Entwurfsbeschreibung

Inhaltsverzeichnis

1. Allgemeines

1.1 Charakterisierung

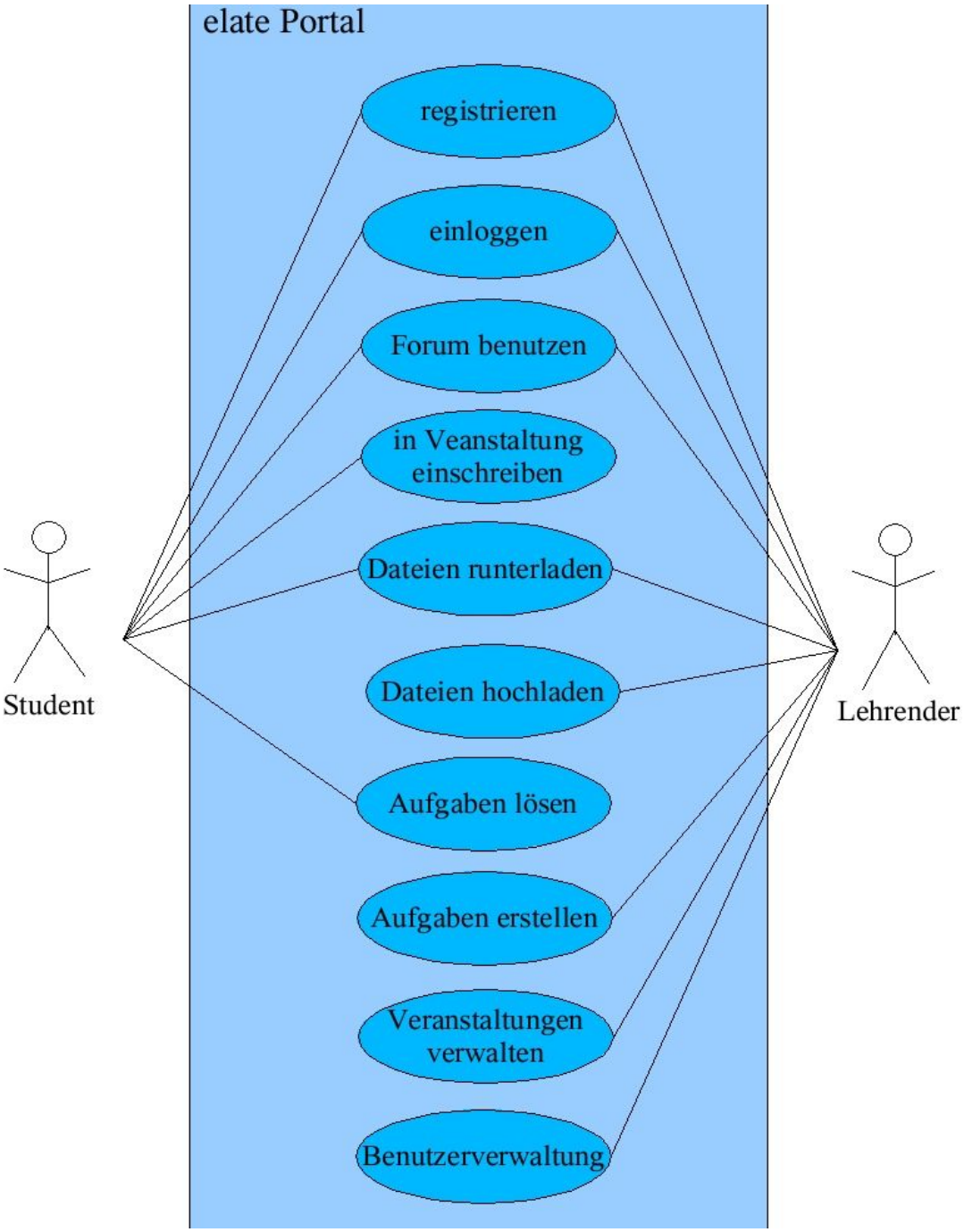
Das elatePortal ist eine Neuentwicklung des UebManagers und dient zur Verwaltung von Lehr-Veranstaltungen. Es soll Studenten und Lehrenden eine einfach zu bedienende Web-Oberfläche bieten, die u.a. „eTesting“ unterstützt.
Mit dem elatePortal kann eine große Anzahl von Veranstaltungen verwaltet werden. Das bedeutet Studenten können sich online einschreiben und Übungsaufgaben oder Skripte runterladen, falls solche von Lehrenden online zur Verfügung gestellt wurden. Dozenten und Professoren können zudem Online-Übungen erstellen, die dann von Studenten auch online gelöst und automatisch bewertet werden. Ein einfaches Forum bietet dabei eine Kommunikationsplatform für Studenten und Lehrende. Das elatePotal ist OpenSource-Software unter GNU-Lizenz und lässt sich skalieren und erweitern.

