Hibernate入门

开始“冬眠”

Hibernate简介

- Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以使用面向对象编程思维来操纵数据库。Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序中使用，也可以在Servlet/JSP的Web应用中使用。
- Hibernate不仅管理Java类到数据库表的映射(包括Java数据类型到SQL数据类型的映射)，还提供数据查询和获取数据的方法，可以大幅度缩短使用JDBC处理数据持久化的时间。

工作原理

- 通过配置文件在值对象和数据库表之间建立起一个映射关系，这样，只需要通过操作这些值对象和Hibernate提供的一些基本类，就可以达到使用数据库的目的。
- Hibernate就是介于Java与JDBC之间的一个持久层，它通过建立数据库表之间的映射来操纵数据库。Hibernate是基于JDBC之上的。

数据库操作的三种方式

- 操作JDBC
- 封装JDBC
- ORM

ORM对象关系映射

- ORM – Object Relational Mapping
- 为了解决面向对象与关系数据库存在的互不匹配的现象的技术。简单地说，ORM是通过描述对象和数据库之间映射的元数据，将Java程序中的对象自动持久化到关系数据库中。本质上就是将数据从一种形式转换为另一种形式。

ORM对象关系映射

- 如今，面向对象思想大行其道，同时关系型数据库也几乎存在于所有应用程序中。在业务逻辑层和用户界面层中，我们是面向对象的。当对象信息发生变化的时候，我们需要把对象的信息保存在关系数据库中。在传统的开发方法中，你可能会写不少数据访问层的代码，用来从数据库保存，删除，读取对象信息等。但是引入ORM之后，你只需要用O/R Mapping保存，删除，读取对象，O/R Mapping负责生成SQL，你只需要关系对象就可以了。

持久层概念

- ORM是通过使用描述对象和数据库之间映射的元数据，将Java程序中的对象自动持久化到关系数据库中。
- 持久化：对数据和程序状态的保持。大多数情况下特别是企业级应用，数据持久化往往也就意味着将内存中的数据保存到磁盘上加以固化，而持久化的实现过程则大多通过各种关系数据库来完成。
- 持久层框架主要有两种方向：直接编写JDBC等SQL语句；使用O/R Mapping技术实现的Hibernate和JDO技术 。另外还有EJB中的实体Bean技术。

