Abschlussbewertung

Inhaltsverzeichnis

Projektbeschreibung:

Ziel war es ein Softwaresystem zur Bestands- und Kundenverwaltung einer Videothek mit einer Onlinepräsenz zu entwickeln.

Für die Bestands- und Kundenverwaltung sowie das Leih- und Reservierungssystem wurde eine robuste Swing-Applikation erstellt. Die Onlinepräsenz mit der Möglichkeit im Bestand nach Artikel zu suchen und Artikel zu buchen wurde mit JSF erstellt. Für die Persistenz kamen Hibernate und HSQL-DB zum Einsatz.

Vorlesungsbewertung:

Die Vorlesung im Allgemeinen haben wir als sehr gut empfunden, besonders sinnvoll war das Skript mit den didaktischen Lücken.
Gut war es auch Kapitel wie zum Beispiel:
„Guter Programmierstil“, „JUnit“, „Anforderungen an den Code“, „MulTEx“ schon in SWPI zu behandeln. Dadurch waren diese Techniken schon bekannt und konnten in SWPII sofort angewendet werden.
Sehr hilfreich waren die regelmäßigen Besprechungstermine mit dem Dozenten, dadurch konnte oftmals Problemen schon im Vorfeld aus dem Weg gegangen werden.

Technologiebewertung:

JAVA 5:

Obwohl wir bei weitem nicht alle Möglichkeiten von Java 5 ausgeschöpft haben, waren einige Neuerungen der Version 5 sehr hilfreich, um den Quellcode vor allem typsicher und redundanzfrei zu halten.

Es kamen unter anderem foreach-Schleifen, Annotations, Enums und Generics zum Einsatz. Leider konnten wir die Annotations nicht durchgängig verwenden, da uns diese Möglichkeit der Kennzeichnung von überschriebenen Methoden erst recht spät bekannt wurde.

Als Nachteilig bei der Verwendung von Enums empfanden wir, dass Hibernate 3 keine und JSF 1.1 nur eine recht rudimentäre Unterstützung dafür bieten.

JSF / MyFaces:

JSF ist ein mächtiges Framework zur Erstellung von graphischen Benutzeroberflächen für Webanwendungen. Es bietet dem geübten Java-Entwickler einen recht leichten Einstieg, da man auf bekannte Techniken wie Eventhandling oder das MVC-Prinzip zurückgreifen kann.
Komfortabel sind auch die sogenannten ManagedBeans, die JSF automatisch zur Verfügung stellt, wenn sie benötig werden. Somit muss sich der Oberflächenentwickler nicht darum kümmern, ob bestimmte Objekte schon geladen sind und verwendet werden können.
Aufgrund von einigen Problemen der JSF ri 1.101 haben wir uns entschieden die freie Implementierung MyFaces in der Version 1.1.3 zu verwenden.
Als sehr nachteilig erwies sich dabei das Mischen von unterschiedlichen Technologien, das aufgrund des Umfangs der Weboberfläche zwingend notwendig wurde. Um das Layout vernünftig aufbauen zu können, kamen in unserem Projekt zusätzlich Struts-Tiles hinzu. Dadurch ergibt sich das Problem, dass JSF die ManagedBeans in den über Tiles eingebundenen jsp-Seiten spät oder gar nicht bereitstellen kann.

Swing:

Obwohl Swing an sich keine komplett neue Technologie für uns darstellte, war es vor allem die Komplexität des Projekts, was uns so manche bisher noch nicht beleuchtete Seite der UI Programmierung näher brachte. Eigens entwickelte Komponenten wie z.B. eine Enum Listbox werden wir auch in zukünftigen Projekten einsetzen können. Wir können es durchweg empfehlen das Event Handling über UIActions zu realisieren, da dies unserer Meinung nach die sauberste Lösung mit minimalen Schnittstellen ist. Leider konnten wir aufgrund der Komplexität und des Zeitmangels nicht auf die Aspekte der Layout-Programmierung eingehen.

MulTEx:

Wir haben in SWP2 mit MulTEx zum ersten Mal das Konzept des zentralen Exception-Handlings in einem Projekt angewendet.
MulTEx ist ein Framework für das Exceptionhandling über mehrere Schichten. Es verbessert die Auswertung der Ursachenkette, die Operationen werden zu den oberen Schichten hin immer abstrakter, dementsprechend auch die Ausnahmen, die ihr Versagen kennzeichnen. In den unteren Schichten ist die jeweilige Ursache genau bekannt und es kommt darauf an, diese gezielt zu erfassen, zu sammeln und zugänglich zu machen.
Besonders positiv zu bewerten ist dabei die Tatsache, dass das von uns fertig gestellte System dadurch sehr übersichtlich und robust wurde.