1.2 Systemvoraussetzungen

Um eine Instanz des elatePortals zu installieren braucht man einen Apache Webserver auf dem Tomcat, Jetspeed-2 und eine Java VirtualMachine (JVM) läuft.
Die Benutzer des elatePortals brauchen nur einen Webbrowser.

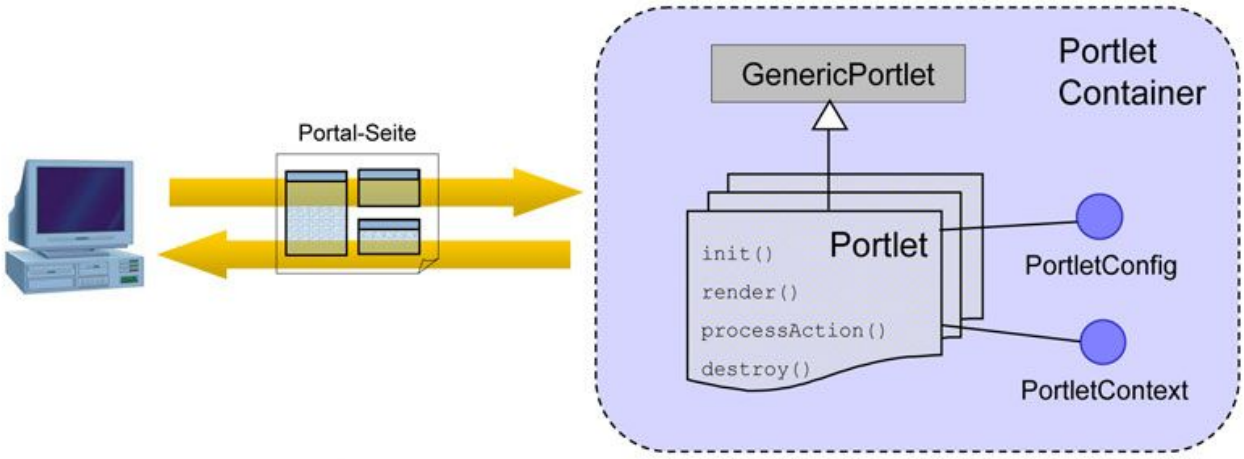
2. Produktübersicht

Die wesentlichen Produktfunktionen in der Übersicht:



3. Grundsätzliche Struktur- und Entwurfsprinzipien des Gesamtsystems

3.1 Aufbau von Portalkomponenten



Request-Typen ähnlich GUI-Komponenten:

