МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные технологии»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ДЛЯ ВЫПОЛНЕНИЯ КОНТРОЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ» ДЛЯ СТУДЕНТОВ ЗАОЧНОЙ И ОЧНО-ЗАОЧНОЙ ФОРМ ОБУЧЕНИЯ

Ростов-на-Дону
ДГТУ
2023

УДК 372.8:004
Составители: М. В. Привалов

Методические указания для выполнения контрольной работы по дисциплине «Современные технологии программирования» для студентов заочной и очно-заочной форм обучения. - Ростов-на-Дону: Донской гос. техн. ун-т, 2023. - 13с.

Рассматриваются современные подходы и технологии разработки программного обеспечения для Web-ориентированных и распределённых информационных систем.

Предназначены для студентов направления 09.04.02 «Информационные технологии» заочной и очно-заочной форм обучения.

УДК 372.8:004

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск зав. кафедрой «Информационные технологии»,
д-р техн. наук, профессор Б.В. Соболь

В печать ___.___. 20___г.
Формат 60×84/16. Объем___усл.п.л.
Тираж ___экз. Заказ №___.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина,1

Оглавление

Применение Java Persistence API и ORM Hibernate для работы с РСУБД в приложениях на Java

Цель работы: получить навыки по работе с реляционными базами данных в Java. Научиться использовать отображение объектов на реляционную модель с использованием современных ORM-каркасов.

Основные теоретические сведения.

Hibernate — это популярный ORM-каркас, цель которого – связать ООП и реляционную базу данных. Работа с Hibernate сокращает время разработки проекта в сравнении с обычным JDBC, так как создаёт связь между таблицами в базе данных (далее – БД) и Java-классами и наоборот. Схематично это можно отразить так (рис. 1):

Рис. 1. –Hibernate и реляционная СУБД в Java-приложении

Преимущества использования Hibernate:

- Обеспечивает простой API для записи и получения Java-объектов в/из БД.
- Минимизирует доступ к БД, используя стратегии fetching.
- Не требует сервера приложения.
- Позволяет нам не работать с типами данных языка SQL, а иметь дело с привычными нам типами данных Java (POJO – Plain Old Java Objects).

- Заботится о создании связей между Java-классами и таблицами БД с помощью XML-файлов или аннотаций, не внося изменения в программный код. Если нам необходимо изменить БД, то достаточно лишь внести изменения в XML-файлы.

Hibernate поддерживает все популярные СУБД: MySQL, Oracle, PostgreSQL, Microsoft SQL Server Database, HSQL, DB2 и может работать в связке с такими технологиями, как Maven и J2EE, а также NoSQL СУБД.

Приложение, использующее Hibernate, очень упрощённо имеет следующую архитектуру (рис. 2.):

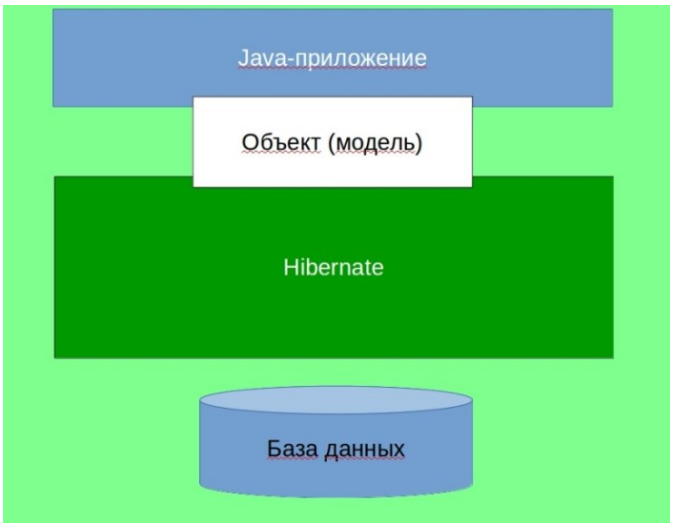

Рис. 2. – Упрощённая архитектура Java-приложения, использующего Hibernate

Если мы рассмотрим строение самого Hibernate более подробно, то этот же рисунок будет уже выглядеть следующим образом (рис. 3):