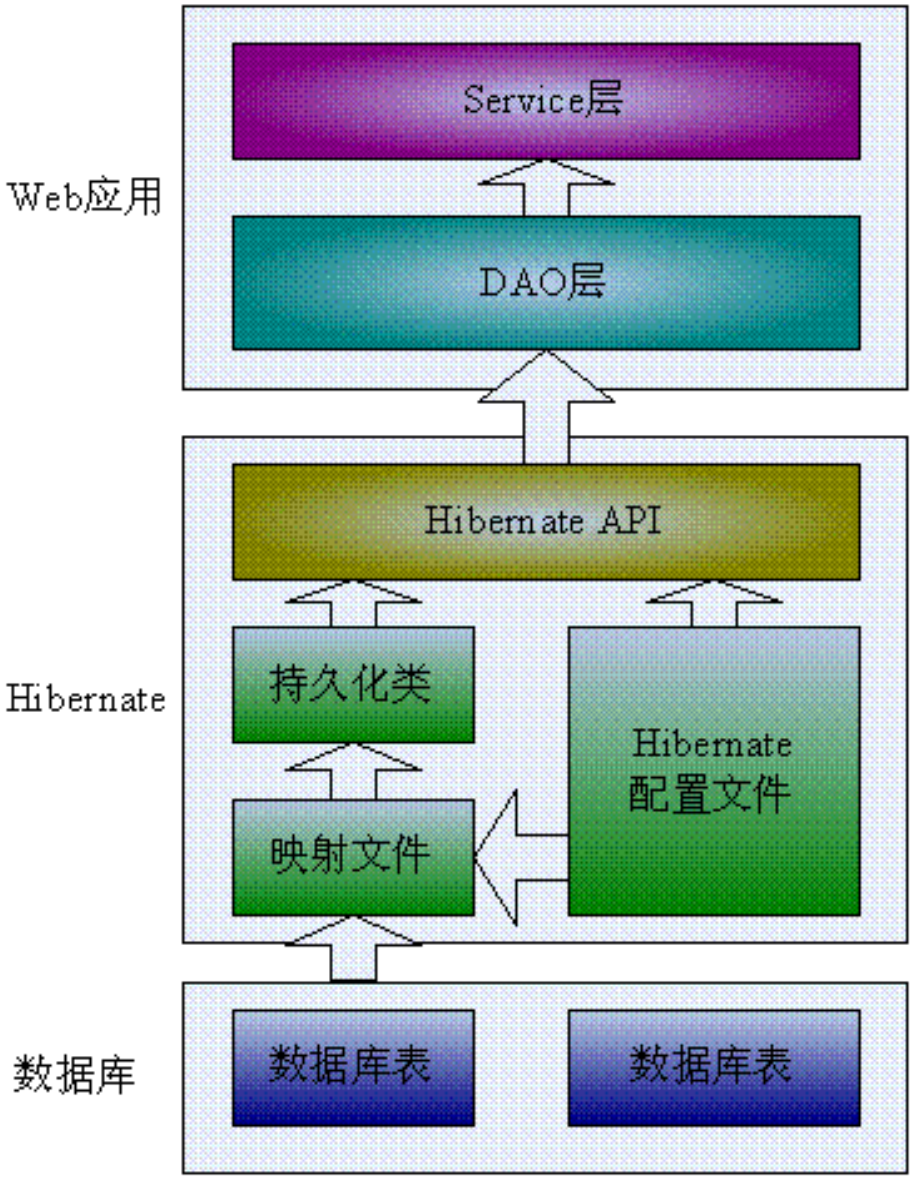
基本映射方式

- 1. 数据表映射类：持久化类被映射到一个数据表。
- 2. 数据表的行映射对象(即实例)：持久化类会生成很多实例，每个实例就对应数据表中的一行记录。当我们在应用中修改持久化类的某个实例时，ORM工具将会转换成对对应数据表中特定行的操作。
- 3. 数据表的列(字段)映射对象的属性：当我们在应用中修改某个持久化对象的指定属性时，ORM将会转换成对对应数据表中指定数据行，指定列的操作。

Hibernate开发流程

- Hibernate内部封装了通过JDBC访问数据库的操作，向上层应用提供面向对象的数据访问API。在Java应用中使用Hibernate包含以下步骤：
- 1. 创建Hibernate配置文件：该文件负责初始化Hibernate配置，包括数据库配置和映射文件配置；
- 2. 创建Hibernate映射文件：每一个数据表对应一个映射文件，该文件描述了数据库中表的信息，也描述了对应的持久化类的信息；
- 3. 创建持久化类：每一个类对应一个数据库表，通过映射文件来关联。
- 4. 编写DAO层：通过Hibernate API编写访问数据库的代码
- 5. 编写Service层：编写业务层实现，调用DAO层类代码。

Hibernate开发流程图



下载安装Hibernate

- Hibernate下载包目录结构介绍

- 由于Hibernate底层依然是基于JDBC的，因此在应用程序中使用Hibernate执行持久化时一定少不了JDBC驱动，比如我们使用的是MySQL数据库，因此还需要将MySQL数据库驱动添加到应用的类加载路径中。

Hibernate配置文件详解

- Hibernate的数据库连接信息是从配置文件中加载的，在配置文件中包含了一系列属性的配置，Hibernate将根据这些属性来连接数据库。
- Hibernate的配置文件有两种形式：一种是XML格式的文件，一种是properties属性文件。
- 在XML格式的配置文件中，除了基本的Hibernate配置信息，还可以指定具体的持久化类的映射文件，这可以避免将持久化类的配置文件硬编码在程序中。XML格式的配置文件的默认文件名为hibernate.cfg.xml。

Hibernate映射文件

- 1. 根元素<hibernate-mapping>
  - package属性
- 2. 使用<class>定义类
- 3. 使用<id>定义主键
  - 在关系数据库表中，主键(Primary Key)用来识别记录，并保证每条记录的唯一性。在Java语言中，通过比较两个变量所引用对象的内存地址是否相同，或者比较两个变量引用的对象值是否相同来判断两个对象是否相等。Hibernate为了解决两者之间的不同，使用对象标识符(OID)来标识对象的唯一性。OID是关系数据库中主键在Java对象模型中的等价物。

Hibernate映射文件

- 4. 使用<generator>设置主键生成方式
  - Identity：采用数据库提供的主键生成机制。
- 5. 使用<property>定义属性

Hibernate.cfg.xml配置文件

- 1. JDBC连接属性
  - 数据库方言：Hibernate底层依然使用SQL语句来执行数据库操作，虽然所有关系型数据库都支持使用标准SQL语句，但所有数据库都对标准SQL进行了一些扩展，所以语法细节上存在一些差异，因此Hibernate需要根据数据库来识别这些差异

