdownEngine). Ansonsten wurde noch Zugriff auf die wichtigsten Elemente der Engine mittels Funktionen (getCourseManager und getConfigManager) gegeben.




Die Engine selbst enthält die Hauptobjekte CourseManager und ConfigManager(ElatePortalConfigManager). Desweiteren enthält es noch die Objekte:

portalSpecificInitializer (welches nur intern genutzt wird), userRegistrationManager, mailSender, userManager, fileRepositoryManager, adminThreads, lostPasswordManager, dAOFactory , psmlTreeBuilder

Die meisten von diesen Objekten sind reine Serviceklassen und nutzen nur andere Elate API's aus der Engine. Der Rest der Objekte ist als Interface deklariert und besitzt eine Jetspeed Implementierungsvariante. Die Interfaces ermöglichen es, dass die API vom Elate-Portal unabhängig von der API der Portaltechnik (Jetspeed 2.0) ist.



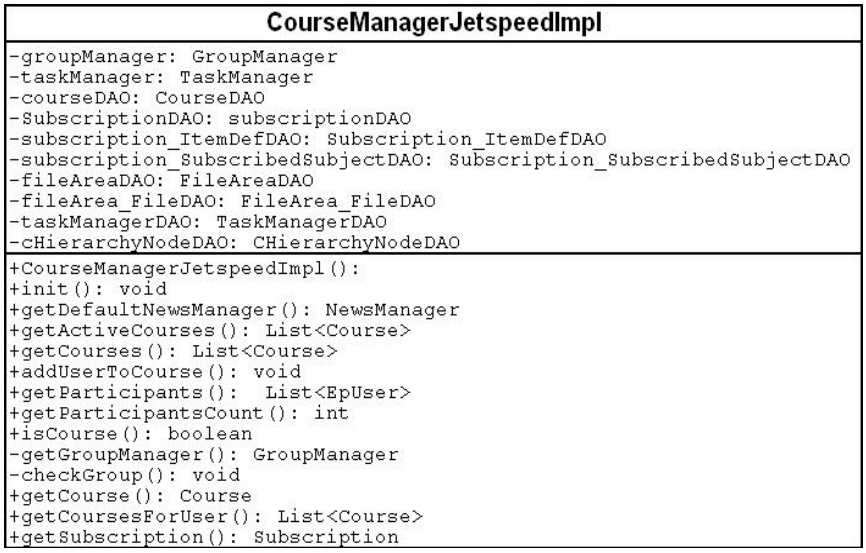

Bemerkung: Die ElatePortalEngine enthält eine innere Klasse mit den Namen Init. Sie dient nur der Initialisierung und ist deshalb nicht weiter von Bedeutung für uns.

Die wichtigsten Teile der Engine sind: CourseManager, ConfigManager, DAOFactory, PSMLTreeBuilder, FileRepositoryManager.

- Der **CourseManager** enthält sämtliche Lehrveranstaltungen, zugehörige Übungsgruppen und in diese eingetragene Nutzer.
- Der **ConfigManager** wird für die Konfigurierung des Elate-Portals genutzt.
- Die **DAOFactory**, welches SpringFrameworks (springframework.beans) benutzt, stellt wichtige Dienste für das Elate-Portal bereit und bildet somit den Kern der Engine. Auf sie wird später noch einmal eingegangen. (Zu integrierende Portlets (unser Wiki) nutzen dieses Objekt aber nicht.)
- Der **PSMLTreeBuilder** ist für die Verarbeitung der PSML-Dateien zuständig. Das Objekt selbst wird von einem zu integrierenden Portal (unser Wiki) nur indirekt benutzt. Die Portale müssen durch die PSML-Datei in das Portal integriert werden. Diese Aufgabe übernimmt dieses Objekt.
- Und zu guter Letzt der **FileRepositoryManager**. Er dient der Dateiverwaltung des Elate-Portals und wird als solcher ebenfalls nicht von einem Portlet, was neu integriert werden muss, benutzt.