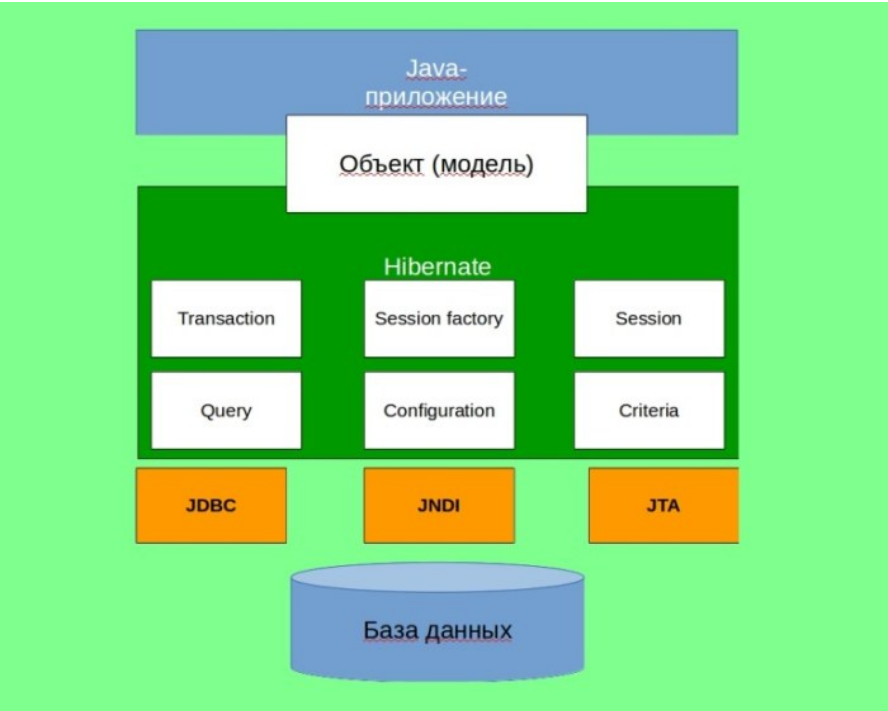

Рис. 3. – Детальная архитектура приложения с Hibernate

Hibernate поддерживает такие API, как JDBC, JNDI, JTA.

JDBC обеспечивает простейший уровень абстракции функциональности для реляционных БД. JTA и JNDI, в свою очередь, позволяют Hibernate использовать серверы приложений J2EE.

Transaction. Представляет собой рабочую единицу, изолирующую порцию работы с БД. В Hibernate транзакции обрабатываются менеджером транзакций.

SessionFactory. Самый важный и самый сложный и функционально насыщенный объект, который представляет собой фабрику. Обычно фабрика создаётся в единственном экземпляре, при запуске приложения. Как минимум одна SessionFactory требуется для каждой БД, при этом каждая из них конфигурируется отдельным конфигурационным файлом.

Session. Сессия используется для получения физического соединения с БД. Обычно сессия создаётся при необходимости, а после этого закрывается. Это возможно, потому что эти объекты крайне легковесны. В целом, можно сказать,

что создание, чтение, изменение и удаление объектов происходит через объект Session.

Query. Этот объект использует HQL или SQL для чтения/записи данных из/в БД. Экземпляр запроса используется для связывания параметров запроса, ограничения количества результатов, которые будут возвращены, и для выполнения запроса.

Configuration. Этот объект используется для создания объекта SessionFactory и конфигурирует сам Hibernate с помощью конифигурационного XML-файла, который объясняет, как обрабатывать объект Session.

Criteria. Используется для создания и выполнения объектно-ориентированных запросов для получения объектов. Критерии часто используются при реализации поиска информации в БД на основании пользовательских критериев.

Необходимые шаги по подключению Hibernate к проекту.