Hibernate.cfg.xml配置文件

- 2. 数据源的连接属性
  - Hiberate并不推荐采用DriverManager来连接数据库，而是推荐使用数据源来管理数据库连接，这样能保证最好的性能。Hibernate推荐C3P0数据源。
  - 数据源是一种提高数据库连接性能的常规手段，数据源会负责维持一个数据连接池，当程序创建数据源实例时，系统会一次性地创建多个数据库连接，并把这些数据库连接保存在连接池中。当程序需要进行数据库访问时，无需进行重新获得数据库连接，而是从连接池中取出一个空闲的数据库连接。当程序使用数据库连接访问数据库结束后，无需关闭数据库连接，而是将数据库连接归还给连接池即可。

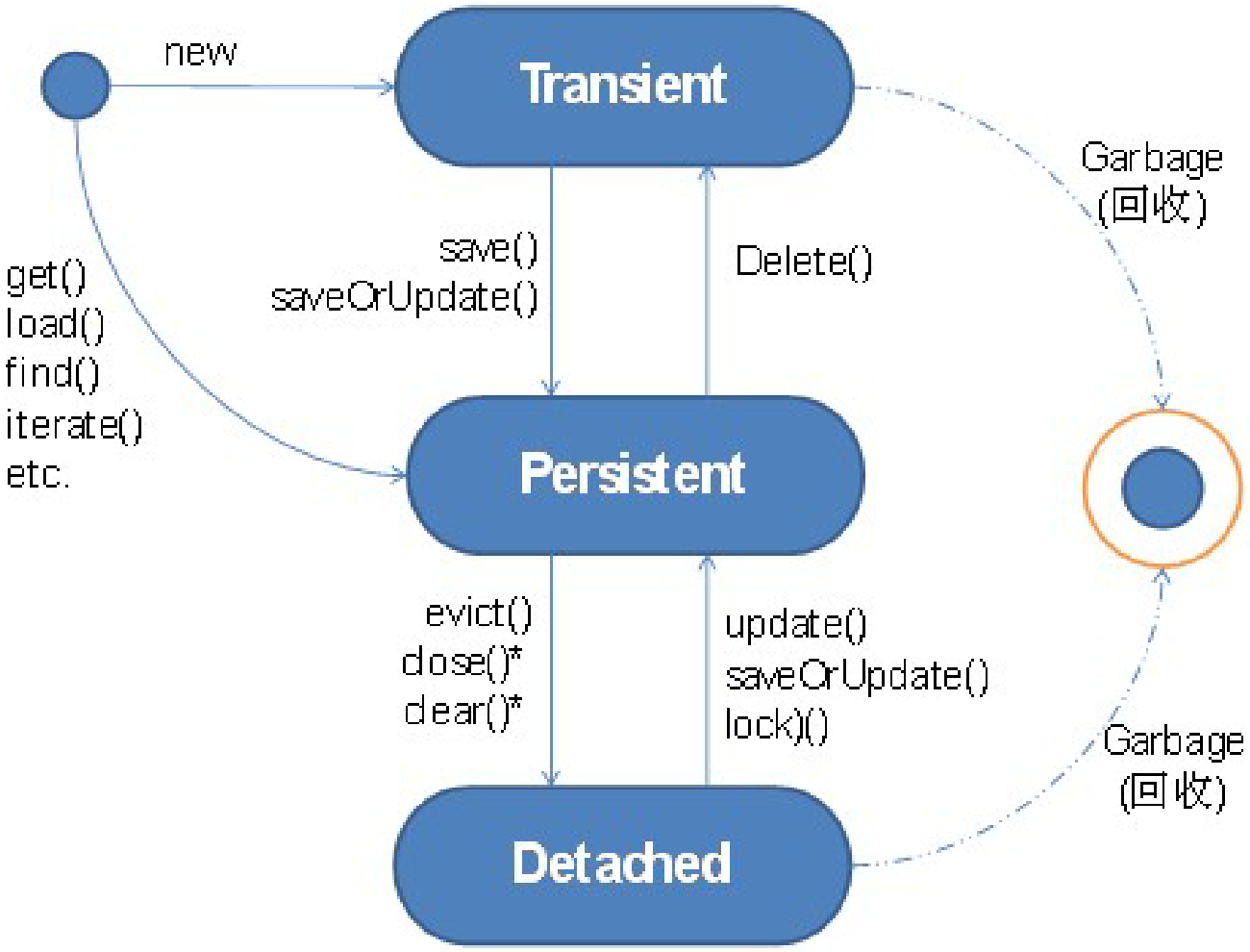
Hibernate持久化操作步骤

- 1. 开发持久化类，由POJO加映射文件组成（PO）
- 2. 获取Configuration
- 3. 获取SessionFactory
- 4. 获取Session，打开事务
- 5. 用面向对象的方式操作数据库
- 6. 关闭事务，关闭Session

