Komponentenmarktplatz für Enterprise Java Beans

Stefan Brüggemann, Jens Happe, Stefan Hildebrandt, Sascha Olliges, Heiko Koziolek,
Florian Krohs, Philipp Sandhaus, Rico Starke, Christian Storm, Timo Warns, Stefan Willer

Universität Oldenburg, Fachbereich Informatik, Abteilung Software-Engineering
`http://se.informatik.uni-oldenburg.de`

Dezember 2002

Im Bereich der Software-Entwicklung findet ein Wechsel von der rein objekt-orientierten hin zur komponentenbasierten Entwicklung statt [4]. Voraussetzung für eine große Verbreitung dieser Methoden und Techniken sind Komponentenarchive [6]. Diese sollen Software-Entwickler unterstützen, selbst entwickelte Komponenten anzubieten und geeignete bereits bestehende Komponenten für eigene Systeme zu finden. Nur mit Hilfe von strukturierten Archiven kann eine Wiederverwendung von Komponenten ermöglicht werden.

In einer Lehrveranstaltung in Form einer Projektgruppe wurde an der Carl-von-Ossietzky Universität Oldenburg ein Marktplatz für und mit Enterprise Java Beans entwickelt. Projektgruppen sind Veranstaltungen im Umfang von 8 SWS im Hauptstudium der Informatikausbildung an der Universität Oldenburg, in denen bis zu zwölf Studierende im Zeitraum von zwei Semestern gemeinsam ein größeres Projekt bearbeiten. Dies fand im Sommersemester 2002 und Wintersemester 2002 / 2003 in der Abteilung Software-Engineering unter der Leitung von Prof. Dr. Wilhelm Hasselbring statt. Vorgaben der Lehrveranstaltung waren, dass eine Web-Anwendung gemäß der J2EE-Architektur [5] implementiert wird, die sowohl im Internet als auch in firmeninternen Netzen eingesetzt werden kann. Dafür sollten eine Benutzerverwaltung und ein Ordnungssystem für Komponenten entwickelt werden. Eingestellte Komponenten sollten formal spezifiziert werden können, um die Aufindbarkeit zu gewährleisten. Ein Screenshot des Marktplatzes ist in Abbildung 1 zu sehen.

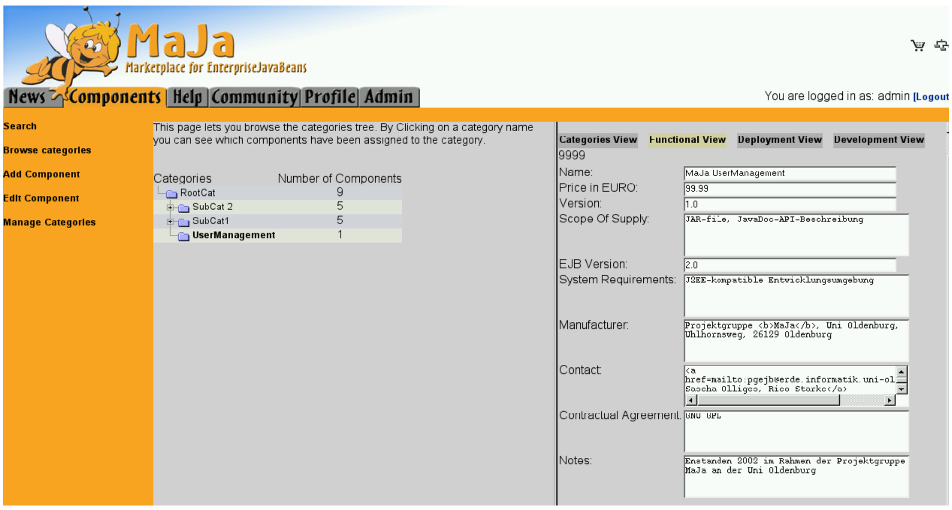

Abbildung 1: Screenshot des Marktplatzes

Technologien

Der Komponenten-Marktplatz ist als Web-Anwendung gemäß der J2EE-Architektur nach einem Model-View-Controller Muster [2] realisiert worden (Abb.2). Zur Modellierung der Entitäten (*Model*) wurden die Möglichkeiten der EJB-Architektur genutzt, wie „Container-Managed Relations" (CMR) und „Container-Managed Persistence" (CMP). Die graphische Benutzeroberfläche (*View*) wurde mit Hilfe von Java Server Pages und Servlets realisiert. Dabei wurde Struts aus dem Jakarta-Projekt (`http://jakarta.apache.org/struts/`) als Framework eingesetzt.