- RenderRequest
- ActionRequest

Greift ein Benutzer auf die Portalseite zu, so werden nacheinander mehrere Module angesprochen. Die alles umschließende Applikation, hier nicht dargestellt, ist ein Servlet Container, Apache Tomcat in unserem Fall. Dieser sorgt dafür das äußere Anfragen des Benutzers an Servlets, weitergeleitet werden, die dann darauf mit Zustandsänderungen und Ausgabe reagieren können. Ein solches Servlet ist der Portlet Container. In einer typischen Portlet-Portalseite ist er für den Benutzer nicht nur für die Portlets zuständig, sondern auch für etwaige Navigationselemente des Portals. Intern liefert der Portlet Container nicht nur eine Umgebung zur Ausführung der Portlets, sondern übernimmt u.U. auch Verwaltungsaufgaben wie das Benutzermanagement. Ein Portlet selber ist ein Java Modul, welches sich für den Benutzer als ein Fenster, innerhalb der Portalseite zeigt. Dies ermöglicht für den Benutzer mehrere Portlets parallel auf einer Portalseite abzulegen. Intern ist ein Portlet eine spezielle Java Klasse, meistens von javax.portlet.GenericPortlet abgeleitet, welche typische Portlet Methoden unterstützen muss. Dazu gehören Initialisierung (findet statt sobald ein Portlet vom Container geladen wird), Zerstörung (um Resourcen freizugeben, wenn das Portlet entladen wird), das "rendern" des Portlets (also das generieren von HTML-Code für die Anzeige beim Benutzer) und die Verarbeitung von Aktionen des Benutzers (z.B. das Abschicken eines Formulars). Die letzten zwei Methoden haben ein gewisse Analogie zu GUIs, da dort auch in Anzeige und der Verarbeitung von Ereignissen unterschieden wird. Während seiner Ausführung wird dem Portlet eine gewisse Laufzeitumgebung durch den Portlet Container zur Verfügung gestellt. Diese beinhaltet z.B. die Konfiguration des Portlets zum Zeitpunkt der Initialisierung, also Initialisierungsparameter und den so genannten Kontext des Portlets. Dieser dient als Schnittstelle zum Portlet Container und zum Portal.

3.2 Frameworks

Ein Framework ist eine Art Programmgerüst, dass die Anwendungsarchitektur vorgibt.
Der Programmierer registriert dabei seine konkrete Implementierung, die dann durch das Framework gesteuert und benutzt werden, statt lediglich Klassen und Funktionen zu benutzen, wie es z.B. Bei Klassenbibliotheken der Fall ist. Das Registrieren der konkreten Klassen kann dabei fest verankert im Programmcode geschehen, oder „von außen" (z.B. durch Konfigurationsdateien) festgelegt werden. Letzteres ist beim elatePortal der Fall.
Ein Framework definiert insbesondere den Kontrollfluss der Anwendung und die Schnittstellen für die konkreten Klassen, die vom Programmierer erstellt und registriert werden müssen. Frameworks werden also im Allgemeinen mit dem Ziel einer Wiederverwendung „architektonischer Muster" entwickelt und genutzt.

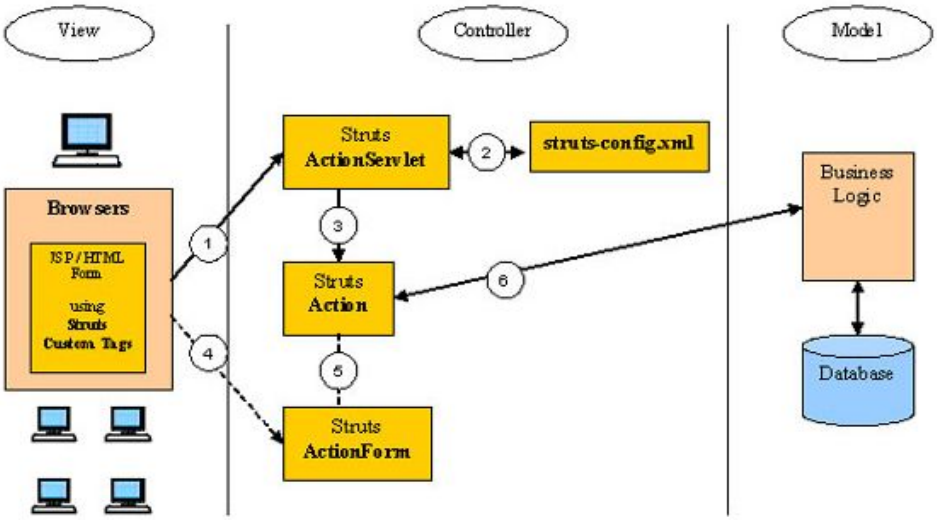
3.3 Stuts-Framework – Model-2-Architektur

