

Vergleich eines Objektrelationalen Mapping mit EJB 3.0 gegenüber dem Einsatz einer Objektdatenbank

Iris Herke, Gero Wedemann

Fachbereich Elektrotechnik und Informatik
Fachhochschule Stralsund
Schwedenschanze 1
18439 Stralsund
Iris.Herke@web.de
Gero.Wedemann@fh-stralsund.de

Abstract: Objektrelationales Mapping verspricht einfachen Zugriff von Objektorientierten Systemen auf relationale Datenbanken. Die technische Umsetzung des Mapping gestaltet sich in der Praxis jedoch sehr aufwändig. Deshalb wurden verschiedene Frameworks geschaffen, die dem Entwickler diese Arbeit erleichtern sollen. Dazu gehört auch der neue EJB 3.0 Ansatz. Als Alternative bietet sich der Einsatz einer Objektdatenbank an, bei dem ein Objektrelationales Mapping überflüssig wird. Um die Vor- und Nachteile beider Strategien zu untersuchen, wurden die Alternativen empirisch untersucht. Dazu wird in diesem Beitrag die Implementierung beider Persistenzstrategien in einer Webapplikation beispielhaft umgesetzt. Die Rückwirkungen der jeweiligen Persistenzstrategie auf die Architektur der Anwendung werden analysiert. Außerdem erfolgt eine Untersuchung des entstandenen Quellcodes hinsichtlich des Softwarequalitätsmerkmals Wartbarkeit mittels Metriken und eines Experiments.

1 Einleitung

Bei der Entwicklung von Software wird zunehmend der Weg der objektorientierten Vorgehensweise eingeschlagen, für die Persistierung der Daten hingegen werden überwiegend relationale Datenbanken verwendet. Der dabei entstehende Strukturbruch zwischen Programmiersprache und Datenbanksystem wird als impedance mismatch bezeichnet. Konkret bedeutet dies, dass komplexe Objekte auf flache Tabellenstrukturen abgebildet werden müssen. Die Abbildung der Objektstrukturen auf Relationen erfolgt durch ein Mapping, welches festlegt, wie die Basiskonstrukte objektorientierter Sprachen wie Klassenbeziehungen, Polymorphie und Vererbung auf relationale Tabellen abgebildet werden. Die Implementierung dieses Mapping ist ein wesentlicher Kostenfaktor, in etwa ein Drittel des Programmieraufwandes [Fm02]. Aus diesem Grunde werden immer neue Werkzeuge und Frameworks erschaffen, die dem Entwickler diese Aufgabe erleichtern sollen. Zu diesen zählt auch der neue EJB 3.0 Standard.

Eine Möglichkeit diesen Strukturbruch zu umgehen, ist der Einsatz einer Objektdatenbank. Mit dem Einsatz einer solchen Datenbank wird der Vorteil der durchgehenden objektorientierten Entwicklung genutzt. Beide Ansätze versprechen eine Beschleunigung der Entwicklung sowie eine bessere Wartbarkeit des Produkts. Im Rahmen eines eigenständigen Forschungsprojektes von etwas neun Monaten an der FH Stralsund werden in der Master-Arbeit von Iris Herke die Einflüsse dieser beiden Persistenzstrategien hinsichtlich der Wartbarkeit und Komplexität des entstehenden Quellcodes mittels Metriken untersucht. Gemessen wird neben dem Umfang auch die Analysierbarkeit und Fehlerträchtigkeit des Quellcodes. Weiterhin wird ein Experiment durchgeführt, indem eine Änderung der Anforderung definiert wird, anschließend wird der entstehende Aufwand für beide Lösungen gemessen.