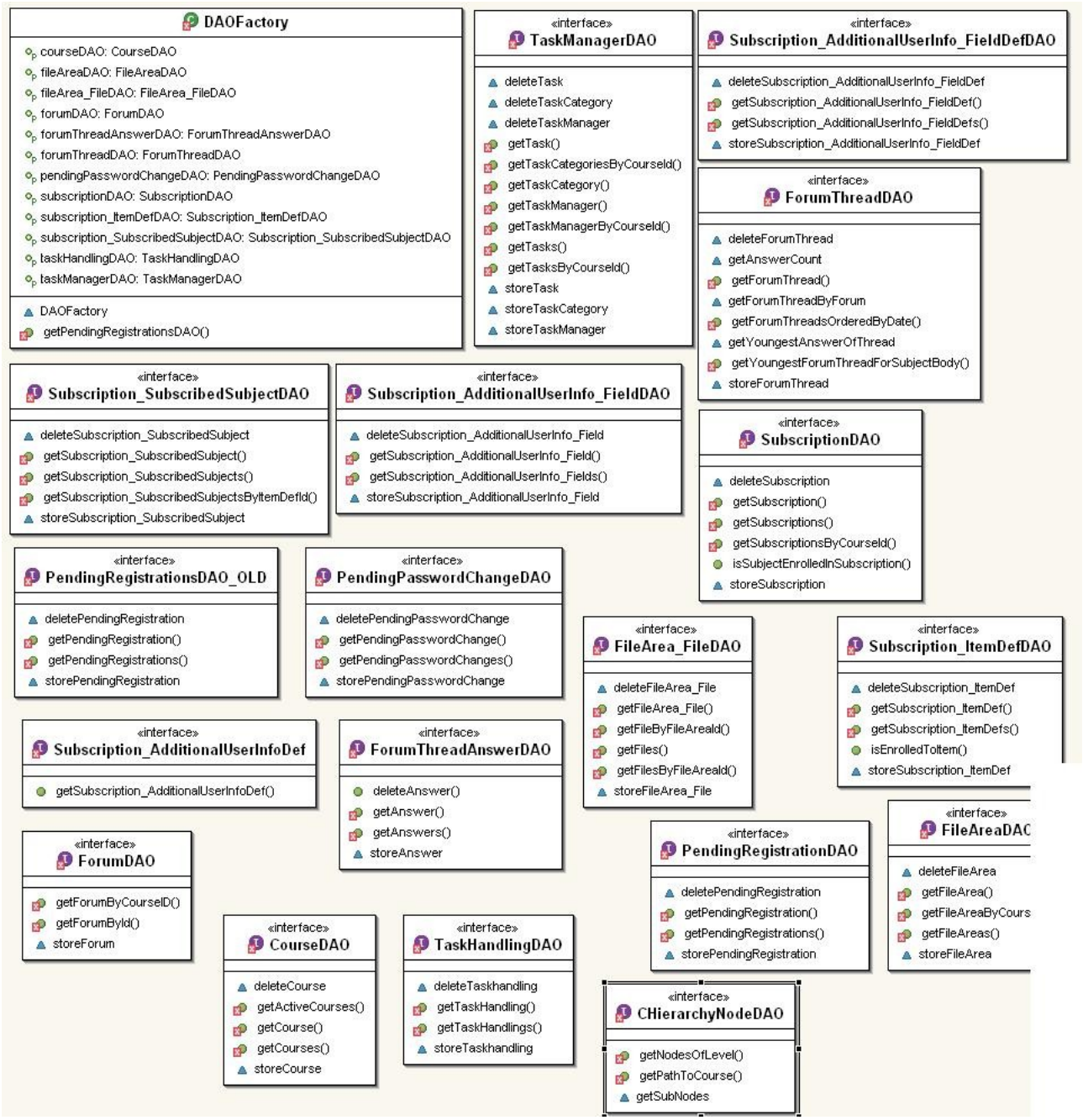
Bleibt also nur noch der CourseManager, der für uns als einziges wirklich von Bedeutung ist. Die Jetspeed Implementierung des CourseManagers (CourseManagerJetspeedImpl) enthält wieder eine Reihe von Objekten.
Diese wären: taskManager, courseDAO, subscriptionDAO, subscription_ItemDefDAO, subscription_SubscribedSubjectDAO, groupManager, fileAreaDAO, fileArea_FileDAO, taskManagerDAO, cHierarchyNodeDAO.