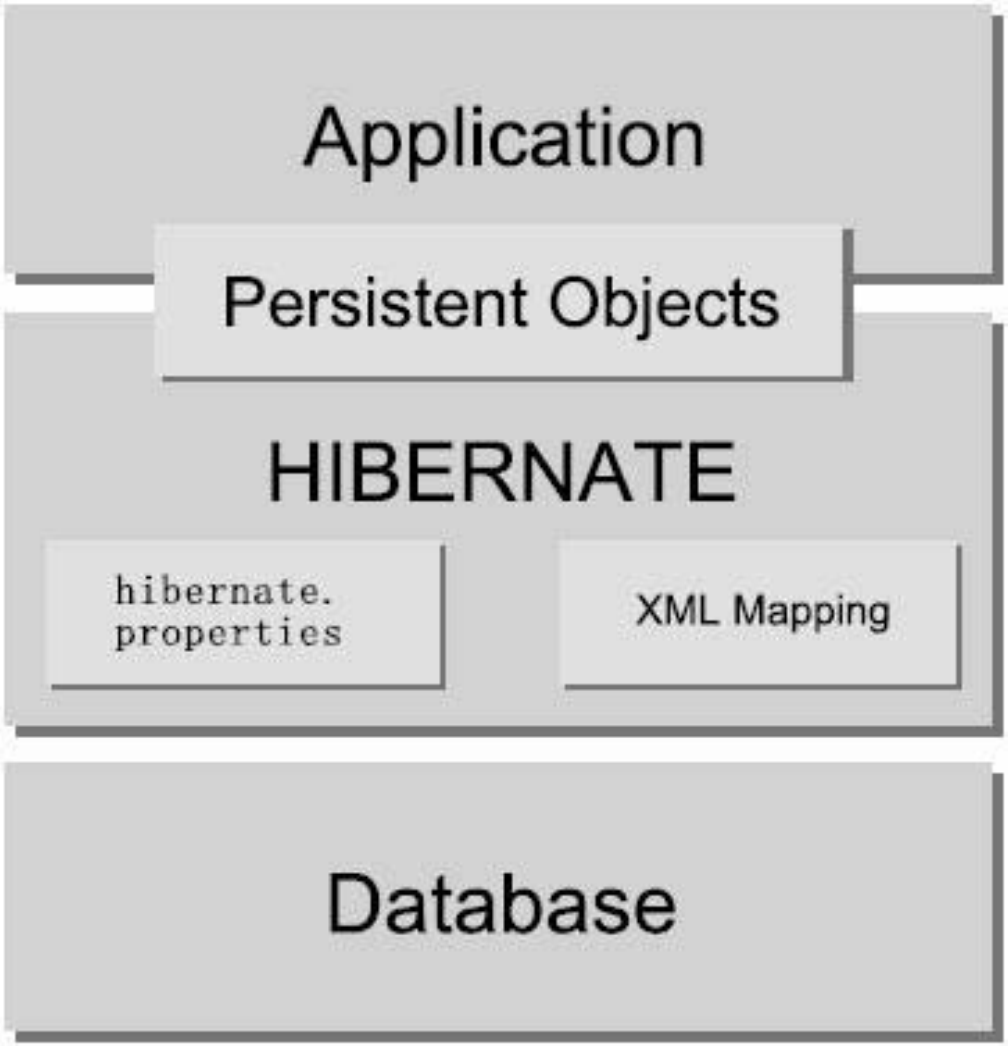
PO的三种状态

- 随PO与Session的三种关联关系，PO可有如下三种状态：
- 1. 瞬态：如果PO实例从未与Session关联过，该PO实例处于瞬态状态。
- 2. 持久化：如果PO实例与Session关联起来，且该实例对应到数据库记录，则该实例处于持久化状态。
- 3. 如果PO实例曾经与Session关联过，但因为Session关闭等原因，PO实例脱离了Session的管理，这种状态被称为脱管状态。
- 对于PO的操作必须在Session的管理下才能同步到数据库。Session由SessionFactory工厂产生，SessionFactory是数据库编译后的内存镜像，通常一个应用对应一个SessionFactory对象。SessionFactory对象由Configuration对象生成，Configuration对象负责加载Hibernate配置文件。

