河北师范大学软件学院
Software College of Hebei Normal University

HIBERNATE

第一讲
Hibernate框架的搭建

BIG DATA Java与大数据分析

Java与移动智能设备开发

目录

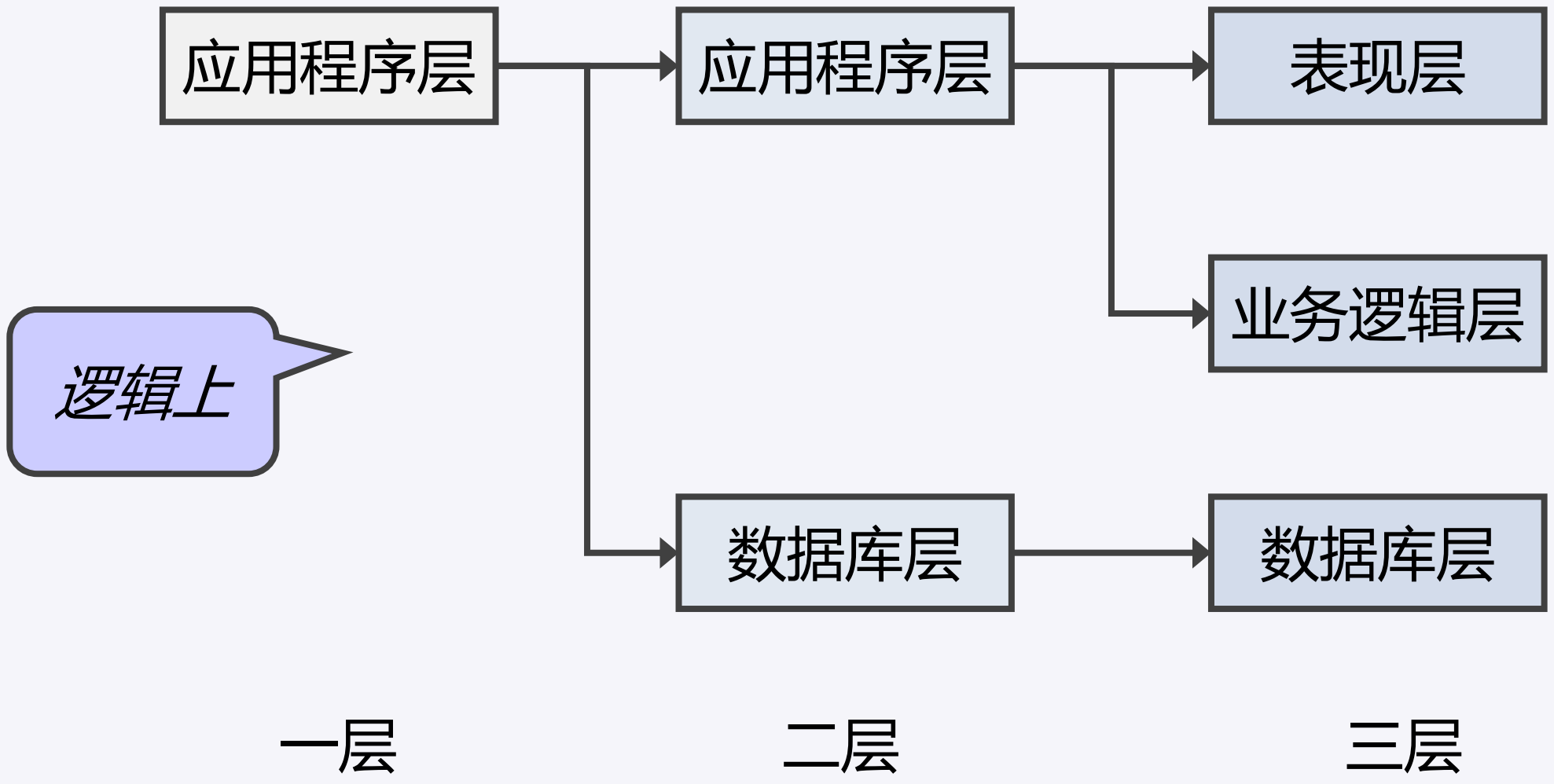
1 **分层体系结构与持久化**

2 软件的模型及ORM

3 Hibernate介绍

4 第一个Hibernate程序

三层体系结构



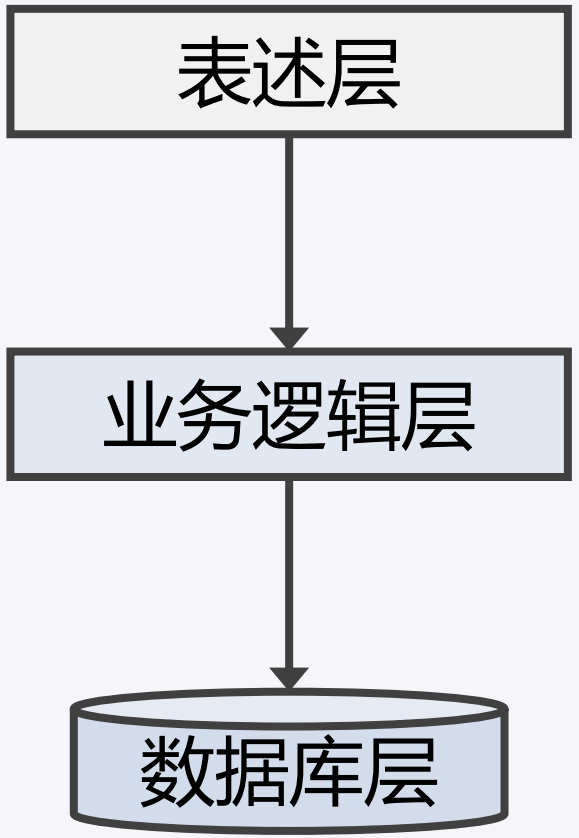
