TMS

Schlussbewertung

Datum 29.01.2012

Version 1.0

Team Nadin Schulte (766254)
Felix Stiehler (766205)
Tim Kreutzer (761752)
Norman Lange (761766)

Schlussbewertung

Technik

Java

Java ist eine weit verbreitete stark objektorientierte Programmiersprache. Wir haben uns für Java entschieden, da alle Projektmitglieder mit der Sprache vertraut sind und wir diese auch im Rahmen anderer Lehrveranstaltungen verwendet haben. Auch wenn niemand von uns bisher Java für Webapplikationen eingesetzt hatte, haben wir uns alle schnell zurecht gefunden und keine größeren Probleme damit gehabt.

Wicket

Das Oberflächenframework Wicket ist Open Source Software und wird von der Apache Foundation betreut und finanziert. Wir haben es verwendet, da es uns simpel und leistungsfähig erschien. Wicket folgt klaren Konzepten, welche sich in allen Teilen des Frameworks wiederfinden. So gibt es eine Klasse für jede Webseite und immer ein Objekt für jedes dynamische HTML-Element.

Probleme bereitete uns lediglich der Requestcycle, welches das Grundkonzept darstellt wie Wicket Webseiten erstellt. Eingriffe in diesen Prozess erwiesen sich als recht kompliziert. Diesbezüglich fällt auch die etwas mangelhafte Dokumentation von Wicket auf.

Hibernate

Hibernate ist eine Open Source Implementierung von JPA, welche sich großer Beliebtheit erfreut. Wir haben uns vor allem wegen seiner großen Verbreitung für ihn entschieden. Hibernate zeichnet sich vor allem durch viele Einstell- und Eingreifmöglichkeiten aus. So kann (und muss) man etwa Transaktionen von Hand verwalten und immer aufpassen, welche Objekte sich schon in einer Hibernate Session befinden, wenn man neue Objekte lädt oder speichert. Diese Flexibilität hat sich für uns eher als Problem, denn als Vorteil herausgestellt, da wir dadurch oft schwer nachvollziehbare Fehler hatten.

JUnit

Dieses Semester haben wir JUnit zum ersten Mal intensiv genutzt. Die Einarbeitung ist sehr einfach. Besonders für testgetriebene Entwicklung ist dieses Framework ein Muss.

MulTeX

Generell ist die Einarbeitung in dieses Framework sehr einfach möglich, was auch an der guten Dokumentation liegt. Nur wenn es etwas spezieller wurde, hatten wir doch ein paar Schwierigkeiten. So zum Beispiel beim Einsammeln der Ausnahmemeldungen mittels Maven. In der Dokumentation steht nur, wie es mit Ant funktioniert. Herr Knabe hat uns dann empfohlen, das Antrun Plugin für Maven zu verwenden, womit man Ant-Tasks mit Maven ausführen kann. Aber auch damit, lief noch nicht alles auf Anhieb. Letztendlich haben wir unverhältnismäßig viel Zeit zur Konfiguration benötigt.

Auch das Zentrale Ausnahmemelden, war in Wicket nicht so einfach umzusetzen. Wir haben es schließlich mit dem Template Method Pattern realisiert.
Nachdem wir alles richtig konfiguriert hatten, hat sich MulTeX aber als sehr praktisch erwiesen.

MySQL

Wir haben uns für MySQL als Datenbank entschieden. Viel von MySQL selber haben wir aber gar nicht mitbekommen, da Hibernate den Großteil der Arbeit übernommen hat. Wir hätten uns vermutlich auch für (fast) jede andere Datenbank entscheiden können.

Werkzeuge

Eclipse

Da die Mehrheit von uns hauptsächlich in Eclipse programmiert, haben wir uns für diese IDE entschieden. Eclipse ist in Sachen Erweiterbarkeit wahrscheinlich unschlagbar. Es gibt unzählige Plugins, mit der man seine IDE nach Belieben zusammenstellen und konfigurieren kann. Für uns relevant waren hauptsächlich das m2eclipse Plugin (für Maven) und Subversive (für SVN).

SVN

Als Versionierungstool haben wir das schon seit vielen Jahren bewährte Subversion (SVN) verwendet, welches uns treue Dienste geleistet hat. Probleme hatten wir lediglich mit unserem Eclipse-Plugin für SVN - Subversive -, welches sich als recht sperrig und unübersichtlich herausgestellt hat. Gerade wenn Konflikte auftraten, konnte man diese nur schwer und nach mehreren Versuchen lösen. Allerdings hatten wir keine Probleme, die sich nicht auch lösen ließen.

Maven

Wir hatten vorher noch nie mit einem Build-Management-Tool wie Ant oder Maven gearbeitet. Jedoch war die Einarbeitung dank m2eclipse einigermaßen problemlos. Es gab auch schon einen Archetype für Wicket, womit wir gleich starten konnten. Bei Problemen hat uns meistens die Maven Dokumentation weitergeholfen.

Maven macht die Projektarbeit deutlich bequemer. So kann man mittels Goals und Plugins z.B. die JavaDocs-Generierung oder auch das Einsammeln der Exception-Meldungen automatisieren und genau bestimmen, zu welchen Zeitpunkt (in welcher Lifecycle-Phase), welches Ziel (Goal) ausgeführt werden soll.

Maven ist für jedes Softwareprojekt zu empfehlen.

Fazit

Alles in allem sind wir mit dem Endergebnis doch recht zufrieden. Wir haben alle Muss-Anforderungen unseres Pflichtenheftes erfüllt und unsere Software läuft – soweit ersichtlich – ohne größere Bugs.

Doch viel wichtiger ist, dass wir während dieses Softwareprojekts viel gelernt haben. Zum einen den Umgang mit neue Techniken und Werkzeugen wie Wicket, Hibernate oder Maven und zum anderen aber auch vieles, was man besser machen kann. So ist zum Beispiel zu empfehlen, von Anfang an testgetrieben zu entwickeln. Auch sollten alle eingesetzten Techniken gleich zu Beginn ordentlich konfiguriert werden.

Desweiteren sollte man sich vorher eine geeignete Kommunikationsstrategie zwischen den Projektmitgliedern überlegen. Mal haben wir uns per Mail, mal per Skype und mal über Assembla abgesprochen. Eine einzige Plattform wäre übersichtlicher gewesen. Trotzdem hat die Teamarbeit gut funktioniert.

Die wöchentlichen Rücksprachen und Meilensteine haben uns motiviert, stetig an unserem Projekt weiterzuarbeiten. Die Rücksprachen waren außerdem eine gute Anlaufstelle um Probleme zu besprechen und zu lösen.

Abschließend wollen wir aber festhalten, dass die Projekt- und Teamarbeit trotz aller Probleme viel Spaß gemacht hat. Ein herzliches Dankeschön an Professor Knabe für seine Unterstützung.