2 Die verwendete Beispielapplikation

Für die beispielhafte Umsetzung der beiden Persistenzlösungen wurde eine reale Webapplikation verwendet, die von Studenten im Fach Softwareprojektorganisation bei Gero Wedemann entwickelt worden war. Die Anwendung wurde in der Programmiersprache Java erstellt und umfasst etwa 180 Klassen und 70 JSPs. Der Applikation liegt eine typische Mehrschichtenarchitektur zugrunde. Die Präsentationsschicht verwaltet die Sessions und interagiert mit dem Benutzer, die Domänenschicht enthält die eigentliche Geschäftslogik und die unterste Schicht stellt die Datenbank da [Fm03]. Die Trennung der Darstellung und Verarbeitung der Daten wurde gemäß dem Model-View-Controller [Bf96] Prinzip realisiert. Dazu wurde das OpenSource-Framework Struts [STR] der Apache Software Foundation verwendet. Für die Organisation der Geschäftslogik wurde das Muster Transaction Script [Kap.2 Fm02] verwendet. Der Zugriff auf die Datenbank erfolgt in Gatewayklassen, für jede Klasse im Domainmodell existiert eine Gatewayklasse. Vorteil dieser Lösung ist die Kapselung der SQL-Statements. Die ursprüngliche Applikation griff über die JDBC Schnittstelle auf eine relationale Datenbank zu.

2.1 Die Objektdatenbank

Für unsere Untersuchung haben wir aus dem großen Angebot der Objektdatenbankhersteller db4o ausgewählt, da es sich um ein OpenSource-Produkt handelt. Weiterhin verspricht db4o eine äußerst einfache Handhabung, da db4o weder eine Object Definition Language noch eine Object Query Language verwendet. Deshalb genügt db4o nicht dem ODMG – Standard [ODM] besitzt aber nahezu alle Eigenschaften die im Grundsatzpapier [Am89] für OODB festgeschrieben wurden. Db4o gibt es derzeit Datenbank für Java und .NET. Bei db4o handelt es sich um eine native Datenbank, d.h. es muss kein Datenbankschema entwickelt, sondern das Domainmodell dient als Datenbankschema [Pj06]. Die Manipulation und Anfragen der Objekte erfolgt in der Programmiersprache der Anwendung. Um ein Objekt zu speichern genügt eine Zeile Code: `db.set (object);` Zwischen Einfügen und Ändern wird nicht unterschieden, um ein Objekt zu modifizieren, wird es der Datenbank entnommen, die Änderung vorgenommen und wieder in die Datenbank geschrieben.

Grundsätzlich gibt es zwei Möglichkeiten wie die Datenbank aus architektonischer Sicht angebunden werden kann: Zum einen könnten die Datenbankaufrufe direkt in der Geschäftslogik implementiert werden. Dies wäre problemlos möglich da mit db4o keine datenbankspezifischen Sprachkonstrukte wie SQL-Statements benötigt werden. Alternativ könnten die Datenbankoperationen weiterhin in speziellen Klassen gekapselt werden. Um die Trennung der Geschäftslogik und der Datenbankzugriffe weiterhin zu gewährleisten, wurde die Integrationsschicht mit den Gatewayklassen erhalten und die Datenbankzugriffe dort gekapselt.

2.2 Datenbankanbindung mit EJB 3.0

Als Enterprise Java Beans [EJB] werden standardisierte Komponenten innerhalb eines J2EE Containers bezeichnet. Die Verwendung von EJBs unterstützt die Trennung von fachlichen und technischen Aspekten. Idealerweise soll sich der Entwickler nur noch der Implementierung der Geschäftslogik widmen, technische Aufgaben übernehmen die EJBs. Unterstützung erhält der Entwickler in Sachen Sicherheit, Skalierbarkeit, Transaktionen und Persistenz.

Nachdem die Entity Beans des EJB 2.0 Standard von vielen Praktikern als ungeeignet kritisiert wurden, sind im neuen Standard 3.0 Java Beans wieder POJOs, einfache Java Klassen, deren Verhalten mittels Annotationen deklariert wird. Die neue Spezifikation verspricht unter anderem dem Entwickler Unterstützung in Bezug auf das OR-Mapping.

Für dieselbe Webapplikation erfolgte eine Anbindung der relationalen Datenbank mit EJB 3.0 durch Anja Bethge im Rahmen einer Bachelor-Arbeit [Ba07]. Der entstandene Quellcode und die architektonische Anbindung werden mit der zu erst erläuterten Persistenzstrategie verglichen.

