

## 第 1 章 Java 应用分层架构及软件模型

### 1.1 应用程序的分层体系结构

#### 1.1.1 区分物理分层和逻辑分层

#### 1.1.2 软件层的特征

#### 1.1.3 软件分层的优点

#### 1.1.4 软件分层的缺点

#### 1.1.5 Java 应用的持久化层

### 1.2 软件的模型

#### 1.2.1 概念模型

#### 1.2.2 关系数据模型

#### 1.2.3 域模型

### 1.3 小结

### 1.4 思考题

## 第 2 章 Java 对象持久化技术概述

### 2.1 直接通过 JDBC API 来持久化实体域对象

### 2.2 ORM 简介

#### 2.2.1 对象 关系映射的概念

#### 2.2.2 描述对象 关系映射信息的元数据

#### 2.2.3 访问 ORM 中间件的 API

#### 2.2.4 常用的 ORM 中间件

### 2.3 实体域对象的其他持久化模式

#### 2.3.1 主动域对象模式

### 2.3.2CMP 模式

### 2.3.3 运用 ORM 的 JPA 模式

## 2.4 小结

## 2.5 思考题

## 第 3 章第一个 helloapp 应用

### 3.1 创建 Hibernate 的配置文件

#### 3.1.1 用 Java 属性文件作为 Hibernate 配置文件

#### 3.1.2XML 格式的 Hibernate 配置文件

### 3.2 创建持久化类

### 3.3 创建数据库 Schema

### 3.4 创建对象 关系映射文件

#### 3.4.1 映射文件的文档类型定义(DTD)

#### 3.4.2 把 Customer 持久化类映射到 CUSTOMERS 表

### 3.5 通过 Hibernate API 操纵数据库

#### 3.5.1Hibernate 的初始化

#### 3.5.2Hibernate 的遗留初始化方式

#### 3.5.3 访问 Hibernate 的 Session 接口

### 3.6 运行 helloapp 应用

#### 3.6.1 创建用于运行本书范例的系统环境

#### 3.6.2 创建 helloapp 应用的目录结构

#### 3.6.3 把 helloapp 应用作为独立应用程序运行

#### 3.6.4 把 helloapp 应用作为 Java Web 应用运行

### 3.7 小结

### 3.8 思考题

## 第 4 章使用注解和 JPA

### 4.1 创建包含注解的持久化类

### 4.2 方式一： 注解和 Hibernate API

#### 4.2.1 创建 Hibernate 配置文件

#### 4.2.2 自动创建数据库表

#### 4.2.3 使用 Hibernate API

### 4.3 方式二： 注解和 JPA API

#### 4.3.1 创建 JPA 的配置文件

#### 4.3.2 使用 JPA API

#### 4.3.3 从 JPA API 中获得 Hibernate API

### 4.4 方式三： 对象 关系映射文件和 JPA API

### 4.5 小结

### 4.6 思考题

## 第 5 章对象 关系映射基础

### 5.1 持久化类的属性及访问方法

#### 5.1.1 基本类型属性和包装类型属性

#### 5.1.2 访问持久化类属性的方式

#### 5.1.3 在持久化类的访问方法中加入程序逻辑

#### 5.1.4 设置派生属性

#### 5.1.5 控制 insert 和 update 语句

5.1.6 映射枚举类型

5.2 处理 SQL 引用标识符

5.3 创建命名策略

5.4 设置数据库 Schema

5.5 运行范例程序

5.6 使用 Hibernate 的对象 关系映射文件

5.6.1 设置访问持久化类属性的方式

5.6.2 映射 Customer 类的虚拟 name 属性

5.6.3 忽略 Customer 类的 avgPrice 属性

5.6.4 映射 Customer 类的 sex 属性

5.6.5 映射 Customer 类的 totalPrice 派生属性

5.6.6 控制 insert 和 update 语句

