Design-Beschreibung zu WebAssign

1 Allgemeines

WebAssign ist eine Software, die für das Online-Verwalten von Übungen gedacht ist. Sie hilft bei der Verwaltung von Übungen indem es die Möglichkeit bietet, Übungsaufgaben zu publizieren sowie deren Lösungen entgegenzunehmen.
Genutzt wird die Software über ein WWW-Interface, wo sich alle Akteure anmelden können und je nach Rechten Aktionen durchführen können. Realisiert ist diese Software über Java Servlets mit Unterstützung von einer Datenbank (z.B. mysql) und CORBA. WebAssign wurde aus Eigeninteresse von der Fernuni Hagen entwickelt und dient somit als Lösung für einen Papier-losen Übungsbetrieb. Dabei wird bei dieser Software auch viel Augenmerk auf die Online-Eingabe und die automatische Korrektur von Übungsaufgaben gelegt.

2 Produktübersicht

Jeder Kurs hat eine Kursstartseite, über die man nach einem Login mit unterschiedlichen Rollen zu unterschiedlichen Seiten gelangt. Nachfolgend sind diese Rollen benannt:

Betreuer

Der Betreuer hat die Möglichkeit, Aufgaben zu verwalten und zu konfigurieren. Zudem kann er sich die Korrekturen und Einsendungen ansehen. Unter Konfiguration kann man vorhandene Übungsaufgaben ansehen und neue erstellen. Dazu gehören Aufgabentexte, Aufgabenstruktur sowie Muserlösungen und Hinweise für den Korrektor. Für Auto-Korrektur ist die Angabe von Informationen zu einem Korrektur-Modul nötig. muss man sowohl die Struktur der Aufgaben als auch Aufgabentexte.
Es ist weiterhin möglich, Studenten, Betreuer und Korrektoren anzumelden und zu bearbeiten. Das Versenden von emails an Teilnehmer eines Kurses ist ebenfalls möglich.

Korrektor

Der Korrektor kann sich die eingereichten Lösungen anschauen und diese bewerten.
Zunächst bekommt er eine Übersicht über die Übungsaufgaben sowie deren Status (unkorrigiert, beim Korrektor, korrigiert, freigegeben). Nach Auswahl von einer Übung kann er sich Lösungen der Studenten ansehen und diese korrigieren und bewerten. Es ist auch möglich, Übungen offline zu kontrollieren. Durch Freigabe einer Aufgabenserie gibt der Korrektor die Aufgaben quasi zurück - die Studenten können sich dann ihre Ergebnisse ansehen.

Student

Studenten können sich, nachdem sie sich zu einer Übung angemeldet haben, im System mittels Matrikel-Nummer und Passwort einloggen und haben dann mehrere Möglichkeiten zur Auswahl. Sie sehen eine Übersicht der gestellten Aufgaben mit ihren Unteraufgaben. Dabei können sie diese Aufgaben ansehen und bearbeiten. Es gibt die Möglichkeit, Übungsaufgaben durch automatische Korrektur interaktiv zu lösen oder Texte zu Aufgaben zu schreiben, die

nicht sofort korrigiert werden. Nachdem man der Meinung ist, fertig zu sein kann man seine Lösungen abschicken und schließt damit die Bearbeitung der Übungsaufgaben ab. Nach Freigabe kann er sich die Korrektur seiner Aufgaben ansehen und eine Musterlösung ansehen.

3 Grundsätzliche Design-Entscheidungen

Die Kommunikation mit der Aussenwelt erfolgt über ein zentrales Java Servlet, welches aus der URL sowie den Request-Informationen (mittels GET, POST oder PUT übermittelt) die Anfrage an andere Module weiterreicht.
Die Software lässt sich von der Grundstruktur in 4 Kategorien unterteilen.
Als Grundlage dient das WebAssign-Basis-Paket, welches die von aussen kommenden Requests entgegennimmt. Es autentifiziert den Nutzer und gibt den Request bei Erfolg an die nächste Stufe weiter.
Die 2. Stufe sind Dienste, die für die Funktionalität von WebAssign sorgen. An diese Dienste werden die Anfragen aus der ersten Stufe weitergeleitet. Diese Dienste lassen sich in eine von 3 Kategorien je nach Art des HTTP-Requests einordnen: POST, GET oder PUT. Die Dienste greifen gegebenfalls auf die Daten (das Modell) der Software zu, die die 3. Stufe bilden. Hier sind verschiedene Daten, die in WebAssign verwendet werden, in Klassen gekapselt, so z.B. Kurs, Aufgabe oder Benutzer. Die Daten selbst beziehen die einzelnen Klassen aus der untersten Stufe.
Als vierte und unterste Stufe ist eine Ebene von Klassen für den Datenbank-Zugriff zu finden. Über diese werden eingegeben Daten permanent gespeichert und können abgerufen werden.

