

Minilab3 Summary

项目目标：

使用JPA连接Minilab1项目中MySQL数据库的数据。

步骤：

- (1) 下载JPA所需要的库函数；
- (2) 创建实体java文件；
- (3) 创建Persistence.xml，此文件将配置数据库并制定实体类；
- (4) 使用JPA添加CRUD操作；
- (5) 根据测试结果修改代码，再测试，直到被测试的代码完全正确。

Questions:

1.What is persistent class?

持久化类：在应用程序中，用来实现业务问题实体的类（如，在电子商务应用程序中的Customer和Order）就是持久化类。不能认为所有的持久化类的实例都是持久的状态——一个实例的状态也可能是瞬时的或托管的。就如同它的名字暗示的，它的实例会被持久性保存于数据库中。

持久化是将程序数据在持久状态和瞬时状态间转换的机制。

持久化类(persistent class)：可以被hibernate保存到数据库，并且从数据库读取的类。

性质：

(1) 持久化类：是指其实例需要被Hibernate持久化到数据库中的类。持久化类符合JavaBean的规范，包含一些属性，以及与之对应的getXXX()和setXXX()方法。

注：get/set方法必须符合特定的命名规则，get和set后面紧跟属性的名字，并且属性名的首字母为大写。

(2) 如果持久化类的属性为boolean类型，那么他的get方法名即可以用get作为前缀，也可以用is作为前缀。

(3) 持久化类有一个id属性，用来唯一标识Account类的每一个对象。这个id属性被称为对象标示符 (OID, Object Identifier)，通常它都用整数表示。

(4) Hibernate要求持久化类必须提供一个不带参的默认构造方法，在程序运行时，Hibernate运用Java反射机制，调用java.Lang.reflect.Constructor.newInstance()方法来构造持久化类的实例。

2.What are the differences between JPA and JDBC? What are their pros and cons?

JPA与JDBC的区别：

JDBC：Java Data Base Connectivity，java数据库连接，用于直接调用SQL 命令，也就是用于执行SQL语句的Java API，是面向数据库的。

JPA：Java Persistence API，Java持久性API，用来操作实体对象，持久性提供了很多实现，编程人员只需要编写实体类，实体类中的主要信息为实体与数据库中表、字段、主键的对应，可以免除编写繁琐的SQL。

JPA的优缺点：

优点：

- (1) 相对与XML来说, 配置简单；
- (2) JPQL和HQL差不多；
- (3) EntityManager有些细节地方比Session好用；
- (4) 未来的演化方向，Hibernate会逐渐放弃对传统查询的支持。

缺点：

- (1) JPQL和HQL比起来变化不大，功能仍然不够强大。较复杂的查询，仍然需要切换到原生SQL
- (2) EntityManager有些细节没有Session好用；
- (3) (JPACriteria) 多了一步代码生成，却没有减少多少复杂度，得不偿失。

JDBC的优缺点：

优点：

- (1) Client-Server架构；
- (2) 支持多线程；
- (3) 设定JDBC环境简单；
- (4) 避免每个client安装；
- (5) 跨平台，资料库转移client不需要重新设定。

缺点：

- (1) 需要事先学习java的基本知识；
- (2) JDBC需要考虑数据库之间的差异。

3.What is the difference between persistence.xml and hibernate.cfg.xml?

persistence.xml是JPA的配置文件，用来映射PU(Persist Unit)的作用是映射表和类，里面也可以配置数据库连接信息。

Hibernate配置文件主要用于配置数据库连接和Hibernate运行时所需的各种属性，这个配置文件应该位于应用程序或Web程序的类文件夹 classes中。Hibernate配置文件支持两种形式，一种是xml格式的配置文件，另一种是Java属性文件格式的配置文件，采用“键=值”的形式。建议采用xml格式的配置文件。xml配置文件可以直接对映射文件进行配置，并由Hibernate自动加载，而properties文件则必须在程序中通过编码加载映射文件。

4. Explain Life Cycle of a JPA Entity.

JPA Entity 生命周期：

JPA中的Entity物件的状态，可以分为四种：New、Managed、Detached、Removed。以下使用Application-Managed EntityManager为例做说明：

New

直接使用new创建出Entity物件，这些物件还没有和资料库发生任何的关系，不对应于资料库中的任一处资料，没有主键对应。

Managed

当物件与资料库中的资料有对于关系，而且与EntityManager实例有关系而EntityManager实例尚未关闭（close），则它是在Managed状态，具体而言，如果将New状态的物件使用EntityManager的persist()或merge()方法加以储存、合并，或是使用find()从资料库载入资料并封装为物件，则该物件为Managed状态。

Managed状态的Entity是在 PersistenceContext 的管理之中，Managed状态的物件对应于资料库中的一笔资料，物件的id值与资料的主键值相同，并且EntityManager实例尚未失效，在这期间对物件的任何状态变动，在EntityManager实例关闭（close）或交易确认之后，资料库中对应的资料也会跟著更新。

如果将EntityManager实例关闭（close），则PersistenceContext失效，Managed状态的物件会成为Detached状态。

Detached

Detached状态的物件，其id与资料库的主键值对应，但脱离EntityManager实例的管理，例如在使find()方法查询到资料并封装为物件之后，将EntityManager实例关闭，则物件由Managed状态变为Detached状态，Detached状态的物件之任何属性变动，不会对资料库中的资料造成任何的影响。

Detached状态的物件可以使用merge()方法，使之与资料库中的对应资料再度发生关系，此时Detached状态的物件会变为Managed状态，也就是被PersistenceContext所管理。

Removed

如果您使用EntityManager实例的remove()方法删除资料，Managed状态的物件由于失去了对应的资料，则它会成为Removed状态，一个成为Removed状态的物件不应该被继续重用，您应该释放任何参考至它的名稱，讓該物件在適當的時候被垃圾回收。

简单的说，New与Detached状态的物件未受EntityManager管理，也就是不在PersistenceContext管理之中，对这两个状态的物件作任何属性变动，不会对资料库中的资料有任何的影响，而Managed状态的物件受EntityManager管理，也就是在PersistenceContext管理之中，对物件的属性变动，在EntityManager实例关闭（close）或交易确认之后，资料库中对应的资料也会跟著更新。

JPA Entity 生命周期如图所示：