5.6.7 映射 Customer 类的 description 属性

5.6.8 设置自定义的命名策略

5.6.9 设置数据库 Schema

5.6.10 设置类的包名

5.7 小结

5.8 思考题

第 6 章映射对象标识符

6.1 关系数据库按主键区分不同的记录

6.1.1 把主键定义为自动增长标识符类型

6.1.2 从序列(Sequence)中获取自动增长的标识符

- 6.2Java 语言按内存地址区分不同的对象
- 6.3Hibernate 用对象标识符(OID)来区分对象
- 6.4Hibernate 的内置标识符生成器的用法
  - 6.4.1increment 标识符生成器
  - 6.4.2identity 标识符生成器
  - 6.4.3sequence 标识符生成器
  - 6.4.4table 标识符生成器
  - 6.4.5auto 标识符生成器
- 6.5 映射自然主键
  - 6.5.1 映射单个自然主键
  - 6.5.2 映射复合自然主键
- 6.6 映射派生主键
- 6.7 使用 Hibernate 的对象 关系映射文件
  - 6.7.1increment 标识符生成器
  - 6.7.2identity 标识符生成器
  - 6.7.3sequence 标识符生成器
  - 6.7.4auto(native)标识符生成器
  - 6.7.5 映射单个自然主键
  - 6.7.6 映射复合自然主键
  - 6.7.7 映射派生主键
- 6.8 小结
- 6.9 思考题

## 第 7 章映射一对多关联关系

### 7.1 映射多对一单向关联关系

#### 7.1.1 TransientPropertyValueException 异常

#### 7.1.2 级联持久化

### 7.2 映射一对多双向关联关系

#### 7.2.1 建立持久化对象之间的关联关系

#### 7.2.2 级联删除

#### 7.2.3 父子关系

### 7.3 映射一对多双向自身关联关系

### 7.4 改进持久化类

### 7.5 使用 Hibernate 的对象 关系映射文件

#### 7.5.1 映射多对一单向关联关系

#### 7.5.2 映射一对多双向关联关系

#### 7.5.3 映射一对多双向自身关联关系

### 7.6 小结

### 7.7 思考题

## 第 8 章通过 JPA 和 Hibernate 操纵对象(上)

### 8.1 Java 对象在 JVM 中的生命周期

### 8.2 理解持久化缓存

#### 8.2.1 持久化缓存的作用

#### 8.2.2 脏检查及清理缓存的机制

### 8.3 Java 对象在持久化层的状态

- 8.3.1 临时对象(Transient Object)的特征
- 8.3.2 持久化对象(Persistent Object)的特征
- 8.3.3 被删除对象(Removed Object)的特征
- 8.3.4 游离对象(Detached Object)的特征
- 8.4 Session 接口的用法
  - 8.4.1 Session 的 save()方法和 persist()方法
  - 8.4.2 Session 的 load()方法和 get()方法
  - 8.4.3 Session 的 update()方法
  - 8.4.4 Session 的 saveOrUpdate()方法
  - 8.4.5 Session 的 merge()方法
  - 8.4.6 Session 的 delete()方法
  - 8.4.7 Session 的 replicate()方法
  - 8.4.8 Session 的 findById()方法
  - 8.4.9 Session 的 refresh()方法
- 8.5 EntityManager 接口的用法
- 8.6 通过 Hibernate API 级联操纵对象图
  - 8.6.1 级联保存临时对象
  - 8.6.2 更新持久化对象
  - 8.6.3 持久化临时对象
  - 8.6.4 更新游离对象
  - 8.6.5 遍历对象图
- 8.7 通过 JPA API 级联操纵对象图

8.8 小结

8.9 思考题

第 9 章通过 JPA 和 Hibernate 操纵对象(下)