Es handelt sich hier um ein Konzept, das die einzelnen Komponenten Datenhaltung, Geschäftslogik und View klar voneinander abgrenzt und dabei Java-Servlets sowie JSP gleichermaßen einbindet. Dieses Konzept ist allgemein bekannt als Model-2-Architektur für Webanwendungen, die ihren Einsatz im vom elatePortal verwendeten Struts-Framework gefunden hat. Eine wichtige Rolle spielt hierbei das ActionFirst Pattern. Das bedeutet, dass View-Komponenten, die durch JSP abgebildet werden, ausschließlich über Struts-Actions (Servlets) referenziert werden, niemals über den Aufruf der Seite, einen HTTP-Request. Dabei wird jeder HTTP-Request von einer Controller-Komponente (einem Servlet) entgegengenommen, bei Struts dem so genannten ActionServlet. Nach einem vom Benutzer initiierten HTTP-Request, z.B. durch Absenden eines Formulares, wird eine Struts-Action ausgeführt, die eine Verbindung zur Geschäftslogik herstellt und diese aktualisiert. Die Ergebnisse der Ausführung werden anschließend in dem HTTP-Request als Attribute abgelegt. Danach wird über einen Action-Forward die eigentliche Darstellung der Seite (View) aufgerufen und mit den im HTTP-Request abgelegten Daten gefüllt. Befindet sich in einer View-Komponente ein HTML-Formular, so wird dieses über ein Java-Bean (Klasse die nur get- und set-Methoden besitzt) mit Werten aus dem Model gefüllt.
Auch die Klassen des elatePortals folgen der Model-2-Architektur als Regelwerk. So wird u.a. die Kommunikation von Views untereinander, wie auch die feste Kodierung von Flusssteuerung in JSP, vermieden.
Ziel dieser Architektur ist die Trennung von Präsentation (View), Datenhaltung (Controller) und Anwendungslogik (Model). Dies erhöht die Übersicht und die Wartbarkeit.
Man kann in Struts vier Hauptkomponenten definieren:

- **JSP**: Präsentation. Wird zur Laufzeit in ein Servlet (Java-Code) umgewandelt und kompiliert.
- **Action**: Controller. Schnittstelle zwischen Darstellung (View) und Anwendungslogik (Holen und Aktualisieren der Daten).
- **FormBean**: Datenhaltung und Validierung (Inhalte von HTML-Formularfeldern in Java-Beans)
- **Anwendungslogik**: Model. Eigentliche Programmlogik.

Alle 3 Komponenten werden in der zentralen Konfigurationsdatei von Struts (`struts-config.xml`) „von außen" miteinander verknüpft und können miteinander kommunizieren.

Ein typischer Vorgang mit Model-2-Architektur:



