Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ HIBERNATE

КУРСОВАЯ РАБОТА

Студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Левенец Сергея Алексеевича

Научный руководитель
доцент, к. ф.-м. н., доцент _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2016

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Hibernate – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения

JDBC – Java DataBase Connectivity, соединение с базами данных на Java

БД – База данных

СУБД – Система управления базами данных

MySQL – свободная реляционная система управления базами данных

PostgreSQL – объектно-реляционная система управления базами данных

Маппинг – процесс составления схемы того, какими данными следует обмениваться, как они будут использоваться и каким системам управления они нужны

JNDI – Java API, организованный в виде службы каталогов, который позволяет Java-клиентам открывать и просматривать данные и объекты по их именам

SQL – формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных, управляемой соответствующей системой управления базами данных

JAR – Java-архив. Представляет собой ZIP-архив, в котором содержится часть программы на языке Java.

ВВЕДЕНИЕ

Сегодняшний день уже нельзя представить без интернета.Вся наша жизнь в той или иной степени связана с всемирной паутиной, окутавшая практически каждый дом. Основным элементом интернета является информация. Товар в интернет - магазине, научная статья, любимая композиция - все это информация, которая должна храниться, обрабатываться и выдаваться пользователем при необходимости. Поэтому, взаимодействие веб-приложения с базой данных является ключевым моментом. Для решения этих задач постоянно разрабатываются различные подходы: разрабатываются новые языки программирования, новые системы управления базами данных, различные библиотеки, позволяющие автоматизировать какую либо часть разработки программного обеспечения, различные надстройки. В данной работе будет рассмотрена библиотека Hibernate, которая взаимодействует с базой данных, позволяющая перевести реляционную модель в объектно-ориентиованную, что дает большую гибкость при разработке. Основная цель работы - определить, насколько эффективно использовать данную библиотеку при разработке веб-приложения. Так же,будет произведено сравнение в производительности по основным операциям к базе данных: выборка, вставка, удаление между рассматриваемой библиотекой и стандартным механизмом доступа к БД - JDBC.

1 Библиотека Hibernate

Hibernate – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного проецирования. Она представляет собой свободное программное обеспечение с открытым исходным кодом (open source), распространяемое на условиях GNU Lesser General Public License. Данная библиотека предоставляет лёгкий в использовании каркас (фреймворк) для отображения объектно-ориентированной модели данных в традиционные реляционные базы данных.

Сопоставление Java-классов с таблицами БД осуществляется с помощью конфигурационных XML файлов, либо с помощью Java аннотаций. Обеспечиваются возможности по организации отношения между классами один-ко-многим и многие-ко-многим. В дополнение к управлению ассоциации между объектами, Hibernate может также управлять рефлексивными отношениями, где объект имеет связь один-ко-многим с другими экземплярами своего типа. [1]

1.1 Работа с Hibernate

Для работы с библиотекой Hibernate необходимо скопировать JDBC драйвер используемой базы данных в global classpath. Hibernate использует JDBC соединения (JDBC connections) для вызова SQL запросов к базе данных, поэтому необходимо предоставить настроенный пул JDBC соединений (JDBC connection pool), либо настроить Hibernate для использования одного поддерживаемых пулов встроенных в Hibernate. Hibernate упакован как JAR-библиотека. Файл hibernate2.jar должен быть скопирован в context classpath вместе с остальными классами приложения.

Теперь необходимо настроить общий маппинг для Hibernate и сервера, на котором реализуется приложение пул соединений с базой данных (shared database connection pool). Сервер привязывает (binds) пул соединений к JNDI, для этого необходимо добавить объявление ресурса (resource declaration) в главный конфигурационный файл.Hibernate запрашивает данные соединения через интерфейс JNDI.

Следующим шагом необходимо настроить сам Hibernate через XML-файл. Основной способ настройки, через файл Java-свойств (properties), эквивалентен, но не предоставляет дополнительных расширенных опций. Кон-

фигурационный XML-файл должен быть размещен в `context classpath` (WEB-INF/classes), под именем `hibernate.cfg.xml`. [2]

1.2 Долгоживущий класс

Наилучшая для Hibernate модель данных долгоживущих классов – это Plain Old Java Objects (POJOs, иногда также называемая Plain Ordinary Java Objects). POJO очень похож на JavaBean со свойствами (properties), доступными через `getter` и `setter` методы, которые прячут внутреннее представление класса от публичного доступа [3]

Каждый долгоживущий `(persistent)` класс должен иметь идентификатор (на самом деле, только классы представляющие сущности `(entities)`; зависимые объекты, компоненты `(components)` могут не иметь идентификатора). Идентификатор используется для того чтобы различать долгоживущие объекты: два объекта `A` и `B` идентичны если является истинным выражение:

```
catA.getId().equals(catB.getId())
```