分层体系结构

- 分层体系结构（Layered Architecture）

  指的是将系统的组件分隔到不同的层中，每一层中的组件应保持内聚性，并且应大致在同一抽象级别；每一层都应与它下面的各层保持松散耦合。

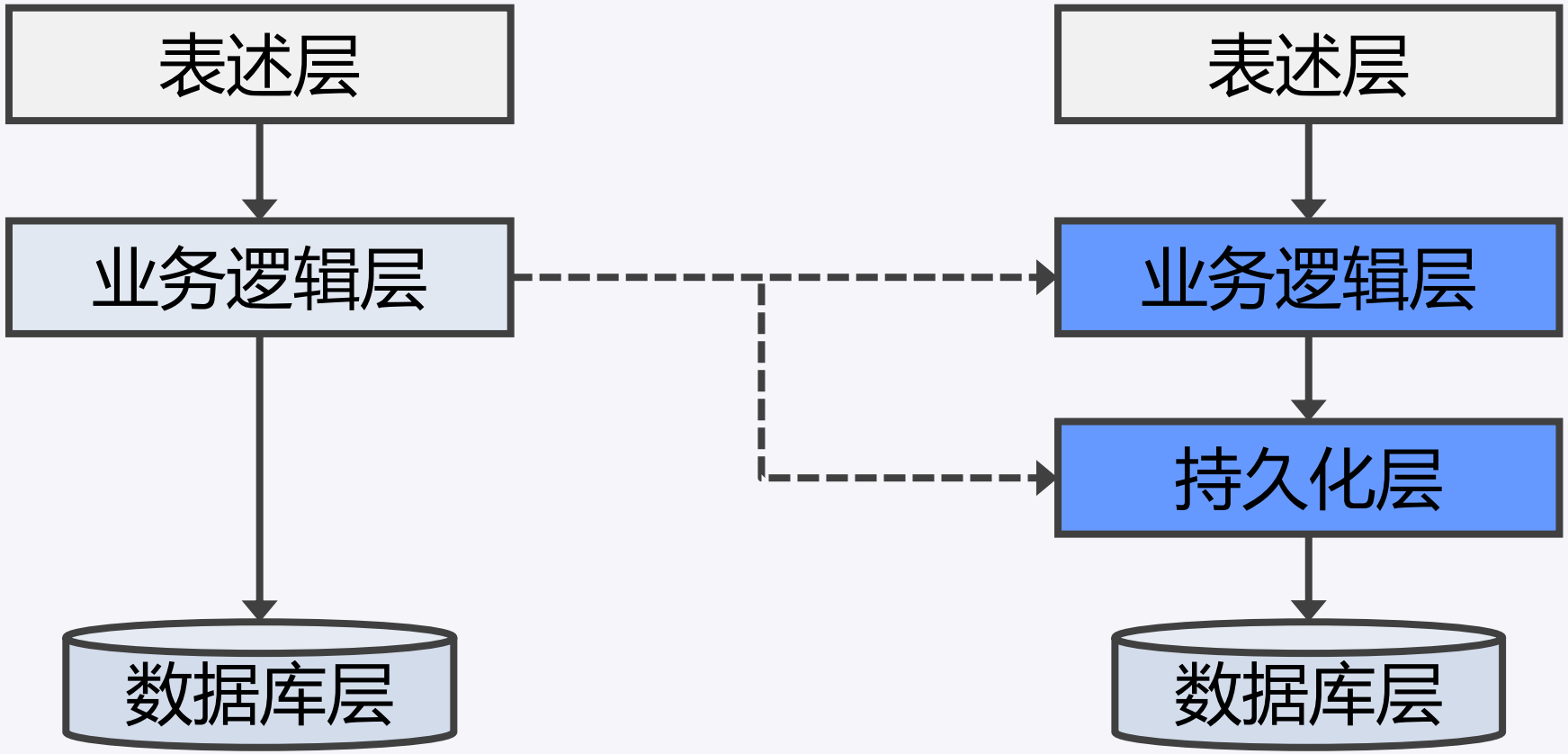
分层体系结构

- 层与层之间存在自上而下的依赖关系，即上层组件会访问下层组件的API，而下层组件不应该依赖上层组件。
- 每个层对上层公开API，但具体的实现细节对外透明。当某一层的实现发生变化，只要它的API不变，不会影响其他层的实现。