Alle Objekte mit DAO nutzen die DAOFactory und werden deshalb später bei der DAOFactory behandelt. Der GroupManager ist ein Objekt vom Jetspeed 2.0. Diesem wird noch einige Aufmerksamkeit für unser Wiki zuteil werden. Der TaskManager enthält sämtliche Daten . Diese Daten würden wir für unser Wiki ebenfalls brauchen. Darum werden wir ein entsprechendes Interface benötigen.




Ansonsten enthält der CourseManager noch wichtige Funktionen, wie getDefaultNewsManager (für Neuigkeiten – wird von uns benötigt), getActiveCourses (für alle laufenden Lehrveranstaltungen), getCourses (sämtliche Lehrveranstaltungen – benötigen wir auch), addUserToCourse (einen Nutzer in einer Lehrveranstaltung eintragen) und getCoursesForUser (alle Lehrveranstaltungen, die ein Nutzer besucht).

Zur DAOFactory:
Die DAOFactory ist so etwas wie eine Service-Fabrik. Sie enthält sämtliche Komponenten, die für die Arbeit mit CVS notwendig sind. Die meisten dieser Komponenten besitzen eine eigene JSP-Datei. Diese enthält ein grobes Grundmuster der zu erzeugenden dynamischen HTML-Seiten. In diesen JSP-Dateien sind aber einige Stellen noch nicht ausgefüllt. An diesen Stellen befinden sich Schlüsselwörter, welche durch die DAOFactory entsprechend der Situation ersetzt werden. So gesehen ist die DAOFactory die Verwaltungsschnittstelle für sämtliche Servicedienste des Elate-Portals. Desweiteren ist die DAOFactory für alle Datenbankzugriffe zuständig. Daher auch das „DAO“ (DataAccessObject).




Was bei der Studie des Elate-Portals und besonders der DAOFactory auffällt, ist, dass zwei Techniken namens Struts und SpringFramework zum Einsatz kommen, zum Bsp. bei StrutsPortletMessaging oder bei der DAOFactory.

Diese Techniken dienen dazu, dass der View-Teil des MVC Modells in so genannte JSP-Dateien oder ausgelagert werden können. Denn Struts und SpringFramework bieten die Möglichkeit Schlüssel in diesen Dateien zu ersetzen und dadurch die dynamischen HTML-Seiten zu erzeugen. Ein weiterer Vorteil ist die Internationalisierung. Einfach mehrere Ressourcen-Dateien oder verschiedene Datenbanktabellen für diese Techniken anlegen (für die verschiedenen Sprachen) und beim Erstellen der dynamischen HTML-Seite die entsprechende Datei oder Tabelle verwenden.
Eine gute Erklärung für Struts zum Beispiel finden sie unter http://www.lalilуna.de/what-is-struts-de.html

DAOFactory, SringFramework und Struts sind eigentlich untrennbare Teile.

## 4. Struktur- und Entwurfsprinzipien der einzelnen Pakete:

### a) ElatePA (elate Porlet Applications):

**Packages:** de.elatePortal, de.elatePortal.engine, de.elatePortal.engine.impl
**Funktion:** Die Klassen dieser Pakete übernehmen die Initialisierung und Ausführung der Anwendungslogik sowie das Veranstaltungsmanagement.
**wichtige Klassen:** ElatePortal: Startet die Engine und erlaubt Zugriff auf wichtige Klassen wie z.B. CourseManager und ConfigManager.
ElatePortalEngine: Initialisiert alle Komponenten und führt elate aus.
ElatePortalServlet: Servlet des ElatePortals, startet und beendet elate.
AdminThreads: Klasse zur Verwaltung von Threads (z.B. Thread für Garbage Collection).
PsmlTreeBuilder: Stellt die für einen User anzeigbaren Seiten zusammen. (Je nach Rolle und Gruppenzugehörigkeit des Users hat dieser Zugriff auf unterschiedliche Seiten.)

**Package:** de.elatePortal.util
**Funktion:** Stellt diverse Klassen mit Hilfsmethoden zur Verfügung, etwa zum Senden von Mails.