Для работы с Hibernate понадобится, как и в предыдущей работе, подключить необходимые библиотеки и выполнить некоторые настройки. Более детально:

1. Подключить в виде JAR-библиотеки драйвер соответствующей СУБД.
2. Подключить JAR-библиотеку с самим Hibernate (рекомендуется версия 5.x).
3. Выполнить конфигурацию ORM-каркаса в hibernate.cfg.xml.
4. Создать XML-конфигурации сущностей или аннотированные классы.

Для конфигурирования сущностей в настоящее время чаще применяют аннотации. Как сконфигурировать сущности, на которые отражаются таблицы в БД, можно изучить на ресурсе https://javarush.ru/groups/posts/hibernate-java или https://proselyte.net/tutorials/hibernate-tutorial/introduction/

Следует отметить, что первые два пункта требуется выполнять, используя системы и технологии управления зависимостями приложения (Maven, Gradle и им подобные). Как с ними работать, показано ниже.

Настройка, сборка проектов и управление зависимостями.

Все современные IDE позволяют управлять зависимостями проектов, собрать, запускать и даже развёртывать приложения. Однако, для универсальности, как правило, используют сторонние средства управления зависимостями и сборки проектов. Для Java приложений это Ant, Maven, Gradle.

Apache Maven — каркас для автоматизации управления зависимостями и сборки проектов, специфицированных на XML-языке POM(Project Object Model).

На примере бесплатно распространяемой популярной IDE Eclipse, создание проекта выполняется через меню File->New->Maven project. На первом экране настройки проекта требуется отметить Create simple project, чтобы не выполняль настройку архетипа проекта. После этого потребуется заполнить базовые поля Maven-проекта:

Group Id: наименование отдела или организации (правила аналогичны именованию пакетов)

Artifact Id: название проекта

Далее будет создан новый проект, в котором присутствует файл pom.xml, отвечающий за настройку разрешения зависимостей и сборку. Его содержимое может быть примерно таким (в зависимости от заполнения указанных выше полей):

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

```
  <groupId>org.dstu</groupId>
  <artifactId>dbdemo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

Структура проекта при этом будет иметь вид, показанный на рис. 4:

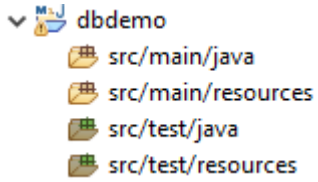

Рис. 4. – Структура нового Maven-проекта

Нас будет интересовать исключительно папка src/main/java, в которой будут находиться все исходные тексты нашего приложения.

Для подключения библиотек требуется создать элемент dependencies и каждую библиотеку вписать как отдельный элемент dependency.

Например, в случае Postgresql драйвер JDBC для доступа к данной СУБД может быть добавлен в конфигурацию следующим образом:

```
  <dependencies>
     <dependency>
          <groupId>org.postgresql</groupId>
          <artifactId>postgresql</artifactId>
          <version>9.4.1211</version>
     </dependency>
  </dependencies>
```

Для сборки проекта Maven используется команда mvn compile.

После сборки (из IDE или командной строки mvn compile) мы получим в случае успеха следующие сообщения:

```
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------------< org.dstu:dbdemo >--------------------------
[INFO] Building dbdemo 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ dbdemo ---
[WARNING] Using platform encoding (Cp1251 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ dbdemo ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1251, i.e. build is platform dependent!
[INFO] Compiling 1 source file to C:\Users\max\eclipse-workspace\dbdemo\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.245 s
[INFO] Finished at: 2019-12-22T17:13:08+03:00
[INFO] ------------------------------------------------------------------------
```

Чтобы не получать предупреждений о платформенно-зависимой сборке, желательно явно указать кодировку исходных файлов и текстовых ресурсов:

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