9.1 与触发器协同工作

9.2 利用拦截器(Interceptor)生成审计日志

9.3 Hibernate 的事件处理 API

9.4 利用 Hibernate 的 Envers 生成审计日志

9.5 JPA 的事件处理 API

9.6 批量处理数据

9.6.1 通过 EntityManager 或 Session 来进行批量操作

9.6.2 通过 StatelessSession 来进行批量操作

9.6.3 通过 JPQL 来进行批量操作

9.6.4 直接通过 JDBC API 来进行批量操作

9.7 通过 JPA 访问元数据

9.8 调用存储过程

9.9 小结

9.10 思考题

第 10 章映射组成关系

10.1 建立精粒度对象模型

10.2 建立粗粒度关系数据模型

10.3 映射组成关系

10.3.1 区分值(Value)类型和实体(Entity)类型

10.3.2 在应用程序中访问具有组成关系的持久化类

10.4 映射复合组成关系

10.5 使用 Hibernate 的对象 关系映射文件

10.5.1 映射组成关系

10.5.2 映射复合组成关系

10.6 小结

10.7 思考题

第 11 章 Hibernate 的映射类型

11.1 Hibernate 的内置映射类型

11.1.1 Java 基本类型以及数字类型的 Hibernate 映射类型

11.1.2 Java 时间和日期类型的 Hibernate 映射类型

11.1.3 Java 大对象类型的 Hibernate 映射类型

11.1.4 JDK 自带的个别 Java 类的 Hibernate 映射类型

11.1.5 使用 Hibernate 内置映射类型

11.2 客户化映射类型

11.2.1 用客户化映射类型取代 Hibernate 组件

11.2.2 用 UserType 映射枚举类型

11.2.3 实现 CompositeUserType 接口

11.2.4 运行本节范例程序

11.3 使用 JPA Converter(类型转换器)

