Fachhochschule Brandenburg
Fachbereich Informatik und Medien

Analyse und praktischer Vergleich von neuen Access-Layer-Technologien in modernen Web-Anwendungen unter Java

Diplomarbeit, vorgelegt von Oliver Kalz

Aufgabenstellung:

Das Internet wird immer häufiger als Plattform für vollständige, verteilte Anwendungen auf der Basis von Java (J2EE, Servlets, JSPs) genutzt.
Derartige Web-Anwendungen beziehen ihre Daten meist aus Datenbanken, womit dem Access Layer als unterste Schicht einer Anwendung eine große Bedeutung zukommt. Der Access Layer ist dafür verantwortlich, die zu verarbeitenden und anzuzeigenden Daten zuverlässig, vor allem aber performant zur Vefügung zu stellen.

Im Verlauf der Arbeit sollen alternative Persistenzmechanismen an praktischen Code-Beispielen vorgestellt und in Hinblick auf ihr Handling und ihre Performance verglichen werden.

Der Begriff der Objektpersistenz:

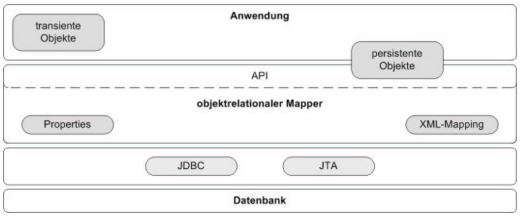
Objekte, die in einer Anwendung erzeugt und benutzt werden, sollen eine beliebige Lebensdauer haben. Der Begriff Objektpersistenz beschreibt den Mechanismus, Objekte über die Laufzeit einer Anwendung hinaus zu erhalten. Solche Objekte werden persistente Objekte genannt und sind solange verfügbar, bis sie explizit gelöscht werden. Um die Verfügbarkeit zu gewährleisten, werden persistente Objekte in den Sekundärspeicher ausgelagert. So gehen diese Daten auch nicht bei einem Absturz der Anwendung verloren.
Für die Auslagerung gibt es verschiedene Möglichkeiten. Objekte lassen sich zum Beispiel als einfache Dateien im Dateisystem ablegen. Java bietet zudem die Serialisierung an. Beide Möglichkeiten sind nicht für große Datenmengen geeignet, da zum Beispiel eine effektive Suche nach Objekten nicht stattfinden kann.
Für die Speicherung umfangreicher Daten werden meist relationale Datenbanken eingesetzt. Sie organisieren die Daten in Tabellen und bieten mit SQL eine standardisierte Anfragesprache. Bei Verwendung dieser Systeme aus Java und anderen Sprachen tritt jedoch der Impedance Mismatch auf: Es treffen zwei unterschiedliche Programmierparadigmen aufeinander. Als Lösung begann Mitte der 1980er Jahre die Entwicklung der objektorientierten Datenbanken. Sie integrieren ein objektorientiertes Datenmodell und vermeiden so den Impedance Mismatch. Allerdings sind diese Datenbanken noch nicht sehr weit entwickelt und es fehlt an Standards, die von den wenigen verfügbaren Systemen eingehalten werden sollten.

Technologien und Werkzeuge:

Eine der ersten Technologien, die objektrelationales Mapping anwendeten, waren die Enterprise JavaBeans (EJBs). Entity Beans können völlig unabhängig von Applikationsserver und Datenbank entwickelt werden. Bei CMP Entity Beans übernimmt der Applikationsserver alle Details der Persistenz. Da es vergleichsweise aufwendig ist, Entity Beans zu entwickelt, wurde 2002 die Java Data Objects von Sun entwickelt. Sie haben das Ziel, die Entity Beans in Zukunft abzulösen.
Neben diesen beiden Technologien existieren eine Reihe weiterer Tools, die mit jeweils eigenen Ansätzen ein objektrelationales Mapping realisieren. Zu diesen Tools gehören beispielsweise der Open-Source-Mapper Hibernate sowie das kommerzielle Tool TopLink von Oracle. Diese Tools unterscheiden sich hauptsächlich in Einarbeitungsaufwand und Bedienungskomfort (Mapping GUI, API-Umfang).