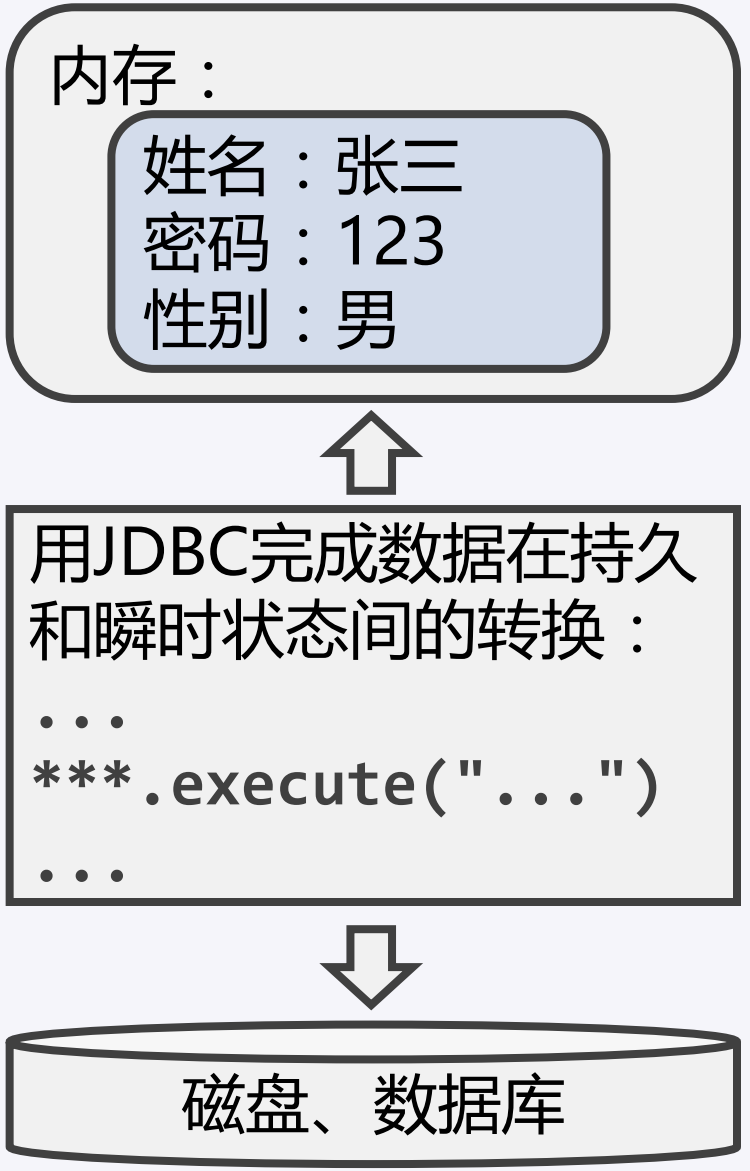
持久化层是怎么来的

- 为了把数据访问细节和业务逻辑分开，可以把数据访问作为单独的持久化层。



什么是持久化

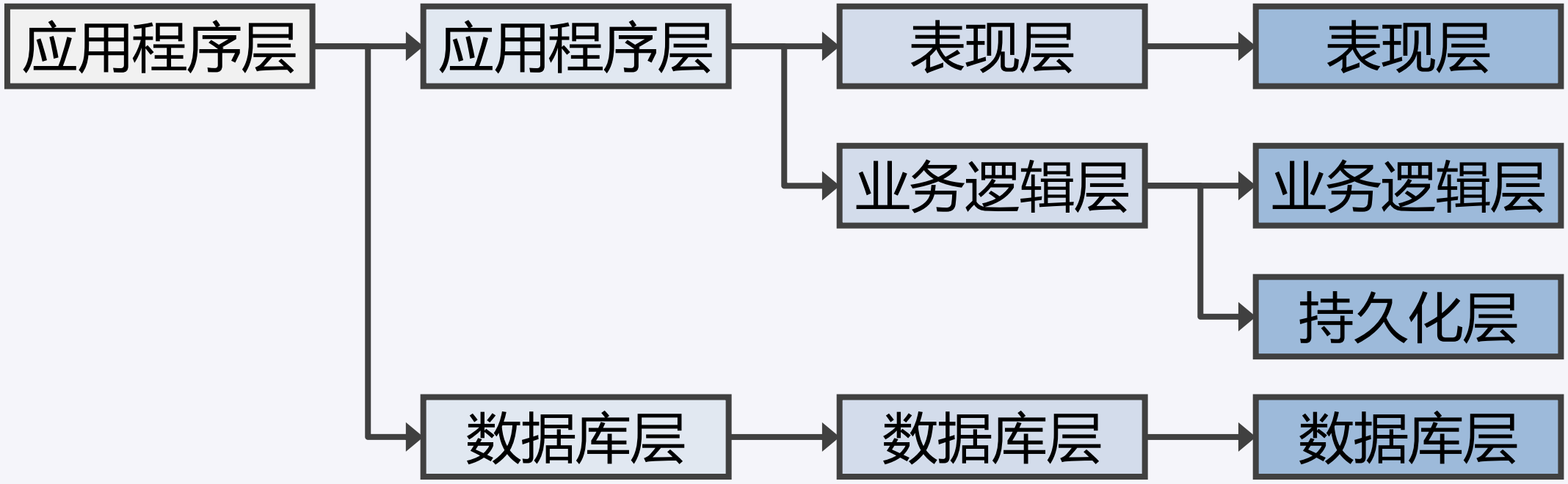
- 什么是持久化？
  - 瞬时状态：保存在内存的程序数据，程序退出后，数据就消失了，称为瞬时状态。
  - 持久状态：保存在数据库（磁盘）的程序数据，程序退出后，数据依然存在，称为程序数据的持久状态。
  - 持久化：将程序数据在瞬时状态和持久化状态之间转换的机制。