```
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------------< org.dstu:dbdemo >--------------------------
[INFO] Building dbdemo 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ dbdemo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ dbdemo ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\Users\max\eclipse-workspace\dbdemo\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.290 s
[INFO] Finished at: 2019-12-22T17:14:44+03:00
[INFO] ------------------------------------------------------------------------
```

Порядок выполнения контрольной работы

По аналогии с предыдущей работой создайте программу, которая получает на вход CSV файл с информацией об объектах согласно варианту. Программа считывает эти данные и помещает их в БД. При этом вся работа с БД должна вестись посредством объектно-реляционного отображения.

Требования к программе.

- Конфигурация зависимостей должна быть выполнена посредством Maven.
- Входной файл задается в качестве параметра при запуске программы.
- Каждый объект должен содержать 3 - 5 свойств.
- Отражение таблиц на объектную модель должно быть задано с применением аннотаций.
- Реализуйте DAO для доступа к соответствующим сущностям.
- Реализуйте и продемонстрируйте работу следующих запросов:
- запроса на вставку новых данных (внесите в БД данные из CSV файла);
- запроса на вывод всех объектов из БД;
- запроса на изменение одного и нескольких объектов в БД;
    - запроса на выборку данных по условию (требуется задать условие, задействующее все таблицы, а сам запрос – в виде HQL - https://proselyte.net/tutorials/hibernate-tutorial/hibernate-query-language/).

Варианты заданий

1) студент, преподаватель, группа;
2) телевизор, монитор, категория товаров;
3) авторучка, карандаш, отдел канцтоваров;
4) катер, водный мотоцикл, причал;
5) диван, кровать, комната;
6) самолет, корабль, (аэро)порт;
7) штатный сотрудник, совместитель, кафедра;
8) эллипс, многоугольник, презентация;
9) книга, журнал, полка;
10) зачет, экзамен, предмет;

11) город, село, регион;
12) млекопитающее, птица, зоопарк;
13) детская коляска, автокресло, гараж;
14) цифровая и обычная фоторамка, фото;
15) кондиционер, обогреватель;
16) квадрат, прямоугольник, рисунок;
17) стол, стул, офис;
18) ноутбук, планшет, установленная операционная система;
19) HDD, SDD, сервер;
20) клавиатура, мышь, компьютер;
21) беговая дорожка, степпер, зал;
22) мобильный телефон, смартфон, приложение.

Содержание отчёта

Отчёт по контрольной работе должен содержать следующее:

1. Определение номера варианта (последние 2 цифры зачётной книжки по модулю 22, плюс 1).
2. Физическую модель данных (схему) для вашего варианта задания.
3. Конфигурацию Hibernate.
4. Код всех классов-сущностей.
5. Код DAO для доступа к данным через спроектированные сущности.
   - Если вы используете однотипные DAO, приведите какой-то один пример реализации DAO-класса.

- Если вы используете реализацию DAO с абстрактным и/или Generic-классом, приведите все описания абстракций и Generic, а также одну полную реализацию для какой-либо сущности.

6. Приведите код, касающийся всех запросов, указанных в задании и результаты выполнения каждого из них.

Внимание! Не допускается вставка в отчёт скриншотов из IDE на тёмном фоне, вместо этого рекомендуется вставить код в отчёт или его приложения с использованием моноширинного шрифта.

ЛИТЕРАТУРА

1. Гуськова, О.И. Объектно ориентированное программирование в Java : учебное пособие / О. И. Гуськова. - Москва : МПГУ, 2018. - 240 с. - ISBN 978-5-4263-0648-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1020593

2. Прохоренок, Н. А. Основы Java: Самоучитель Учебное пособие / Прохоренок Н.А. - СПб:БХВ-Петербург, 2017. - 704 с. ISBN 978-5-9775-3785-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/978545

3. Будилов, В. А. Интернет-программирование на Java: Пособие / Будилов В.А. - СПб:БХВ-Петербург, 2014. - 698 с. ISBN 978-5-9775-1931-1. - Текст : электронный. - URL: https://znanium.com/catalog/product/940239

4. Руководство по Hibernate / Записки задумчивого программиста – Proselyte. – Текст: электронный. – URL: https://proselyte.net/tutorials/hibernate-tutorial/introduction/