11.4 操纵 Blob 和 Clob 类型数据

11.5 小结

## 11.6 思考题

## 第 12 章映射继承关系

### 12.1 继承关系树的每个具体类对应一个表

#### 12.1.1 用注解来映射

#### 12.1.2 用对象 关系映射文件来映射

#### 12.1.3 操纵持久化对象

#### 12.1.4 其他映射方式

### 12.2 继承关系树的根类对应一个表

#### 12.2.1 用注解来映射

#### 12.2.2 用对象 关系映射文件来映射

#### 12.2.3 操纵持久化对象

### 12.3 继承关系树的每个类对应一个表

#### 12.3.1 用注解来映射

#### 12.3.2 用对象 关系映射文件来映射

#### 12.3.3 操纵持久化对象

### 12.4 选择继承关系的映射方式

### 12.5 映射复杂的继承树

#### 12.5.1 用注解来映射

#### 12.5.2 用对象 关系映射文件来映射

### 12.6 映射多对一多态关联

### 12.7 小结

### 12.8 思考题

## 第 13 章 Java 集合类

### 13.1 Set(集)

#### 13.1.1 Set 的一般用法

#### 13.1.2 HashSet 类

#### 13.1.3 TreeSet 类

#### 13.1.4 向 Set 中加入持久化类的对象

### 13.2 List(列表)

### 13.3 Map(映射)

### 13.4 小结

### 13.5 思考题

## 第 14 章 映射值类型集合

### 14.1 映射 Set(集)

### 14.2 映射 Bag(包)

### 14.3 映射 List(列表)

### 14.4 映射 Map (映射)

### 14.5 对集合排序

#### 14.5.1 在数据库中对集合排序

#### 14.5.2 在内存中对集合排序

### 14.6 映射组件类型集合

### 14.7 小结

### 14.8 思考题

## 第 15 章 映射实体关联关系

## 15.1 映射一对一关联

### 15.1.1 按照外键映射

### 15.1.2 按照主键映射

## 15.2 映射单向多对多关联

## 15.3 映射双向多对多关联关系

### 15.3.1 用@ManyToOne 注解映射双向关联

### 15.3.2 使用组件类集合

### 15.3.3 把多对多关联分解为两个一对多关联

## 15.4 小结

## 15.5 思考题

## 第 16 章 Hibernate 的检索策略

### 16.1 Hibernate 的检索策略简介

### 16.2 类级别的检索策略

#### 16.2.1 立即检索

#### 16.2.2 延迟检索

### 16.3 一对多和多对多关联的检索策略

#### 16.3.1 立即检索(FetchType.EAGER)

#### 16.3.2 多查询语句立即检索(FetchMode.SELECT)

#### 16.3.3 延迟检索(FetchType.LAZY)

#### 16.3.4 增强延迟检索(LazyCollectionOption.EXTRA)

#### 16.3.5 批量检索(@BatchSize 注解)

#### 16.3.6 使用子查询语句(FetchMode.SUBSELECT)

## 16.4 多对一和一对一关联的检索策略

### 16.4.1 立即检索(FetchType.EAGER)

### 16.4.2 延迟检索(FetchType.LAZY)

### 16.4.3 无代理延迟检索

### 16.4.4 批量检索(@BatchSize 注解)

## 16.5 控制左外连接检索的深度

## 16.6 在程序中动态指定立即左外连接检索

## 16.7 定义和检索对象图

## 16.8 用@FetchProfile 注解指定检索规则

## 16.9 属性级别的检索策略

## 16.10 小结

## 16.11 思考题

## 第 17 章检索数据 API(上)

### 17.1 检索方式简介

#### 17.1.1JPQL 检索方式

#### 17.1.2QBC 检索方式

#### 17.1.3 本地 SQL 检索方式

#### 17.1.4 关于本章范例程序

#### 17.1.5 使用别名

#### 17.1.6 多态查询

#### 17.1.7 对查询结果排序

#### 17.1.8 分页查询

17.1.9 检索单个对象(getSingleResult()方法)

17.1.10 按主键依次处理查询结果(属于 Hibernate 的功能)

17.1.11 可滚动的结果集(属于 Hibernate 的功能)

17.1.12 绑定参数

17.1.13 设置查询附属事项

17.1.14 定义命名查询语句

17.1.15 调用函数

17.1.16 元模型类(MetaModel Class)

17.2 设定查询条件

17.2.1 比较运算

17.2.2 范围运算

17.2.3 字符串模式匹配

17.2.4 逻辑运算

17.2.5 集合运算

17.2.6 case when 语句

17.3 小结

17.4 思考题

第 18 章 检索数据 API(下)

18.1 连接查询

18.1.1 默认情况下关联级别的运行时检索策略

18.1.2 立即左外连接

18.1.3 左外连接

- 18.1.4 立即内连接
- 18.1.5 内连接
- 18.1.6 立即右外连接
- 18.1.7 右外连接
- 18.1.8 交叉连接
- 18.1.9 隐式连接
- 18.1.10 关联级别运行时的检索策略
- 18.1.11 用 Tuple 包装查询结果
- 18.2 投影查询
  - 18.2.1 用 JavaBean 包装查询结果
  - 18.2.2 过滤查询结果中的重复元素
- 18.3 报表查询
  - 18.3.1 使用聚集函数
  - 18.3.2 分组查询
  - 18.3.3 优化报表查询的性能
- 18.4 高级查询技巧
  - 18.4.1 动态查询
  - 18.4.2 集合过滤
  - 18.4.3 子查询
  - 18.4.4 本地 SQL 查询
- 18.5 查询性能优化
  - 18.5.1 Hibernate API 中 Query 接口的 iterate()方法