Zur Realisierung von WebAssign werden neben Java Servlets und Klassen weitere Technologien eingesetzt:
Um Studenten zu identifizieren kann auf einen zentralen LDAP-Verzeichnisdienst zugegriffen werden. Somit soll es möglich sein, zentrale Daten über Studenten zu erhalten, so dass sich nicht jeder Student immer wieder neu anmelden muss, sondern auf die Daten vom Einschreiben an der Universtität zugegriffen werden kann.
Um die Möglichkeit zu bieten, Autokorrekturen zu realisieren wird Corba verwendet. So können eingesandte Lösungen zu anderen Servern verschickt werden, die für die Autokorrektur zuständig sind und der Status dieser Korrektur entgegengenommen werden und dem Student sofort wieder mitgeteilt werden. Somit müssen die Algorithmen zur Autokerrektur nicht an diese Software gebunden werden und auch die Programmiersprache ist nicht festgelegt, da es CORBA-Implementierungen für viele verschiedene Programmiersprachen gibt.
Die Speicherung der Daten erfolgt wie oben bereits erwähnt mittels einer Datenbank. Der Zugriff auf Daten ist damit sehr schnell und man kann auch zur Lastverteilung die Datenbank auf einem anderen Server laufen lassen als WebAssign selbst. Welche Datenbank verwendet wird schreibt die Software nicht vor. Skripte zur Tabellengenerierung werden für MySQL und Oracle mitgeliefert. Der Zugriff auf die Datenbank erfolgt mit JDBC, für welches es für alle gängigen Datenbanken Treiber gibt.

4 Paket- und Klassenstruktur

Die Klassen von Webassign sind in die folgenden Pakete unterteilt:

- WebAssign
  Dieses Paket entspricht dem Basis-Paket (Stufe 1) von WebAssign. Das Paket baut sich wie folg auf:

  - webassign
    Diese Klasse ist das zentrale Servlet von WebAssign. Es leitet die Anfragen, die von den Browsern der Benutzer gesendet werden, an die Klasse *WebApplicationServer* weiter, indem es die Anfrage je nach Typ an eine Methode übergibt, die GET-, POST- oder PUT-Requests verarbeitet.

  - mywebassign
    Diese Klasse ist der Klasse *webassign* sehr ähnlich. Sie unterscheidet sich dadurch von *webassign*, dass es sich hierbei um einen personalisierten Zugriff auf WebAssign handelt.

  - WebApplicationServer
    Für diese Klasse existiert nur eine Instanz, welche in der Klasse *webassign* statisch angelegt ist. Diese Klasse analysiert die Anfrage eines bestimmten Typs und leitet es nach erfolgreicher Autorisierung des Benutzers an Klassen aus der Paket **WebAssign.services** weiter.

  - WebAssignServer
    Diese Klasse enthält ausschließlich Klassenmethoden und -variablen. Sie dient dazu, Ressourcen zu verwalten, welche die gesamte Software betreffen, so zum Beispiel Informationen über die Systemumgebung.

  - WebAssignRequestInfo
    Diese Klasse enthält Informationen über das Request des Benutzers und bietet die Möglichkeit, Werte für die Antwort (Response) zum Benutzer zu setzen. Sie enthält unter anderem auch die Information, ob der Benutzer autorisiert ist oder nicht.

  - WebAssignAdminThread
    Diese Klasse wird beim Initialisieren des WebAssign-Servlets gestartet und führt administrative Aufgaben parallel zum Betrieb von WebAssign aus.

  - Kontext Diese Klasse dient der zentralen Speicherung von Informationen zu einem bestimmten Kontext, damit nicht bei jeder Verwendung die Datenbank befragt werden muss. Dabei sind aber nur einfache und keine komplexeren Informationen gespeichert.

- WebAssign.services
  Dieses Paket entspricht der 2. Stufe von WebAssign. Sie enthält die verschiedenen Services, untergliedert in weitere Pakete je nach Art des Requests (GET, POST oder PUT) Durch diese Untergliederung gibt es in diesem Paket nur eine Klasse:

  - WebAssignService
    Dies ist die Basisklasse für alle Services. Sie enthält Funktionen, die von allen abgeleiteten Services gebraucht werden könnten.