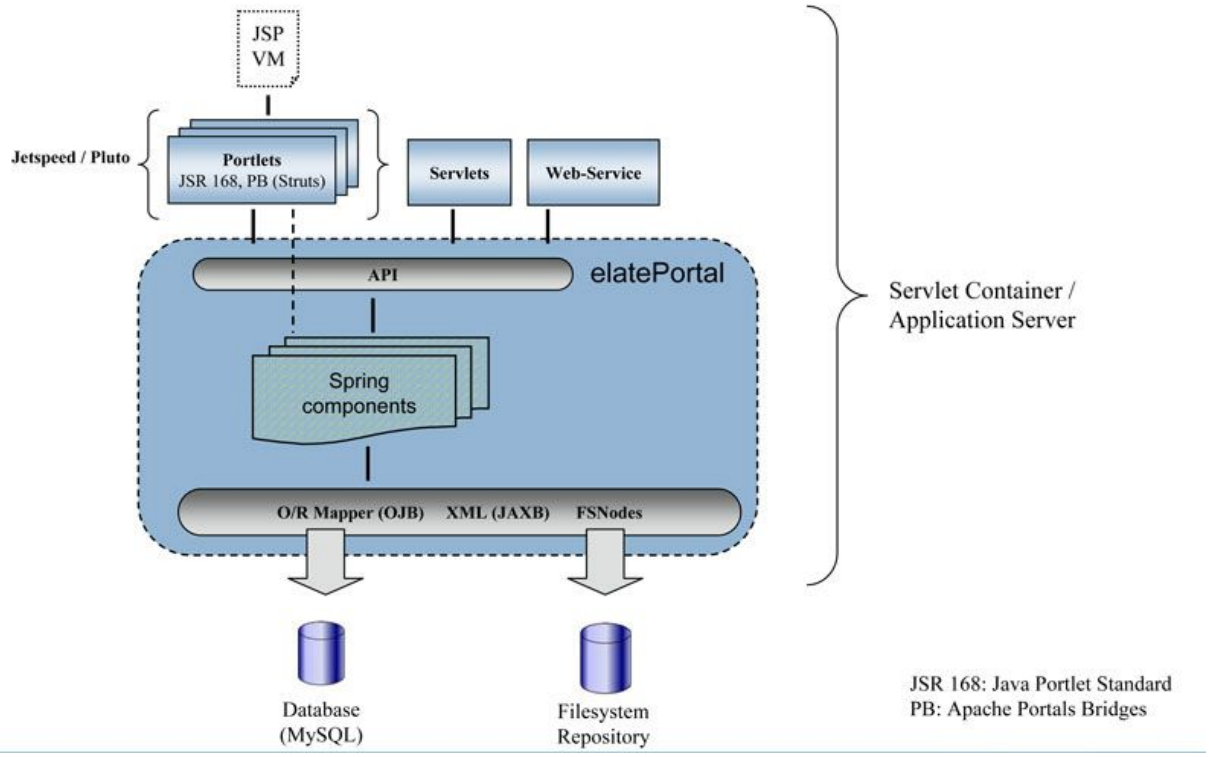
1. Anfrage vom Webbrowser, z.B. über JSP, wird an ActionServlet übermittelt.
2. Das ActionServlet ermittelt anhand der struts-config.xml welche Action-Unterklasse benutzt wird.
3. Das ActionServlet übergibt Anfrage an entspechende Action-Unterklasse.
4. Die übermittelten Daten aus dem HTML-Formular werden automatisch in der ActionForm-Unterklasse gespeichert.
5. Die Action-Klasse kann auf die Daten der ActionForm-Klasse (Java-Bean) zugreifen, die dann auch an die Anwendungslogik übergeben wird (6).
6. Die Action-Klasse benutzt die eigentliche Anwendungslogik.

3.4 Spring-Framework

Das Framework Spring basiert im Wesentlichen auf folgenden Prinzipien:
- **Umkehrung der Kontrolle**: Den Objekten werden die abhängigen Objekte bzw. Ressourcen zugewiesen. Sie müssen sie nicht selbst suchen.
- **Aspekt orientierte Programmierung**: Dadurch kann man vor allem technische Aspekte wie Transaktionen oder Sicherheit isolieren und den eigentlichen Code davon frei halten.
- **Templates**: dienen dazu, die Arbeit mit einigen APIs zu vereinfachen indem Ressourcen automatisch aufgeräumt und Fehlersituationen einheitlich behandelt werden.

Damit wird ein Programmiermodell möglich, dass auf Klassen ohne externe Abhänigkeiten wie z.B. Namenskonvetionen, oder zwingend einzuhaltende Schnittstellen basiert (so genannte POJOs), die dadruch in verschiedenen Umgebungen (auf einem Server oder in einer Client Anwendung) lauffähig sind. So geschriebene Anwendungsteile werden auch als „Spring components“ bezeichnet.