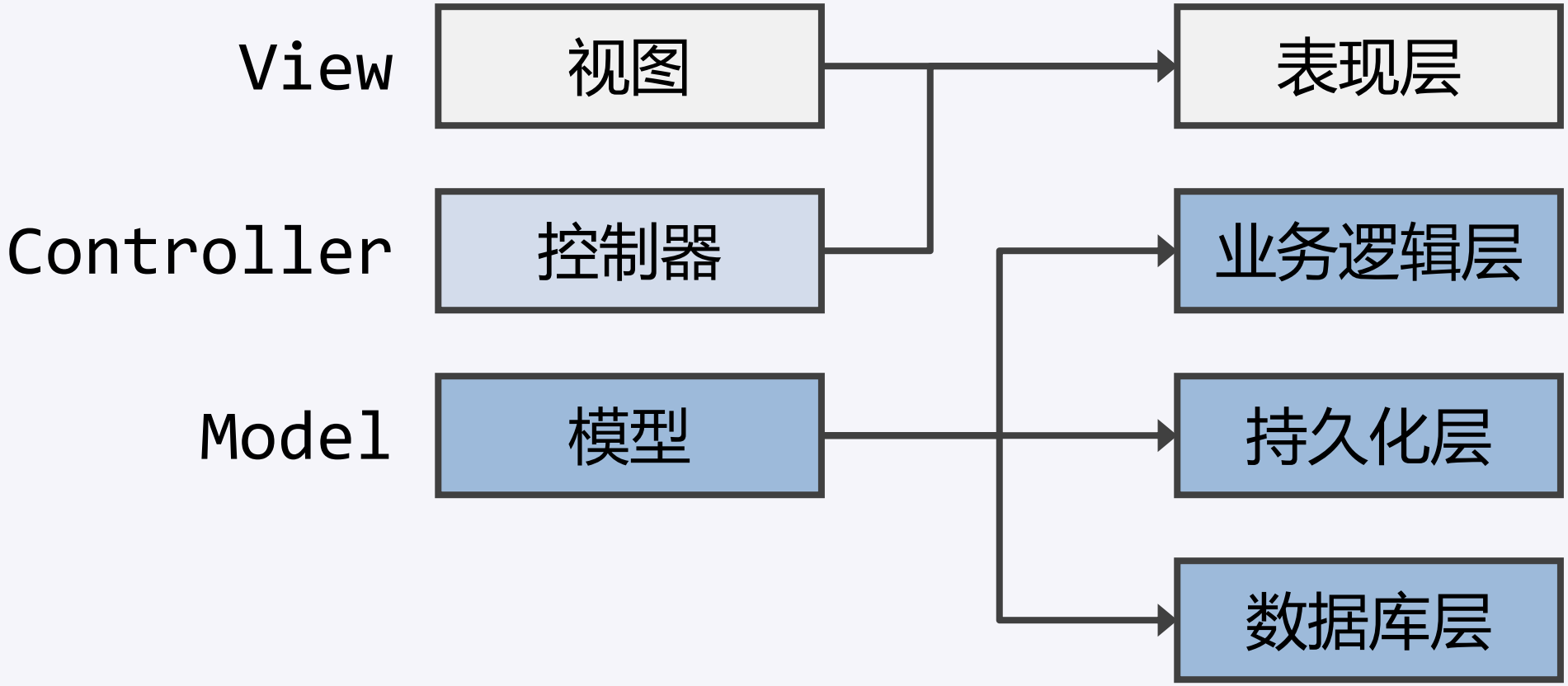
持久化层的作用

- 持久化层

　　持久化层封装了数据访问细节，为业务逻辑层提供面向对象的API，使业务逻辑层可以专注于实现业务逻辑。



MVC设计模式与四层结构的对应关系



持久化层设计目标

- 持久化层的设计目标：
  - 代码可重用性高，能够完成对象持久化操作；
  - 如果需要的话，能够支持多种数据库平台；
  - 具有相对独立性，当持久层发生变化时，不会影响上层实现。
- Hibernate是持久化层框架。