Данная концепция называется идентичность средствами базы данных `(database identity)`. Hibernate поставляется с несколькими генераторами идентификаторов. [3]

1.3 Основы объектно-реляционного маппинга

Объектно - реляционному маппингу уделяется особое внимание, так как от него будет зависит эффективность приложения. При разработке программного обеспечения этому пункту следует уделить как можно большее внимание. При грамотном подходе к этому процессу можно выиграть в производительности, а сам программный код будет более читаемым. Маппинг может служить мощным инструментом проектирования базы данных, так как благодаря утилите Hibernate SchemaExport можно сгенерировать DDL скрипт. Сгенерированная схема будет включать в себя ограничения ссылочной целостности `(referential integrity constraints)`, основные и внешние ключи для сущностей и таблиц коллекций. Данная утилита также создает таблицы для последовательностей(sequences) и спроецированных id-генераторов `(identity generators)`. [4]

1.3.1 Объявление маппинга

Объектно-реляционный маппинг описывается в виде XML документа. Данный документ легко читается и интуитивно понятен при ручном редактирование. Язык описания такого XML - а ориентирован на Java, это означает что маппинг конструируются вокруг объявлений долгоживущих java-классов, а не таблиц БД.

Маппинг можно создавать и редактировать вручную, а можно воспользоваться дополнительным программным обеспечением для генерации документов описывающих маппинги. Например: XDoclet, Middlegen и AndroMDA.

1.3.2 Many-to-one

Обычная связь с другим долгоживущим классом объявляется, используя элемент `many-to-one`. В реляционных терминах это ассоциация многих к одному. В действительности это просто ссылка на объект. [5]

1.3.3 One-to-one

Ассоциация "один к одному"с другим долгоживущим классом можно объявить, используя элемент `one-to-one`.

Для ассоциации по первичному ключу нужно добавить следующее отображение для классов `Employee` и `Person` соответственно.

```
<one-to-one name="person" class="Person"/>
<one-to-one name="employee" class="Employee" constrained="true"/>
```

Как альтернативный вариант описания связи "один к одному"от `Employee` к `Person`, через уникальный внешний ключ можно использовать следующую запись:

```
<many-to-one name="person" class="Person" column="PERSON_ID" unique="true"/>
```

Эта ассоциация может быть двунаправленной после добавления следующего выражения к маппингу класса `Person`:

```
<one-to-one name"employee" class="Employee" property-ref="person"/>
```

1.3.4 Dynamic-component

Элемент <component> отображает поля вложенного объекта на колонки таблицы родительского класса. Компоненты могут в свою очередь определять свои собственные свойства, компоненты или коллекции. [5]

1.4 Hibernate cache

Hibernate cache это 3 уровня кэширования:

- Кеш первого уровня (First-level cache);
- Кеш второго уровня (Second-level cache);
- Кеш запросов (Query cache). [6]

Кэш первого уровня всегда привязан к объекту сессии. Hibernate всегда по умолчанию использует этот кэш и его нельзя отключить.

Если кэш первого уровня привязан к объекту сессии, то кэш второго уровня привязан к объекту-фабрике сессий (Session Factory object). Что как бы подразумевает, что видимость этого кэша гораздо шире первого уровня. [2]

2 Java DataBase Connectivity

JDBC (англ. Java DataBase Connectivity — соединение с базами данных на Java) — платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета `java.sql`, входящего в состав Java SE.

JDBC основан на концепции так называемых драйверов, позволяющих получать соединение с базой данных по специально описанному URL. Драйверы могут загружаться динамически (во время работы программы). Загрузившись, драйвер сам регистрирует себя и вызывается автоматически, когда программа требует URL, содержащий протокол, за который драйвер отвечает. [1]

2.1 Интерфейсы

JDBC API содержит два основных типа интерфейсов: первый — для разработчиков приложений и второй (более низкого уровня) для разработчиков драйверов.

Соединение с базой данных описывается классом, реализующим интерфейс `java.sql.Connection`. Имея соединение с базой данных, можно создавать объекты типа `Statement`, служащие для исполнения запросов к базе данных на языке SQL.

Существуют следующие виды типов Statement, различающихся по назначению:

`java.sql.Statement` - Statement общего назначения;

`java.sql.PreparedStatement` - Statement, служащий для выполнения запросов, содержащих подставляемые параметры (обозначаются символом '?' в теле запроса);

`java.sql.CallableStatement` - Statement, предназначенный для вызова хранимых процедур.

Интерфейс `java.sql.ResultSet` позволяет легко обрабатывать результаты запроса. [3]

3 Практическая часть

3.1 Анализ производительности

Для анализа производительности были созданы три таблицы: `hibernate`, `jdbc`, `cashHibernate`. Одна для манипулирования данными при помощи библиотеки Hibernate, вторая, для манипуляции данными с помощью стандартного механизма взаимодействия между языком программирования Java и базами данных JDBC, третья для Hibernate cash.