**Package:** de.elatePortal.db.dao, de.elatePortal.db.obj
**Funktion:** Interfaces und Klassen der DAOFactory Services. DAO (Data Access Object) ist ein Designpattern, welches eine Art Brücke (das DAO-Objekt) zwischen Datenobjekten und Datenquellen bildet, die den Zugriff auf die Daten vereinfacht. Im Falle von Elate dient es dem Speichern der Objekt in Tabellen der MySQL-Datenbank.

**Package:** de.elatePortal.db.filesystem
**Funktion:** Beinhaltet Klassen für den Dateizugriff.

**Packages:** de.elatePortal.db.jaxb, de.elatePortal.db.jaxb.impl, de.elatePortal.db.jaxb.impl.runtime
**Funktion:** Klassen, die die Speicherung in XML-Dateien erlauben. Benutzen JAXB (JavaTM Architecture for XML Binding), welche das Mappen zwischen Javaobjekten und XML- Dateien erlauben.

**Package:** de.elatePortal.db.vo
**Funktion:** Implementiert verschiedene Klassen, die nur dem Speichern von Daten dienen.

**Packages:** de.elatePortal.components.courses.subscriptions, de.elatePortal.components.download, de.elatePortal.components.lostpassword, de.elatePortal.components.news.impl, de.elatePortal.components.security, de.elatePortal.components.taskimpl, de.elatePortal.components.userregistration, de.elatePortal.components.courses
**Funktion:** Implementierung von Kernfunktionen des Elate-Portals. Dazu gehört etwa das Veranstaltungsmanagement, die Einschreibung in Veranstaltungen/Kurse, der Download von Dateien und das Usermanagement.
**Wichtige Klassen:** DownloadServlet: Ein Serlvet, das Anfragen für Downloads bearbeitet.
EpUserManagerImpl: Eine UserManager-Implementierung für Jetspeed.
TaskFactoryImpl: Erzeugt und verwaltet sog. Tasklets.*

*(* Eine Taskengine erlaubt es Dozenten, Aufgabenserien und Prüfungen online zu erzeugen, welche dann ebenfalls online gelöst werden können. (Pakete de.thorstenberger.taskmodel.* ) Wird im Rahmen dieser Analyse nicht genauer betrachtet, da wir uns nur auf api und elatePA konzentrieren.)*