Die Geschäftslogik ist komplett in Enterprise Java Beans implementiert. Insbesondere sind die Funktionalitäten, die Benutzern zur Verfügung gestellt werden (*Controller*), in Session Beans gekapselt.

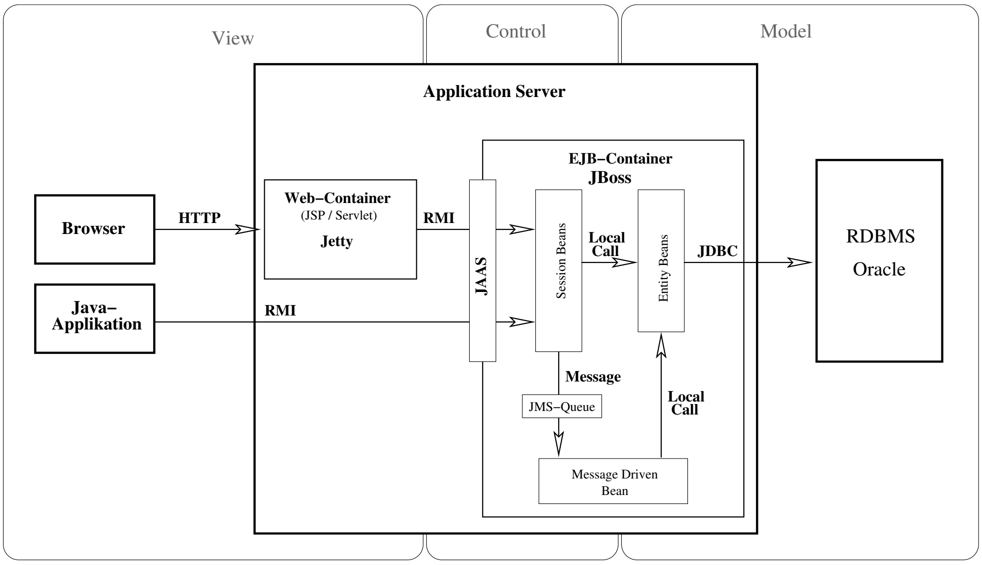

Abbildung 2: Technische Architektur

Während der Entwicklung diente JBoss als Applikationsserver für die Enterprise Java Beans. Dieser wird mit dem JSP- und Servlet-Server Jetty geliefert, der für die Generierung der dynamischen HTML-Seiten genutzt wurde. Als Datenbank-Management-Sytem wurde Oracle genutzt.

Komponenten

Der Marktplatz für Komponenten wurde komponentenbasiert entwickelt. Unter den gegebenen Anforderungen wurden fünf Hauptkomponenten identifiziert, aus denen der Marktplatz besteht: Benutzermanagement, Komponentenmanagement, Kategorisierung, Spezifikation und Bezahlung / Warenkorb. Diese Komponenten werden auch selbst im Marktplatz angeboten.

Die Komponente Benutzermanagement dient zur Verwaltung der beim Marktplatz registrierten Benutzer. Mit Hilfe von JAAS (`http://java.sun.com/products/jaas/`) wurde dabei ein Rollensystem eingeführt, das zwischen Gästen, Käufern, Anbietern und Administratoren unterscheidet. Die Komponente Komponenten-Management ist für die Haltung der Daten von angebotenen Komponenten zuständig und vermerkt, welchem Besitzer die Komponente gehört. Die Komponente Kategorisierung realisiert das Ordnungssystem des Marktplatzes, in dem Komponenten Kategorien zugeordnet werden. Mögliche Kategorien sind z.B. Anwendungsdomänen. Die Komponente Spezifikation setzt den Spezifikationsrahmen des Marktplatzes um. Dieser ist in Anlehnnung an das Pendant der Gesellschaft für Informatik [3] entwickelt worden. Die Komponente Bezahlung / Warenkorb verwaltet von einem Benutzer ausgewählte Komponenten in einem Warenkorb und ermöglicht prototypenhaft die Bestellung und Bezahlung dieser Komponenten. Die Module zur Bezahlung sind als „Message Driven Beans“ implementiert, um zur Laufzeit weitere Module hinzufügen zu können. Diese Module melden sich beim Marktplatz an und können dann vom Benutzer zur Abrechnung ausgewählt werden.