  Alle weiteren Services leiten sich indirekt aus der Klasse *WebAssignService* ab. In jedem der Unterpakete **WebAssign.services.get**, **WebAssign.services.post** und **WebAssign.services.put** gibt es eine Klasse *WebAssignGetService*, *WebAssignPostService* und *WebAssignPutService*, welche von *WebAssignService* abgeleitet ist und von denen sich jeweils alle Services des entsprechenden Pakets ableiten.
  Zu den typischen GET-Services gehören Anfragen, die lediglich Informationen fordern, so zum Beispiel Übersichts-Seiten oder auch Eingabemasken.
  Bei POST-Services werden Informationen vom Benutzer an das System gesendet, so zum Beispiel nach dem Ausfüllen von Formularen wie Aufgabenlösungen und Login-Daten.
  Die PUT-Services sind dafür zuständig, ganze Dateien hochzuladen. Dies ist beim Layout-Anpassen von HTML-Seiten oder beim Upload von Daten der Korrektoren.

- WebAssign.model
  Dieses Paket entspricht der dritten Stufe von WebAssign. Alles, was man in der realen Welt als ein Objekt sehen könnte, hat hier eine Klasse, die Informationen über das jeweilige Objekt aufnehmen und wiedergeben kann.

  - WebAssignModel
    Von dieser Klasse leiten sich alle Klassen dieses Pakets ab. Sie enthält vorgefertigte Methoden, um Datenbankabfragen durchzuführen und Informationen zu Kursen herauszugeben.

  Von der Klasse *WebAssignModel* werden nun viele Klassen abgeleitet, von denen im Folgenden einige aufgezählt und ggf. erklärt werden:

  - Benutzer
    Von dieser Klasse sind alle Benutzer des Systems abgeleitet, so die Klassen *Administrator, Betreuer, Gast, Korrektor, Student, Unauthorisiert.*

  - Aufgabe
    Diese Klasse enthält alle Informationen über eine Aufgabe, so zum Beispiel den Aufgaben-Namen, die Aufgaben-Nummer oder die Art der Aufgabenabgabe.

  - Einsendung
    Diese Klasse enthält die Lösungs-Einsendung eines Studenten mit allen nötigen Informationen wie z.B. der Matrikel-Nummer des Studenten und Angaben, zu welcher Aufgabe dies die Lösung ist.

  - Korrektur
    Diese Klasse beinhaltet Daten zu Korrektur einer *Einsendung.*

- WebAssign.database
  Dieses Paket entspricht der vierten Stufe von WebAssign. Sie enthält Klassen, um das

Zusammenspiel von WebAssign und der Datenbank zu gewährleisten. Der Zugriff auf die Datenbank erfolgt ausschließlich durch Klassen aus diesem Paket. Damit ist der Rest der Software nicht direkt an die Datenbank gebunden.

  - AbstractDatabase
    Diese Klasse dient als abstrakte Oberklasse zu allen weiteren Klassen dieses Pakets. Sie stellt die Basis-Operationen für die elementaren Datenbankoperationen zur Verfügung. Diese Klasse dient als Wrapper für das JDBC-API.

Die Klasse *AbstractDatabase* findet sich für spezielle Datenbanken in den folgenden Klassen abgeleitet, wobei einige spezifisch für die Fernuni Hagen sind:

  - MysqlDatenbank
  - OracleDatenbank
  - InformixDatenbank
  - HISDatenbank
  - URZDatenbank

Des weiteren existiert noch eine Hilfsklasse:

  - Tupel
    Diese Klasse enthält ein n-Tupel an Daten, wobei jedem der Einträge des Tupels ein Name zugeordnet ist, über den man auf den Inhalt zugreifen kann. Es ist also eher eine Hash-Tabelle als ein Tupel.

- WebAssign.corba
  Dieses Paket bietet eine Anzahl von Klassen, um WebAssign mit externen Servern zu verbinden, die für die Auto-Korrektur von Aufgaben zuständig sind.
- WebAssign.kserver
  Dieses Paket enthält eine Anzahl von Klassen, die den Download und Upload von Korrekturen zu ermöglichen.
- WebAssign.util
  Dieses Paket enthält Hilfs-Klassen, die von verschiedenen Klassen benötigt werden. Einige sind:

  - Mailer
    Diese Klasse ist zum Versenden von Mails zuständig
  - Exec
    Diese Klasse ist dafür zuständig, externe Programme auszuführen
  - PageParser
    Diese Klasse ist dafür zuständig, HTML-Templates zu parsen, damit Platzhalter durch Daten ersetzt werden können

- DirectoryReader
  Diese Klasse ermöglicht das Einlesen eines Verzeichnisses und das zurückgeben der in diesem Verzeichnis befindlichen Dateien

Es gibt noch wenige weitere Pakete, deren Bedeutung allerdings eher gering ist und deshalb hier nicht erwähnt wird.
Eine komplexe Übersicht über die Klassenbeziehungen sieht man in der mitgelieferten Rational Rose-mdl-Datei.