18.5.2Hibernate 的查询缓存

18.6 小结

18.7 思考题

第 19 章 Hibernate 高级配置

19.1 配置数据库连接池

19.1.1 使用默认的数据库连接池

19.1.2 使用配置文件指定的数据库连接池

19.1.3 从容器中获得数据源

19.1.4 由 Java 应用本身提供数据库连接

19.2 配置事务类型

19.3 把 SessionFactory 与 JNDI 绑定

19.4 配置 JNDI

19.5 配置日志

19.6 使用 XML 格式的配置文件

19.7 小结

19.8 思考题

第 20 章 声明数据库事务

20.1 数据库事务的概念

20.2 声明事务边界的方式

20.3 在 mysql.exe 客户程序中声明事务

20.4Java 应用通过 JDBC API 声明 JDBC 事务

20.5Java 应用通过 Hibernate API 声明 JDBC 事务

20.5.1 处理异常

20.5.2 Session 与事务的关系

20.5.3 设定事务超时

20.6 Java 应用通过 Hibernate API 声明 JTA 事务

20.7 Java 应用通过 JTA API 声明 JTA 事务

20.8 小结

20.9 思考题

第 21 章 处理并发问题

21.1 多个事务并发运行时的并发问题

21.1.1 第一类丢失更新

21.1.2 脏读

21.1.3 虚读

21.1.4 不可重复读

21.1.5 第二类丢失更新

21.2 数据库系统的锁的基本原理

21.2.1 锁的多粒度性及自动锁升级

21.2.2 锁的类型和兼容性

21.2.3 死锁及其防止办法

21.3 数据库的事务隔离级别

21.3.1 在 mysql.exe 程序中设置隔离级别

21.3.2 在应用程序中设置隔离级别

21.4 在应用程序中采用悲观锁

21.4.1 利用数据库系统的独占锁来实现悲观锁

21.4.2 由应用程序实现悲观锁

21.5 利用版本控制来实现乐观锁

21.5.1 使用整数类型的版本控制属性

21.5.2 使用时间戳类型的版本控制属性

21.5.3 为持久化对象设置锁

21.5.4 强制更新版本

21.6 实现乐观锁的其他方法

21.7 小结

21.8 思考题

第 22 章管理 Hibernate 的缓存

22.1 缓存的基本原理

22.1.1 持久化层的缓存的范围

22.1.2 持久化层的缓存的并发访问策略

22.2 Hibernate 的二级缓存结构

22.3 管理 Hibernate 的第一级缓存

22.4 管理 Hibernate 的第二级缓存

22.4.1 获得 EHCACHE 缓存插件的类库

22.4.2 在 persistence.xml 文件中配置第二级缓存

22.4.3 在持久化类中启用实体数据缓存、自然主键缓存和集合缓存

22.4.4 设置 EHCACHE 的 ehcache.xml 配置文件

22.4.5 获取第二级缓存的统计信息

22.4.6 设置第二级缓存的读写模式

22.4.7 在程序中控制第二级缓存

22.4.8 查询缓存

22.5 小结

22.6 思考题

第 23 章管理 Session 和实现对话

23.1 管理 Session 对象的生命周期

23.1.1 Session 对象的生命周期与本地线程绑定

23.1.2 Session 对象的生命周期与 JTA 事务绑定

23.2 实现对话

23.2.1 使用游离对象

23.2.2 使用手动清理缓存模式下的 Session

23.2.3 通过 JPA API 来实现对话

23.3 小结

23.4 思考题

第 24 章 Spring、JPA 与 Hibernate 整合

24.1 本章范例所涉及软件的 Java 类库

24.2 设置 Spring 的配置文件

24.3 编写本章范例的 Java 类

24.3.1 编写 Customer 实体类

24.3.2 编写 CustomerDao 数据访问接口和类

24.3.3 编写 CustomerService 业务逻辑服务接口和类

24.3.4 编写测试类 Tester

24.4 小结

第 25 章运用 Spring 和 Hibernate 创建购物网站

25.1 实现业务数据

25.2 实现业务逻辑

25.3 控制层访问模型层

25.4 netstore 应用的订单业务

25.5 小结

附录 A 标准 SQL 语言的用法

A.1 数据完整性

A.1.1 实体完整性

A.1.2 域完整性

A.1.3 参照完整性

A.2 DDL 数据定义语言

A.3 DML 数据操纵语言

A.4 DQL 数据查询语言

A.4.1 简单查询

A.4.2 连接查询

A.4.3 子查询

A.4.4 联合查询

A.4.5 报表查询

附录 B Java 语言的反射机制

附录 C 发布和运行 netstore 应用

附录 D 思考题答案

附录 E 书中涉及软件获取途径