Entwicklung

Zur Durchführung der Projekts wurde der „Rational Unified Process“ [1] als Prozessmodell gewählt und um einige Techniken des „Extreme Programming“ [7] erweitert. So haben wir z.B.

bei der Entwicklung der eigentlichen Komponenten Pair-Programming praktiziert und testgetrieben gearbeitet. Das Testen erfolgte mit dem Framework JUnit (`http://www.junit.org/`), das sich sehr gut in die verwendete Entwicklungsumgebung IDEA (`http://www.intellij.com/idea/`) integrieren ließ. Für die Analyse und Modellierung des Systems wurde die „Unified Modelling Language" (`http://www.uml.org/`) genutzt, deren Diagramme mit Together ControlCenter (`http://www.togethersoft.com/`) umgesetzt wurden. Der komplexe Build-Prozess wurde durch Ant vom Apache-Projekt (`http://jakarta.apache.org/ant/`) unterstützt. Das Versionsmanagement und der geordnete Zugriff auf gemeinsame Daten wurde durch das „Concurrent Version System" (`http://www.cvshome.org/`) geregelt.

Erfahrungen

Mit der Nutzung des EJB-2.0 Standards konnte innerhalb des Back-Ends mit Local-Interfaces statt mit Remote-Interfaces gearbeitet werden. Diese Vermeidung von „Remote Method Invocations" (RMI) brachte erhebliche Geschwindigkeitssteigerungen und ermöglichte ein „information hiding" der Entitäten, da nur die Session Beans mit Remote-Interfaces versehen wurden. Durch die Nutzung von CMP und CMR konnte auf eine manuelle Abbildung des in der UML spezifizierten Datenbankmodells auf Datenbankrelationen verzichtet werden.

Die Nutzung von Struts bot gegenüber reinen JSP's den Vorteil der Entwicklung nach MVC-Kriterien auch innerhalb des Web-Containers. Dies sichert eine bessere Wartbarkeit und Lesbarkeit des Quellcodes. Erste Versionen der graphischen Benutzungsoberfläche, die nicht mit Struts realisiert worden waren, waren meist schwer zu verstehen und recht unübersichtlich.

Die Wahl von JAAS für die Umsetzung der Sicherheitsanforderungen zeigte die Vorteile von J2EE, da damit sehr schnell eine eigene flexible Benutzerverwaltung realisiert werden konnte. Ein erster Ansatz, der nicht auf JAAS basierte, stellte sich schnell als zu kompliziert heraus.

Durch die Verwendung von JUnit-Tests konnte während des gesamten Entwicklungsprozesses die Funktionalität der entwickelten EJB's ständig verifiziert werden. Eine vollständig testgetriebene Entwicklung konnte sich allerdings nicht durchsetzen. Insbesondere für automatisierte Tests der Benutzungsoberfläche schien das JUnit-Framework nicht geeignet.

Um performante und schnell zu implementierende Suchfunktionalitäten umzusetzen, musste der Applikationsserver JBoss so modifiziert werden, dass er EQL-Statements nach dem zukünftigen EJB-Standard 2.1 akzeptiert.

Literatur

[1] Grady Booch, Ivar Jacobsen, and James Rumbaugh. *The Unified Software Development Process.* Addison-Wesley, 1999.

[2] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns.* John Wiley & Son Ltd, 1. edition, 1996.

[3] Gesellschaft für Informatik, Arbeitskreis 5.10.3. Komponentenorientierte betriebliche Anwendungssysteme. http://www.fachkomponenten.de/. Stand: 1. Dezember 2002.

[4] Wilhelm Hasselbring. Component-based software engineering. In S. K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering*, volume 2, pages 289–305. World Scientific Publishing, 2002.

[5] Sun Microsystems, Inc. Java2 Plattform, Enterprise Edition J2EE. http://java.sun.com/j2ee/. Stand 1. Dezember 2002.

[6] Clemens Szyperski. *Component Software - Beyond Object-Oriented Programming.* Addison-Wesley, 1. edition, 1998.

[7] Don Wells. Extreme Programming: A Gentle Introduction. http://www.extremeprogramming.org/. Stand: 1. Dezember 2002.