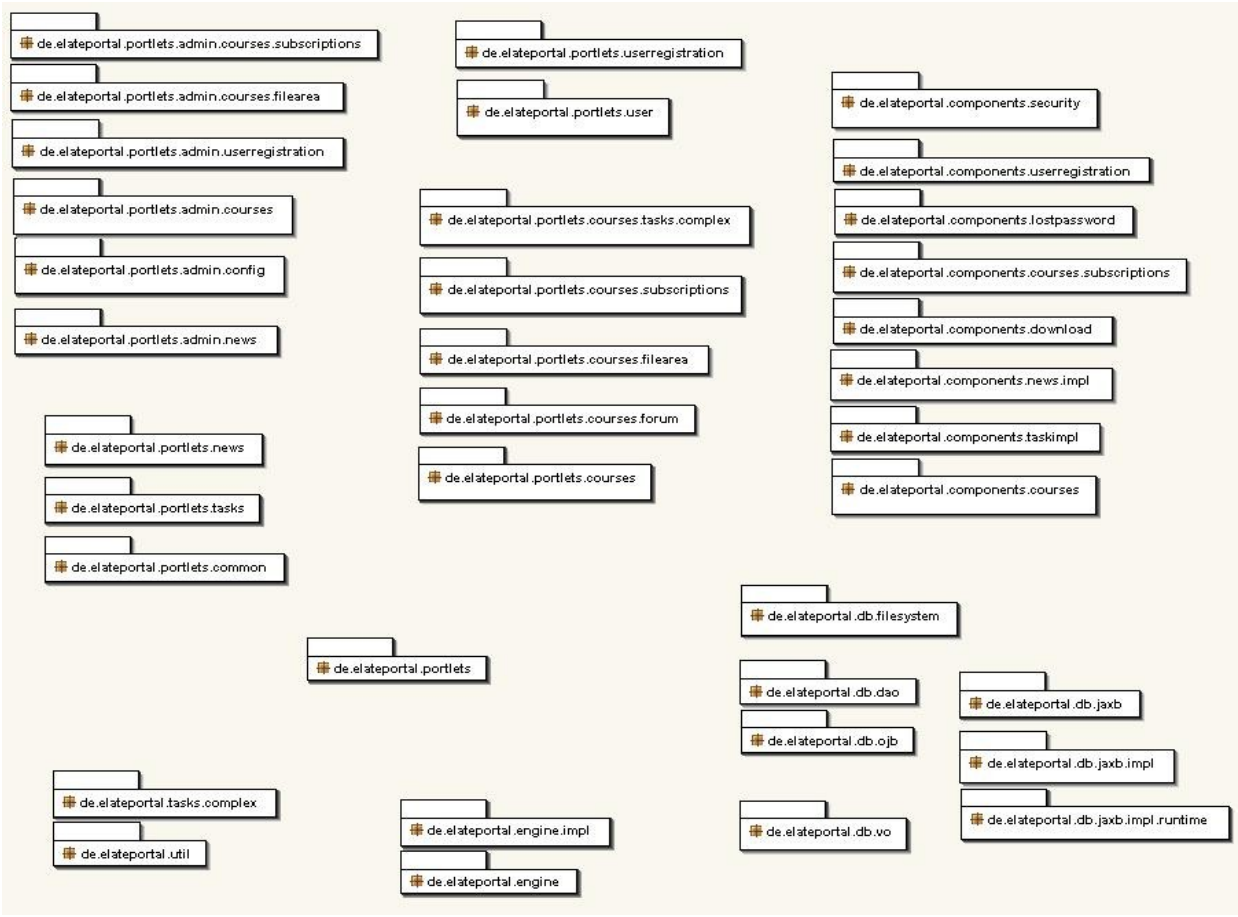
CourseManagerImpl: Eine Jetspeed-spezifische Implementierung einer Klasse zur Verwaltung von Veranstaltungen/Kursen. Erlaubt etwa das Hinzufügen von Usern zu Veranstaltungen.

**Packages:** de.elatePortal.portlets.common, de.elatePortal.portlets
**Funktion:** Stellen portletspezifische Funktionen und Standartportlets bereit. Die Portlets benutzen .vm oder .jsp zur Darstellung des Inhalts. Es wird das Struts-Framework verwendet.

**Packages:** de.elatePortal.portlets.courses, de.elatePortal.portlets.courses.filearea, de.elatePortal.portlets.courses.forum, de.elatePortal.portlets.courses.subscription, de.elatePortal.portlets.courses.tasks, de.elatePortal.portlets.courses.tasks.complex
**Funktion:** Stellt alle Portlets und Funktionen zur Verfügung, die zu den Veranstaltungen/Kursen gehören (Einschreibung, ein veranstaltungsspezifisches Forum etc.).

**Packages:** de.elatePortal.portlets.admin.config, de.elatePortal.portlets.admin.courses, de.elatePortal.portlets.admin.courses.filearea, de.elatePortal.portlets.admin.courses.subscriptions, de.elatePortal.portlets.admin.news, de.elatePortal.portlets.admin.userregistration
**Funktion:** Implementiert Adminportlets des Elate-Portals.

**Packages:** de.elatePortal.portlets.user, de.elatePortal.portlets.userregistration, de.elatePortal.portlets.news
**Funktion:** Implementiert Benutzerregistrierung und News.