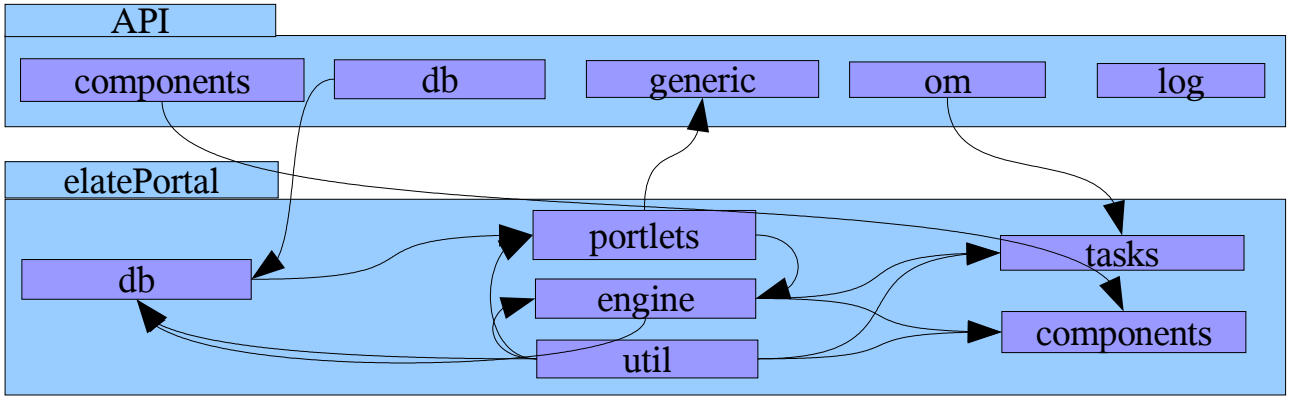
3.5 Gesamtstruktur des Systems



Das ElatePortal wurde als Portlet Application entwickelt und dabei in 4 Schichten unterteilet, die hier von oben nach unten abgebildet sind.
Schicht 1 ist die Core-Ebene die die Basis für alles weitere bildet. Das sind u.a die JavaVirtual Machine und Jetspeed bzw. alternativ Pluto. Der View wird dabei von kompakten templatebasierenden VelocityPortlets übernommen. Lediglich wo es die Komplexität des Problems nicht anders zu zulässt werden JSPs verwendet. Der 1. Schicht untergeordnet ist die Anwendungsebene in der sich Controller befinden, die also die Behandlung und Zuordnung der Rohdaten übernimmt. Dazu gehören die bereits erwähnten Struts-Actions (Model-2-Architektur) die mit so genannte PortletBridges eingebunden sind und natürlich auch die Portlets selbst, aber auch die Engine, die alle inneren Interaktionen verwaltet. Die Api wiederrum wird unter Shared-Libs deployed und ermöglicht so auch externen Anwendungen auf Applikationsfunktionen zuzugreifen. Die Zentrale Schnittstelle hierfür ist die „ElatePortalServices.java“. Darauf basierend findet sich mit der Bussineslogik Sicht 3. Sie besteht aus Spring components, das erleichtert weiterhin die Skalier- und Erweiterbarkeit des elatePortals. Sie beinhalten die eigentliche Umsetzung der einzelnen Funktionenen der Applikation. Schicht 4 schließlich ist für die dauerhafte Datenhaltung zuständig. Dafür steht neben einem Object Relational Mapper, der die Abbildung von Objekten auf eine Datenbank ermöglicht, auch ein XML-Binder (JAXB) zur Verfügung, mit dem Objekte zudem über XML-Dateien im Dateisystem abgelegt werden können.

4. Grundsätzliche Struktur- und Entwurfsprinzipien der einzelnen Pakete

4.1 Paketübersicht:



API
In der API befinden sich alle Interfaces. Diese sind in folgenden Paketen:
- components: news, lostpassword, userregistration, courses
- db: filesystem
- generic: exceptions
- om: filearea, security, subscriptions (news, couse, data, forum, tasks)
- log: logger

elatePortal
Das elatePortal beinhaltet Pakete, die die Interfaces der API implementieren. Diese sind:
- db: Es wird eine MySQL-Datenbank in Kombination mit dem Filesystem (Dateisystem) verwendet. Beispielsweise werden die zum Download bereitgestellten Dateien in einem Ordner gesammelt. Das Paket db sorgt dafür, dass die Baumstruktur dieses Ordners in ein Objekt übertragen wird.
- portlets: Die Portlets übernehmen die Schnittstelle zum Webbrowser. Sie stellen die grafische Oberfläche des elatePortals dar. Über sie werden auch dessen Funktionen ausgeführt.
- util: Dieses Paket stellt kleine Funktionen bereit, die von vielen anderen Paketen benötigt werden.
- tasks: Dieses Paket dient zur Impementierung des Task-Interfaces. (Online-Aufgaben, eTesting) Aufgaben und die Lösungen der Studenten werden als XML-Datein abgelegt.
- components: Beinhaltet die meisten Teilkomponenten aus denen sich das die Funktionen (Anwendungslogik) des elatePortals zusammensetzten. Einige interessante Pakete sind z.B.:
  - security.EpUserImpl:
    Ermöglicht es Daten über den Benutzer abzufragen. Zum Beispiel ob ein Benutzer in einem bestimmten Kurs ist oder welche Rolle er hat.
  - news.impl
    Ein Manager mit dem man News bearbeiten, erstellen und löschen kann. Die News werden auf der Startseite des Portals angezeigt und können von Lehrenden bearbeitet werden.