Objektrelationales Mapping:

Das objektrelationale Mapping beschreibt die Abbildung der Daten eines Objekts auf relationale Strukturen. Im Normalfall wird eine Klasse auf eine Tabelle abgebildet, die Attribute auf Tabellenspalten. Somit wird ein Objekt über das Mapping mit seiner persistenten Repräsentation in der Datenbank verbunden.
Werkzeuge, die diese Abbildung realisieren, werden O/R-Mapper genannt. Sie stellen eine Schicht von Java-Klassen zur Verfügung, die zwischen einer Java-Applikation und einer relationalen Datenbank zum Einsatz kommt. Die Schicht agiert als objektorientierter Wrapper um die Datenbank und ermöglicht einer Applikation, ihre persistenten Objekte auf ein Datenbank-Schema abzubilden.
Durch eine solche Persistenzschicht wird die Applikation vor Änderungen in der Datenbank geschützt, wodurch Änderungen am Datenbank-Schema einen geringeren Einfluß auf die Applikation haben.
Die Menge der O/R-Mapper läßt sich in zwei Gruppen einteilen: in Schemageneratoren, die ausgehend von vorhandenen Java-Klassen ein Datenbankschema erzeugen, und Codegeneratoren, die aus einem vorhandenen Datenbankschema eine komplette Persistenzschicht erzeugen.



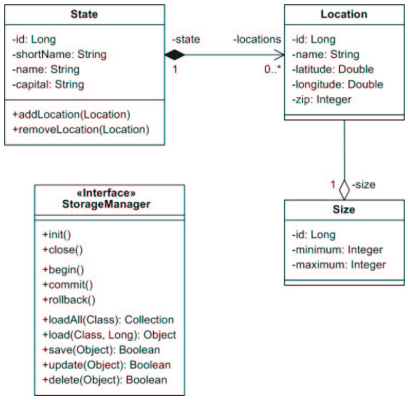
allgemeine Architektur eines O/R-Mappers

Fazit:

Auf dem Markt herrscht eine große Vielfalt an Möglichkeiten, mit denen die Persistenzschicht einer Web-Anwendung realisiert werden kann.
Bei den vorgestellten Werkzeugen entscheiden letztendlich das Projektumfeld, der Funktionsumfang sowie die anfallenden Kosten über den Einsatz in einer Produktivumgebung.

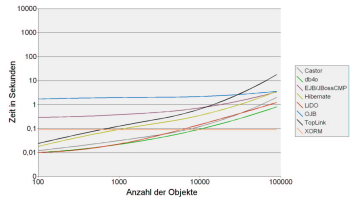
Da sich die objektorientierten Datenbanken auf absehbare Zeit nicht gegen die etablierten relationalen Systeme durchsetzen werden, wird das objektrelationale Mapping auf lange Sicht die einzige und bevorzugte Technik bleiben, mit persistenten Objekten unter Verlagerung des Impedance Mismatch zu arbeiten.

Die Beispielanwendung:



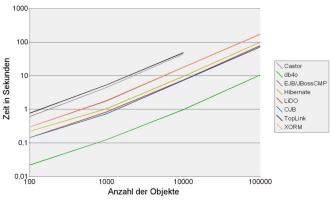
Die Beispielanwendung modelliert eine Ortsdatenbank für Deutschland mit Klassen für Bundesland, Ort und Ortsgröße. Die Grafik zeigt die UML-Darstellung der Klasse ohne die obligatorschen Konstruktoren und get-/set-Methoden.
Das Interface StorageManager bietet alle notwendigen Methoden der Persistenzschicht, um Objekte zu laden, zu speichern und zu löschen. Für die Transaktionskontrolle stehen ebenfalls Methoden zur Verfügung. Die Anwendung enthält sechs Implementierungen dieses Interfaces, jeweils mit einem der in der Arbeit vorgestellten Tools. Die Geschäftslogik greift über diesen StorageManager auf die persistenten Objekte zu.

Performance:



Performance ausgewählter Werkzeuge beim Lazy Loading von Objekten (logarithmische Skala)

Untersucht wurde die Performance von insgesamt acht der in der Arbeit näher vorgestellten Tools. Verglichen wurden das Laden sowie das Speichern von Objekten.
Das Laden von Objekten wird von den Tools durch Einsatz des Lazy Loadings optimiert. Dies bedeutet, daß ein Objekt nicht sofort vollständig geladen wird, sondern nur seine ID oder seine primitiven Member. Insbesondere referenzierte Objekte werden beim ersten Zugriff nachgeladen. Das Laden mit Lazy Loading geschieht bei allen Testkandidaten mit linearer Komplexität. Erst beim vollständigen Laden zeigen sich deutliche Unterschiede. Einige Tools verschlechtern sich auf quadratische Komplexität.
Das Speichern von Objekten wird von allen Testkandidaten mit linearer Komplexität erledigt.



Performance ausgewählter Werkzeuge beim Speichern von Objekten (logarithmische Skala). Die lineare Komplexität wird deutlich .

Betreuer: Prof. Dr. S. Edlich
Prof. Dr. Th. Preuß

Abgabe: 08.01.2004