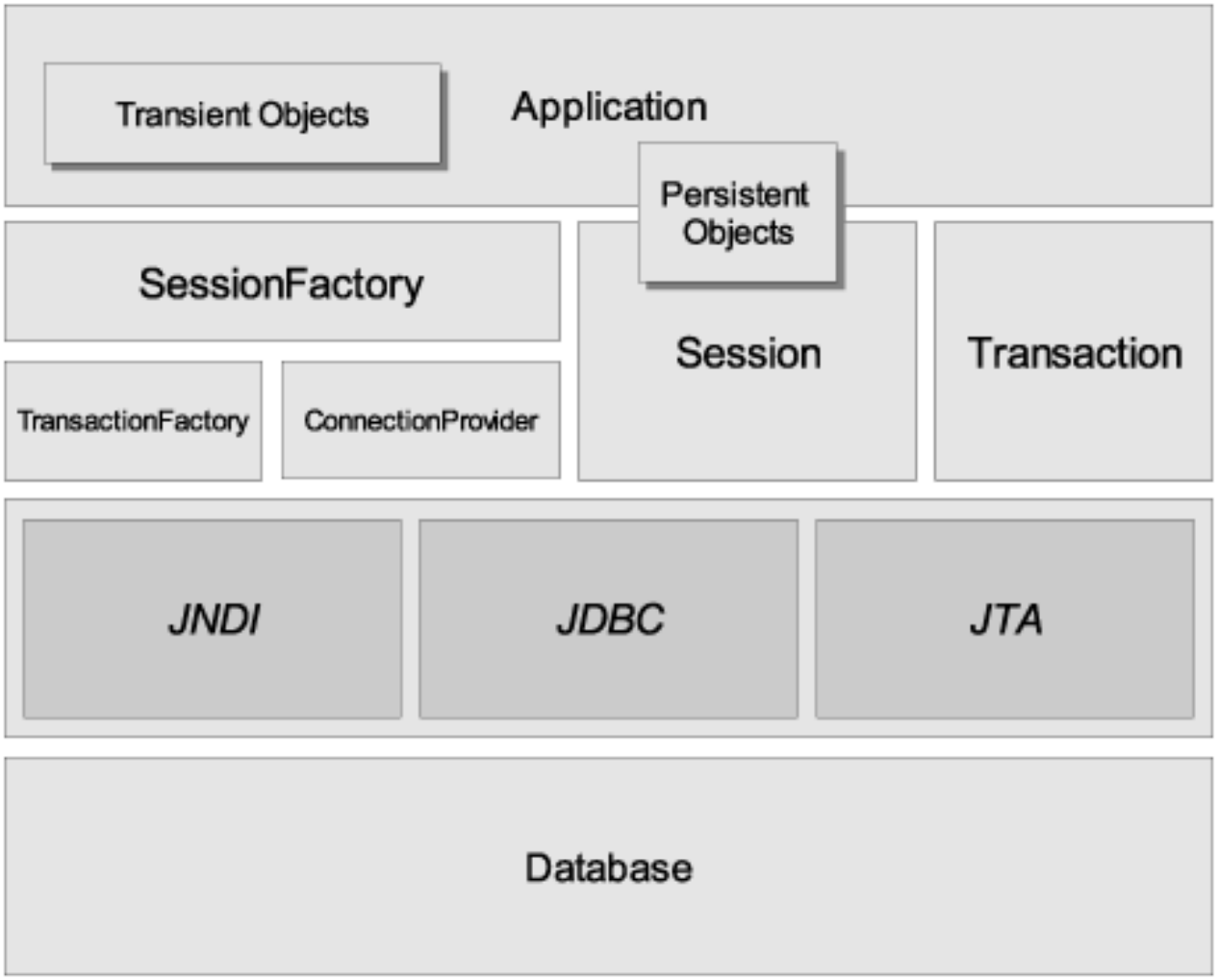
PO的三种状态


new
Transient
Garbage
(回收)
save()
saveOrUpdate()
Delete()
get()
load()
find()
iterate()
etc.
Persistent
evict()
close()*
clear()*
update()
saveOrUpdate()
lock)()
Garbage
(回收)
Detached

Hibernate体系结构



Hibernate体系结构



深入理解持久化对象

- Hibernate采用低侵入式设计，这种设计对持久化类几乎不作任何要求。也就是说，Hibernate操作的持久化类基本上都是普通的，传统的Java对象(POJO).
- 虽然Hibernate对持久化类没有太多的要求，但我们还是应该遵守如下规则：
  - 1. 提供一个无参数的构造器
  - 2. 提供一个标识属性：虽然Hibernate允许持久化类没有标识符属性，而是让Hibernate内部来追踪对象的识别，但这样做将导致Hibernate许多功能无法使用。
  - 3. 为持久化类的每个属性提供setter和getter方法：Hibernate默认采用属性方式来访问持久化类的属性。
  - 4. 使用非final的类
  - 5. 重写equals()和hashCode()方法