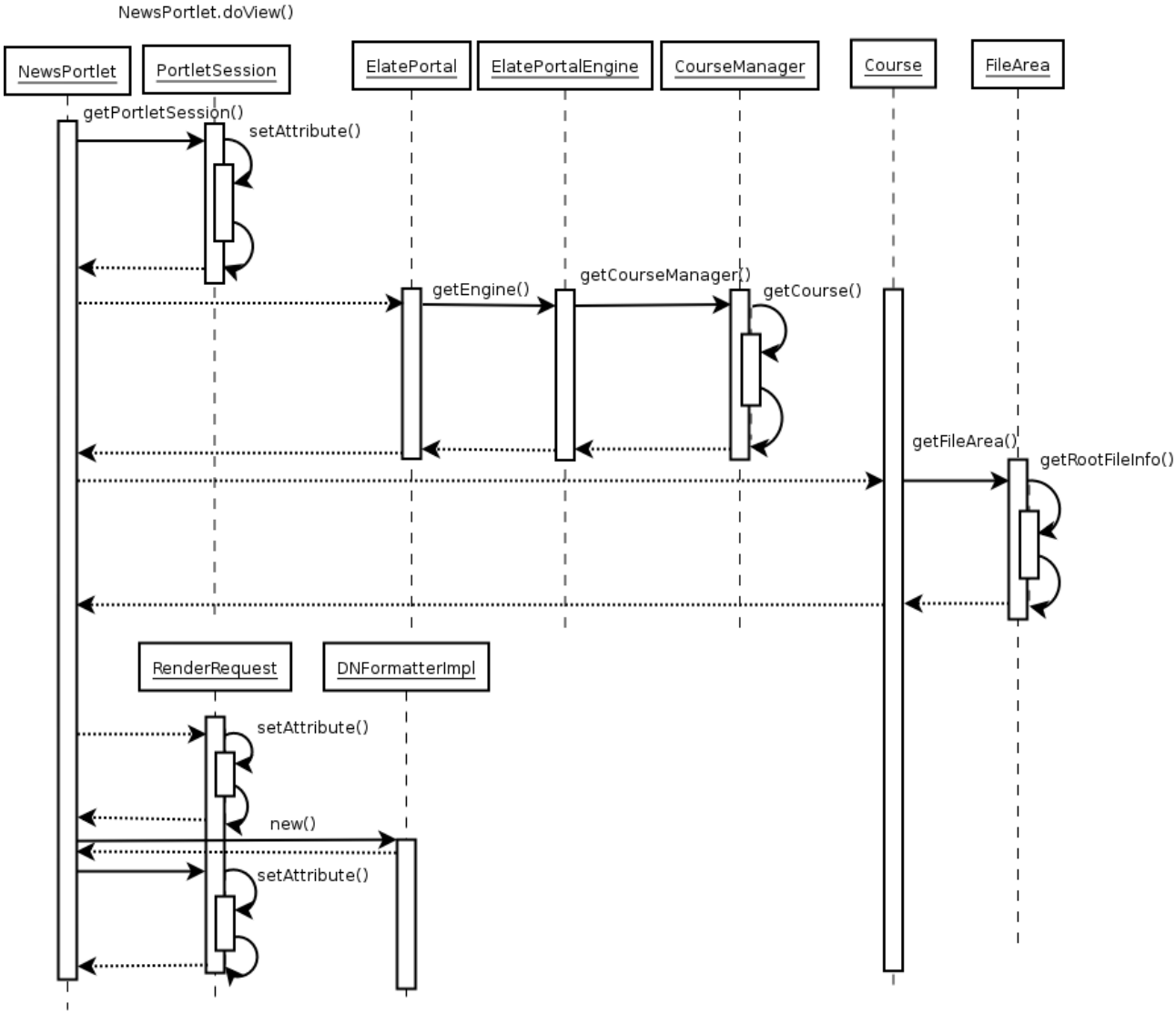



**b) api:**

Die Paketstruktur der api ist ähnlich zu der von elatePA. Es werden zahlreiche Interfaces deklariert, deren Implementierung dann in elatePA speziell für das Jetspeed-2-Portal erfolgt. Die Interfaces können auch für andere Portale entsprechend implementiert werden, sodass Elate dann auf diesen genauso funktioniert, wie auf Jetspeed-2.

Es werden ebenfalls einige Hilfsklassen und Exceptions deklariert.

Ein Beispiel für ein Portlet:


NewsPortlet.doView()
NewsPortlet
PortletSession
ElatePortal
ElatePortalEngine
CourseManager
Course
FileArea
getPortletSession()
setAttribute()
getEngine()
getCourseManager()
getCourse()
getFileArea()
getRootFileInfo()
RenderRequest
DNFormatterImpl
setAttribute()
new()
setAttribute()