Schlussbewertung

FB VI

SOFTWAREPROJEKT II

WS 09/10

TEAM

Alexander Kalden	729631
Dominik Eckelmann	745097
Marcel Pierry	736477
Julian Heise	745011
Besha Taher	741297

Inhalt

Schlussbewertung

Bewertung der Technologien

1. Java Server Faces (JSF) und Frontend allgemein

Das Zusammenspiel mit den JSPs und den Backing Beans ist sehr gut gelungen.

Den JSP-Code haben wir nicht generiert bzw. per Visual-JSF erstellt sondern selbst erstellt, dadurch hatten wir mehr Spielraum und konnten individuellere JSP-Seiten erstellen, es hat auch mehr Zeit in Anspruch genommen. Wir haben außerdem bestimmte Teile der JSPs, die statisch sind, ausgelagert. Das hat uns die Arbeit erspart, sie immer wieder bei jeder Seite neu zu schreiben/aktualisieren.

Ein spezielles Problem war der Umgang mit Variablen, die Auflistungsobjekte darstellten. Diese Variablen konnten nicht z.B. in if-Abfragen verglichen oder geprüft werden.

Einerseits mussten die JSPs getestet werden, andererseits musste man gleichzeitig auch die Funktionalitäten testen, d. h. DB- und LG-Funktionalitäten. Das hat das Arbeiten an den JSPs verlangsamt, weil man erst auf die DB- und LG-Korrektur warten musste, bevor man eine JSP weiter testen kann.

Ein JSF-Projekt aufzustellen und eine Datenbank anzubinden hat sehr viel Zeit in Anspruch genommen. Die richtigen Konfigurationen für den Server in Zusammenhang mit den JSF-Libraries hat Fehlermeldungen beim Starten des Servers verursacht, die nicht genau waren bzw. keine genauen Aussagen enthielten darüber, was die Fehlerursache war. Diese Fehlersuche hat am meisten Zeit in Anspruch genommen. Aber nachdem das JSF-Projekt einmal richtig konfiguriert war, konnten wir das Projekt bei SVN hochladen und somit konnten die anderen Teammitglieder über den Eclipse ganz einfach das fertig eingestellte Projekt importieren und hatten keine Probleme mit den Einstellungen mehr.

Style-Eigenschaften wurden alle mit CSS gemacht und zum größten Teil von der Struktur getrennt in einer externen Datei angelegt.

Im Frontend wurden Validierungen per JavaScript durchgeführt mit Hilfe der JavaScript-Bibliothek jQuery.

Zum Hochladen von Dateien (z.B. Bild für ein Produkt) wurde die Apache Tomahawk-Bibliothek eingesetzt.

Anfangs wurde für jede Art von Fehlermeldung eine JSP-Seite angelegt, d.h. es gab mehrere JSPs für die verschiedenen Fehlermeldungen. Daher wurde eine neue Vorgehensweise aufgebaut, wodurch es nur noch zwei zentrale JSPs gibt für Erfolg und Fehler.

2. Java Beans

Die Oberfläche wurde von Anfang an so ausgelegt, dass jeder HTML-Dialog ein eigenes Java Backing Bean bekommen sollte. An diesem Design hat sich im Laufe des Entwicklungsprozesses nichts mehr geändert. Auch die Grundstruktur der Beans hat sich nach dem ersten Implementationszyklus nicht mehr wesentlich geändert. Lediglich gegen Ende des Projektes kamen einige etwas größere Änderungen, da für alle Beans die Transaktionslogik einheitlich angepasst werden musste, was jedoch kein nennenswertes Problem darstellte.

Auch das Exception-Handling findet hier wie geplant statt. Der einzige Zusatz ist, dass eine Transaktion selbst ihre Exceptions fängt, loggt und dann weiterreicht, sodass die Bean nur entsprechende Navigation Rules auslöst, um dem Benutzer eine angepasste Fehlermeldung anzeigen zu können. Der Ursprüngliche Fehler ist dann im Log für Betreiber und Entwickler ersichtlich.

Insgesamt erfüllen die Java Beans nur die Funktion einer etwas intelligenteren Zugriffsschicht, die zwischen der LG-Schicht und JSPs vermittelt und Fehler behandelt. Es werden stets Aktionen von hier aus verteilt und Navigation Rules ausgelöst.