目录

1 分层体系结构与持久化

2 **软件的模型及ORM**

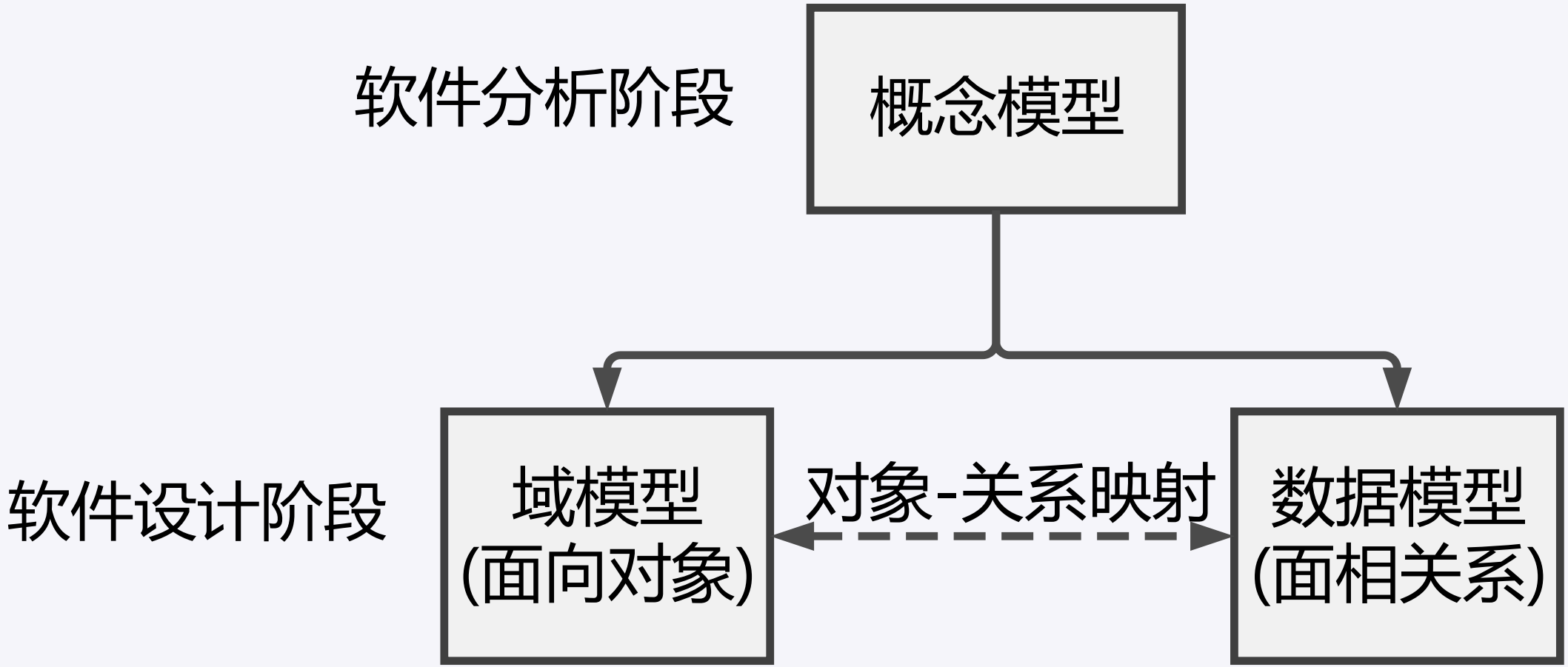
3 Hibernate介绍

4 第一个Hibernate程序

软件的模型

- 在软件开发领域，模型用来表示真实世界的实体。
- 在软件开发的不同阶段，需要为目标系统创建不同类型的模型：
  - 在分析阶段，需要创建概念模型；
  - 在设计阶段，需要创建域模型和数据模型。