3 Zusammenfassung der Ergebnisse

Die Implementierung der beiden Persistenzlösungen ist abgeschlossen. Bei der Objektdatenbank ist die Speicherung komplexer Objekte sehr einfach. Das Auffinden von Objekten kann nur über vorher definierte Fachschlüssel erfolgen. Die Navigation innerhalb der Datenbank ist nur in Richtungen möglich, die bereits im Domainmodel festgelegt wurden. Das Löschen eines Objektes erfordert vom Entwickler eine erhöhte Aufmerksamkeit, da die Referenzen auf das zu löschende Objekt nicht selbstständig von der Datenbank gelöscht werden.

Bei EJB 3.0 Entity Beans erzwang die Implementierung der Persistenz eine Änderung der Architektur im Bezug auf den Zugriff auf die Datenbank. Ursprünglich waren die SQL-Statements ausschließlich in den Gatewayklassen gekapselt, diese agieren in dieser Persistenzlösung als Sessionbeans. Die Modellklassen wurden zu Entity Beans und enthalten JPQL-Statements. Die Entity-Beans sind in diesem Fall Active Records [Fm03].

Im Vergleich zeigt sich, dass beide Lösungen den Entwickler bei der Speicherung komplexer Objekte unterstützen. Die Objektdatenbank benötigt kein Datenbankschema. Die Objekte können so wie sie in der Applikation existieren persistiert werden. Bei der Lösung mit Entity Beans erzeugt der Persistenzmanager des EJB Containers das Datenbankschema über das gekoppelten Persistenzframework.

Bei EJB3 werden Datenbank-Anfragen in der Java Persistenz Query Language (JPQL) formuliert. Bei db4o hingegen können die Abfragen direkt in der Programmiersprache der Applikation formuliert werden. Dies hat den Vorteil, dass der entstehende Code vom Compiler geprüft wird und der Entwickler seine bevorzugte Sprache verwenden kann. Als weiterer Vorteil der Verwendung einer Objektdatenbank muss kein zusätzlicher Code geschrieben werden. Die Lösung mittels Entity Beans hingegen erzwingt die Implementierung zusätzlicher Interfaces.

Derzeit wird der entstandenen Quellcode der beiden Lösungen hinsichtlich Wartbarkeit und Komplexität zu untersuchen. Dafür werden verschiedene Softwaremetriken eingesetzt. Weiterhin soll ein Experiment durchgeführt werden: Es wird ein Anforderungsänderung definiert und in beiden Lösungen umgesetzt. Gemessen wird der Aufwand der in beiden Fällen entsteht. Die Ergebnisse dieser Untersuchungen werden auf der Konferenz vorgestellt.

Literaturverzeichnis

- [Am89] Atkinson, M. et al.: The Object-Oriented Database System Manifesto. Proc. 1st International Conference on Deductive and Object-Oriented Databases, Kyoto, 1989
- [Ba07] Bethge, A.; Erprobung der Entity Beans des EJB 3.0 Standard zur Einbindung in eine relationale Datenbank am Beispiel einer existierenden Webapplikation, Bachelorthesis, FH Stralsund, 2007
- [BF96] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M.: Pattern-Oriented Software Architecture. A System of Patterns, John Wiley & Sons, 1996
- [DB40] <http://www.db4o.com>
- [EJB] <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>
- [Fm02] Fowler, M.: Patterns of Enterprise Application-Architecture, Addison Wesley, 2002
- [Hi07] Herke, I.; Untersuchung der Eignung und Qualität einer Objektdatenbank im Vergleich mit EJB 3.0 anhand einer existierenden Webapplikation: Messen des Aufwandes, Komplexität und Wartbarkeit mittels Metriken, Masterthesis, FH Stralsund, 2007 (in Vorbereitung)
- [Pj06] Paterson, J.; Edlich, S.; Hörning, H.; Hörning, R.: The Definitive Guide to db4o, Springer Verlag, New York, 2006
- [STR] <http://struts.apache.org/>