Hibernate:

Die Wahl des OR-Mapping Tools fiel auf Hibernate, das zwar kein Standard ist, aber das wohl am meisten genutzte. Die Basiskonfiguration mit einer lokalen Anbindung der HSQLDB Datenbank war schnell funktionsfähig und recht unkompliziert. Vorteilhaft war vor allem das alle Teammitglieder lokal auf ihrer Datenbank arbeiten konnten und wir nicht darauf angewiesen waren einen zentralen DB Server aufzusetzen. Dies hatte aber leider zur Folge, dass auch viel Zeit für die Synchronisation bzw. Fehlerbeseitigung in den einzelnen DBs verlorenging.
Schwierig war es auch die teilweise komplexen Hierarchien auf Hibernate Mappings abzubilden, sowie Queries für die Suche nach solchen zu entwickeln, was aber definitiv zu einem tieferen Einblick in Hibernate geführt hat. Teilweise mußtem die Hibernate Queries durch konventionelle SQL Statements ergänzt werden, die sich wiederum nachteilig auf die Wartbarkeit der DB Schicht auswirken (z.B. Abfrage nach bestimmten Werten in einer Collection eines Objekts). Insgesamt können wir die Kombination aus Hibernate und HSQLDB nur empfehlen, da man relativ schnell und unkompliziert Persistenz erreicht.

JUnit:

JUnit ist ein Framework zum automatisierten Unit - testing für Java.
Unit - testing wurde anfänglich hauptsächlich im Umfeld des sogenannten "Extreme Programming" eingesetzt, erfreut sich jedoch in der jüngeren Geschichte zunehmender Beliebtheit. Hauptanwendung findet das Framework im "Test driven Development". Hierbei werden im Normalfall zuerst die Testfälle spezifiziert und danach der Code entwickelt, der diese Testfälle erfüllt. Bei diesem Vorgehen ist stets eine Korrektheit des entwickelten Codes und vor allem eine vollständige Abdeckung mit Testfällen gewährleistet. Die zu testenden Units werden durch bestimmte Klassen und Methoden dargestellt. Ein JUnit - Test wird ebenso wie die zu testende Software in Java codiert und kann entweder fehlschlagen oder erfolgreich sein.
Es gibt zwei verschiedene Arten von Fehlschlägen: Failures und Errors.
Failures sind Testergebnisse die von dem Erwarteten Ergebnis abweichen. Zum Beispiel wenn der Wert einer Variablen vom erwarteten Wert abweicht oder eine andere als die erwartete Exception geworfen wurde.
Errors sind Exceptions die im Testverlauf nicht erwartet wurden. Bei erfolgreichem Testdurchlauf leistet jeder Testling seine erwartete Funktionalität. Dieses Vorgehen ist zur Analyse weitaus geeigneter als sogenannte "Blackboxtests", bei denen ein System als Ganzes auf seine Funktionalität getestet wird, da bei diesen oft nicht einfach herauszubekommen ist wo ein eventuelles Fehlverhalten entstanden ist.

Arbeitsumfang:

Der Arbeitsaufwand für das Projekt war sehr hoch. Das Projekt nahm unter der Woche ca. 2-3 Stunde pro Abend (nach Uni/Arbeit) sowie ca. 25 Stunden des Wochenendes in Anspruch. Wir empfehlen allen Kommilitonen die das Software Projekt noch vor sich haben, die Belastung durch andere Fächer gering zu halten.

Resume & Teamarbeit:

Die Teamarbeit in unserem Projekt lief insgesamt sehr gut. Alle Mitglieder waren sehr motiviert und interessiert.
Das Arbeiten verlief hauptsächlich online, über Tools wie Skype, VNC und Trillian. Sehr nützlich war der SVN-Server des Software-Labors um verteilt arbeiten zu können. Die Aufteilung der einzelnen Bereiche, wie wir sie in SWPI festgelegt hatten, konnte nicht konsequent durchgehalten werden und so waren einige Mitarbeiter mit mehreren Bereichen des Softwareprojekts beschäftigt.

Abschließend können wir sagen, dass es eine sehr interessante Erfahrung war ein vollständiges Projekt von Anfang bis Ende zu entwickeln. Wir haben in beiden Semestern viel gelernt und für die Zukunft mitgenommen.