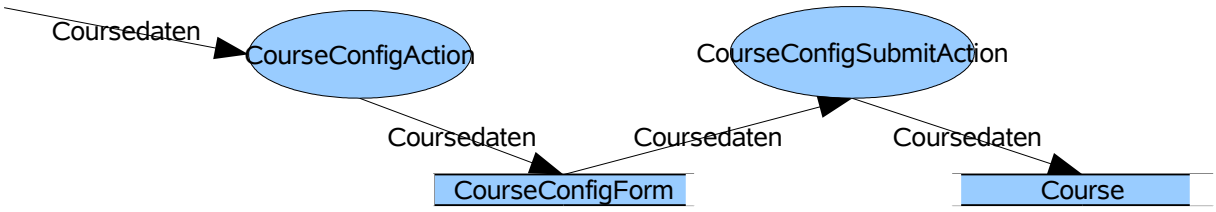
- download
  In diesem Paket gibt es mehrere Klassen, die den Download-Service implementieren.
- courses
  Dieses Paket enthält Klassen zum Verwalten der Veranstaltungen. So können z.B. Studenten in einen Kurs ein- oder ausgetragen werden, Lese- und Schreibrechte gesetzt werden, etc.

➢ engine:
Dieses Paket stellt mit Controllern und Managern fundamentale Dienste des Portals zur Verfügung. So können innerhalb des elatePortals Objekte in Form von Spring components geholt und gelesen werden. Die beinhalteten Managerobjekte übernehmen verschiedene Aufgaben:

- AdminThreads
  Manager, der regelmässig Aufgaben durchführen lässt, die dort auch erstellt werden können.
- ElatePortalConfigManager
  Managt das Laden und Speichern fundamentaler Portalkonfiguration. (z.B. Host und Port)
- FileRepositoryManager
  Ein einfacher Manager für den Zugriff auf Dateien ausgehend von einem festgelegten Wurzelverzeichnis.

Die Engine spielt vor allem beim Initialisieren des Portals eine wichtige Rolle. Der JetspeedPortalInitializer stellt dabei sicher, das notwendige Einträge wirklich in der Jetspeed Benutzerdatenbank existieren und dann vom Portal verwendet werden können. Dabei wird das Paket UserAttributesImporter verwendet um Benutzerinformationen in die Jetspeed Benutzerdatenbank zu importieren. Zum Speichern der Benutzerinformationen wird eine CSV-Datei verwendet, die im richtigen Verzeichnis abgelegt sein muss.

4.2 Konkrete Umsetzung der Model-2-Architektur



Das Grunddesign der Klassen folgt der Model-2-Architektur. Hier ein exemplarisches Beispiel: Sollen die Coursedaten eines Courses geändert werden, läuft dies nach folgendem Schema: Die 1. Anfrage des Webbrowsers wird an ActionServlet übermittelt, welches die CourseConfigAction (execute) aufruft. Die Cousedaten aus dem Formular werden automatisch in die ActionForm-Klasse(Course) übertragen, von wo sie von CourseConfigAction in die CourseConfigForm-Klassse übertragen werden. Danach kommt eine 2. Anfrage des Webbrowsers welche ebenfalls vom ActionServlet übermittlet wird und nun aber die CourseConfigSubmitAction(execute) aufruft. Die Coursedaten werden aus der ActionForm-Klasse(CourseConfigForm) mittels der CourseConfigSubmitAction in die Course-Klasse übertragen.