3.2 Операция INSERT

Поочередно, во все 3 таблицы произведем вставку сгенерированной строки из 200 символов. Эксперимент проведен для 100, 1000, 10000, 100000, 1000000 вставок. Результат работы занесены в таблицу 1.

Таблица 1 – Результат выполнения запросов

количество	Hibernate, м/с	Hibernate cache, м/с	JDBC, м/с
100	1158	1162	4004
1000	1712	1163	45404
10000	4928	4449	471864
100000	27793	26242	4661352
1000000	245337	223683	45216387

3.3 Операции SELECT и DELETE

Была произведена выборка всех записей из трех таблиц

На следующем шаге операция удаления из трех таблиц: Hibernate, Hibernate Cashe и JDBC.

Результат работы занесены в таблицу 2.

Таблица 2 – Результаты для операций SELECT и DELETE

Операция	Hibernate, м/с	Hibernate cache, м/с	JDBC, м/с
SELECT	18166	2426	10682
DELETE	55450	5163	55300

3.4 Разработка веб-приложения

Для разработки веб-приложения были использованы следующие технологии: Язык программирования: `Java EE` База данных: `MySql` Веб-контейнер: `Glasfish` Библиотеки: `Hibernate`, `PrimeFaces` Возможности приложения: Регистрация и авторизация пользователя, добавление записей, просмотр записей

других пользователей, добавление пользователей в друзья, возможность добавления аватарок, поиск пользователей по сайту, мультиязычность, разделение ролей.

Возможность регистрации и авторизации, и разделение ролей реализовано на уровне сервера. Для переключение языков использована локализация, благодаря которой можно сопоставлять соответствие каждому слову на нужные языки. Для взаимодействия с базой данных использована библиотека Hibernate. Использован навигационный компонент Java, благодаря чему можно легко проектировать возможное поведение пользователей.

Заключающим этапом практической части является тестирование производительности веб-сайта. Для данной цели была написана вспомогательная программа, которая эмитировала активную деятельность пользователей. С помощью генератора случайных чисел генерировалось число от одного до пяти. Сгенерированное число является номером операции: 1 – удаление записи. 2 – вставка новой записи, 3 – выборка всех записей, 4 – обновление поля Title, 5 – выборка по критерию(критерий like и between). Всего будет произведено 10000000 операций в сумме.

Время выполнения 10000000 заняла менее двадцати минут. Это довольно таки не плохой показатель времени

ЗАКЛЮЧЕНИЕ

Подводя итоги стоит отметить эффективность использования библиотеки Hibernate в веб-приложениях. Как показала практика - это очень мощный и гибкий инструмент. Исследование производительности так же доказывает эффективность данной библиотеки. Hibernate - относительно новый подход для взаимодействия с базами данных, который еще не успел получить широкого распространения. При переходе к данной технологии компаниям придется выделять дополнительное финансирование для обучения своих сотрудников. Но в замен, компания, перешедшая на данную библиотеку получает в свое распоряжение очень мощный инструмент, позволяющий вести более эффективную разработку. К достоинствам данного компонента стоит отнести:

- Производительность;
- Независимость от СУБД;
- Таблицы базы данных представлены в виде объектов;
- Гибкость.

К недостаткам:

- Требуется дополнительные знания, что увеличивает стоимость программиста;
- Громоздкость кода при построении сложной выборки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Java SE Technologies - Database [Электронный ресурс]. — URL: http://www.oracle.com/technetwork/java/javase/jdbc/index.html (дата обращения: 16.05.2016).

2 Hibernate JavaDocs [Электронный ресурс]. — URL: http://docs.jboss.org/hibernate/orm/5.1/javadocs/ (дата обращения: 16.05.2016).

3 Processing SQL Statements with JDBC [Электронный ресурс]. — URL: https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html (дата обращения: 16.05.2016).

4 Basic O/R Mapping [Электронный ресурс]. — URL: https://docs.atlassian.com/hibernate2/2.1.8/reference/mapping.html (дата обращения: 16.05.2016).

5 Chapter 9;Manipulating Persistent Data [Электронный ресурс]. — URL: https://docs.atlassian.com/hibernate2/2.1.8/reference/manipulatingdata.html#manipulatingdata-updating (дата обращения: 16.05.2016).

6 Java EE — Hibernate Framework [Электронный ресурс]. — URL: http://www.linuxtopia.org/online_books/jboss_documentation/jboss_app_platform_4.3_hibernate_reference_guide/jboss_hibernate_HQL_The_Hibernate_Query_Language-The_group_by_clause.html (дата обращения: 16.05.2016).