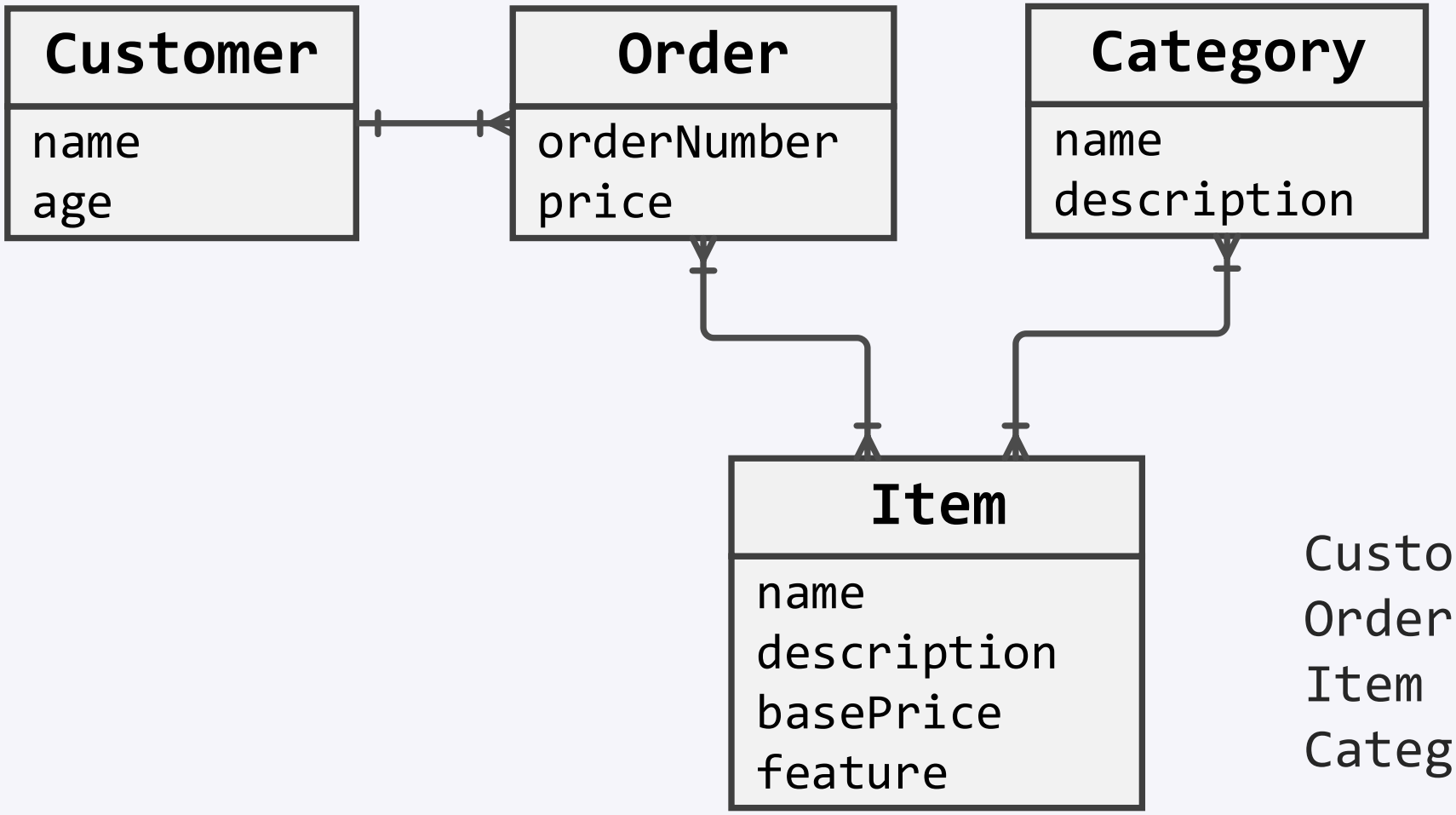
模型之间的关系



概念模型

- 概念模型用来模拟问题域中的真实实体。
- 概念模型描述了每个实体的概念和属性，以及实体之间的关系。
- 概念模型并不描述实体的行为。

购物网站应用的概念模型

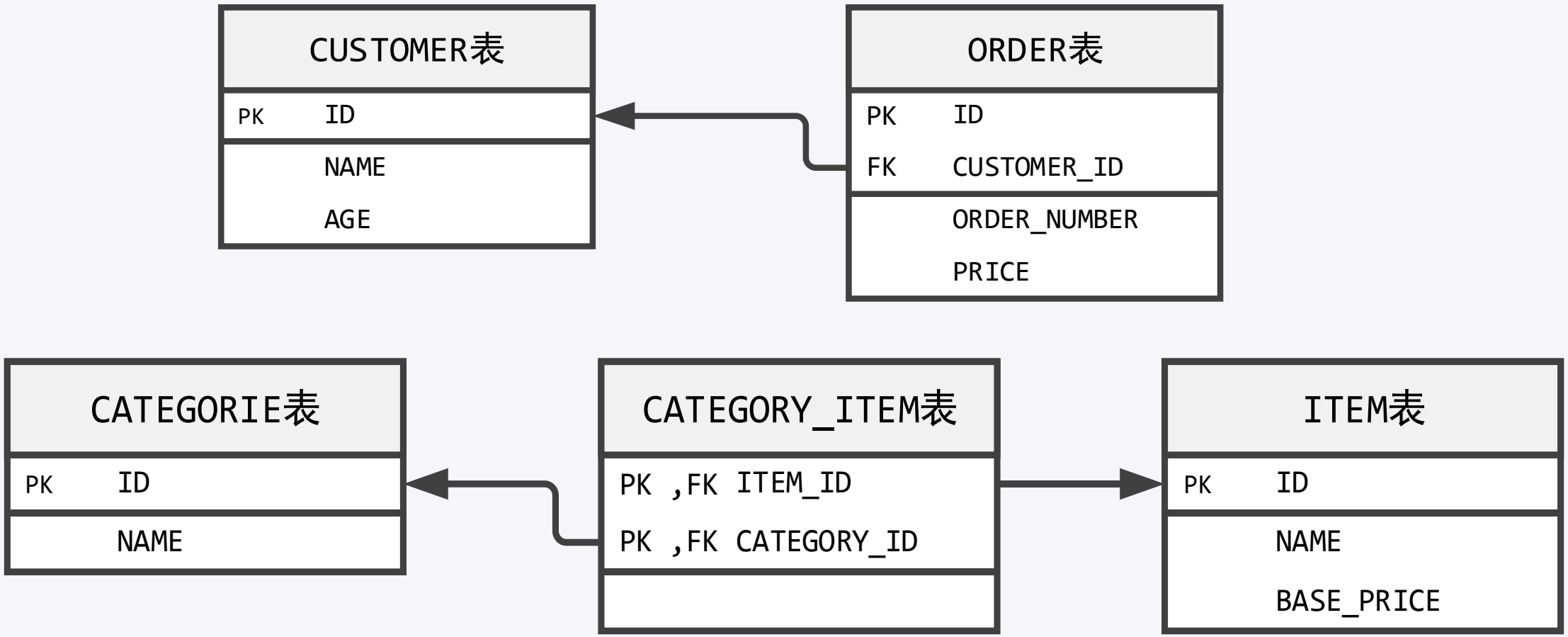


Customer：用户
Order　　：订单
Item　　：商品
Category：商品类别

关系数据模型

- 关系数据模型是在概念模型的基础上建立起来的，用于描述这些关系数据的静态结构，它由以下内容组成：
  - 一个或多个表；
  - 表与表之间的参照完整性；
  - 表的所有索引；
  - 触发器；
  - 视图。