3. JUnit

Mit Hilfe von JUnit 4 wurde das komplette Projekt getestet und es hat sich als sehr hilfreich bewiesen. Dadurch konnten zuständige Personen im Backend ihr Code testen, bevor andere zuständige Personen im Frontend mit Programmierfehlern aus dem Backend beschäftigen müssen. Außerdem konnte man dadurch bei Fehlern im Frontend sicherstellen, ob der Fehler in der DB-Schicht, LG-Schicht oder UI-Schicht befindet und so konnten Fehler schneller gefunden werden.

4. Hibernate

Die Datenbank wurde mit Hibernate erstellt und verwaltet. Durch die Hibernate-Annotationen war es sehr einfach und übersichtlich, Tabellen anzulegen, miteinander zu verknüpfen oder neue Spalten anzulegen. Außerdem wurden Datenbankabfragen erheblich erleichtert durch die Hibernate Query Language (HQL). Dadurch erspart man sich eine Menge komplizierter MySQL-Abfragen. Insbesondere Fehlermeldungen und Exceptions von Hibernate waren von Vorteil, da sie sehr genau beschrieben sind und genau auf die konkreten Fehlerquellen verweisen.

5. Java

Das gesamte Projekt wurde mit Java entwickelt. Java in Kombination mit Java Server Faces (JSF) und Hibernate hat sich als sehr hilfreich erwiesen, da die eingesetzten Frameworks gut miteinander eingesetzt werden konnten. Es sind keine besonderen Probleme in Zusammenhang mit Java aufgetreten.

Etwas aufwändig war das Schreiben der Java Doc-Kommentare, da sie immer wieder mit dem Code aktualisiert werden mussten und der Code sich oft geändert hat. Unserer Erfahrung nach werden genau aus diesem Grund Java-Docs nicht geschrieben, weil sie sonst zu unnötigen Verwirrungen führen, wenn sie nicht immer aktualisiert werden und weil es sehr viel Zeit kostet, sie zu aktualisieren.

6. Rich Domain Model

Nach einigen Überlegungen haben wir uns für das Rich Domain Model entschieden. Das Rich Domain Model hat jedoch anfangs zu Problemen geführt, da in der LG-Schicht sowohl Attribute als auch Logik-Operationen sind. Die Attribute wurden mit Hibernate-Annotationen versehen und wurden somit je Klasse eine Tabelle und je Attribut eine Spalte in der jeweiligen Tabelle abgebildet. Das gehört unserer Ansicht nach in die DB-Schicht, musste aber aufgrund des Rich Domain Models in der LG-Schicht konfiguriert werden. Außerdem ist dadurch die LG-Schicht unübersichtlich geworden mit zunehmendem Code im Laufe der Entwicklung. Es war nicht einfach, die Zuständigkeiten ganz exakt zu trennen, da die Hibernate-Annotationen Zuständigkeit der Personen für die DB-Schicht waren, sich aber in der LG-Schicht befanden.

Bewertung der Werkzeuge

1. Eclipse

Als Entwicklungsumgebung wurde Eclipse eingesetzt. Durch die vielen PlugIns hat Eclipse bei der Entwicklung große Vorteile mitgebracht. Das Subclipse-PlugIn hat dem Umgang mit SVN wesentlich erleichtert. Außerdem gibt es bei Eclipse die Möglichkeit, durch wenige einfache Schritte einen Server zu konfigurieren und mit dem Projekt zu verknüpfen.

2. Enterprise Architect

Bei der Erstellung von Klassendiagrammen, Use-Case-Diagrammen und sonstigen Diagrammen wurde Enterprise Architekt 7 verwendet. Diese Software hat den besonderen Vorteil, dass man aus einem Diagramm Code generieren kann in beliebigen Programmiersprachen und umgekehrt aus dem Code Klassendiagramm generieren. Das hat sich als besonders wichtig erwiesen in Softwareprojekt II, da sehr oft das Klassendiagramm bzw. der Code geändert werden musste und das Projekt einen immer größeren Umfang angenommen hat, so dass es uns gar nicht möglich wäre, bei jeder Änderung das Klassendiagramm bzw. Code selbst mit zu ändern.

3. Google-Groups

Für die Kommunikation innerhalb des Teams wurde in Softwareprojekt I Google Groups eingesetzt. Das war eine gute Möglichkeit, alle Teammitglieder in Überblick zu haben. Durch die Möglichkeit, Dokumente hochzuladen, konnten Enterprise Architekt und Word Dateien gut ausgetauscht werden, ohne dass man diese Dokumente immer per E-Mail an das gesamte Team schicken muss, insbesondere, weil im Team anfangs noch nicht so häufig SVN verwendet wurde (da bei Softwareprojekt I noch nicht viel programmiert werden musste).