表与表之间的参照完整性



域模型

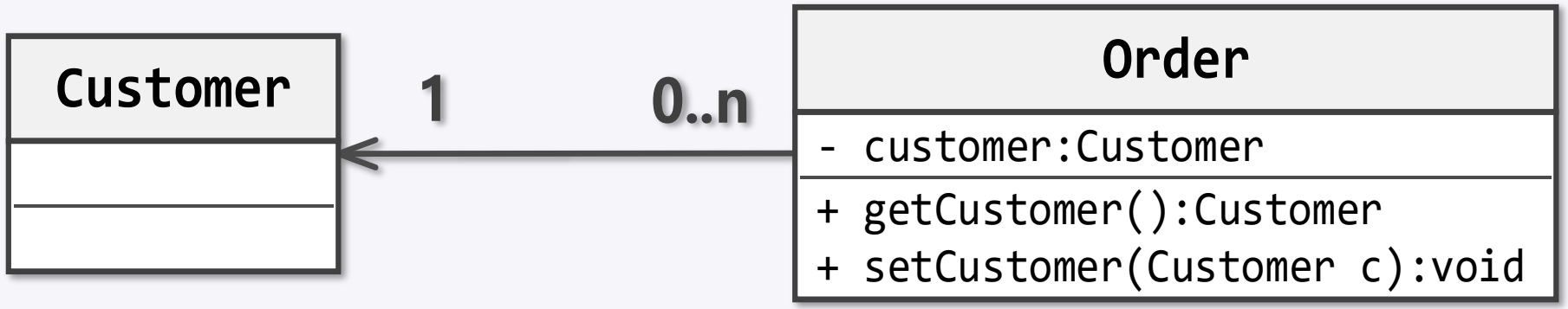
- 域模型是面向对象的，在面向对象术语中，域模型也可称为设计模型。
- 域模型由以下内容组成：
  - 具有状态和行为的域对象；
  - 域对象之间的关系；

域对象

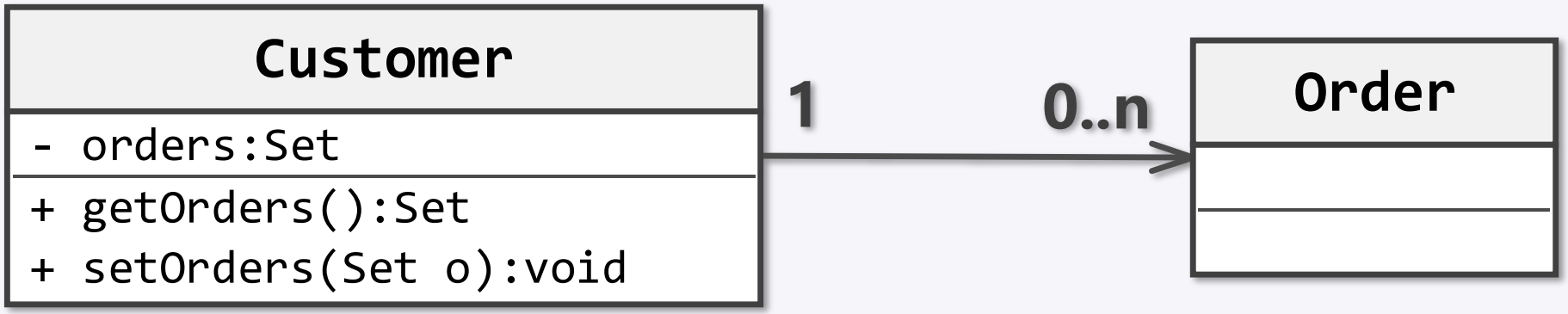
- 域对象可以代表业务领域中的人、地点、事物或概念等。域对象分为以下几种：
  - 实体域对象：业务领域的名词；
  - 过程域对象：业务领域的动词；
  - 事件域对象：业务领域中的事件。

域对象关联关系

- 从Order到Customer的多对一单向关联。

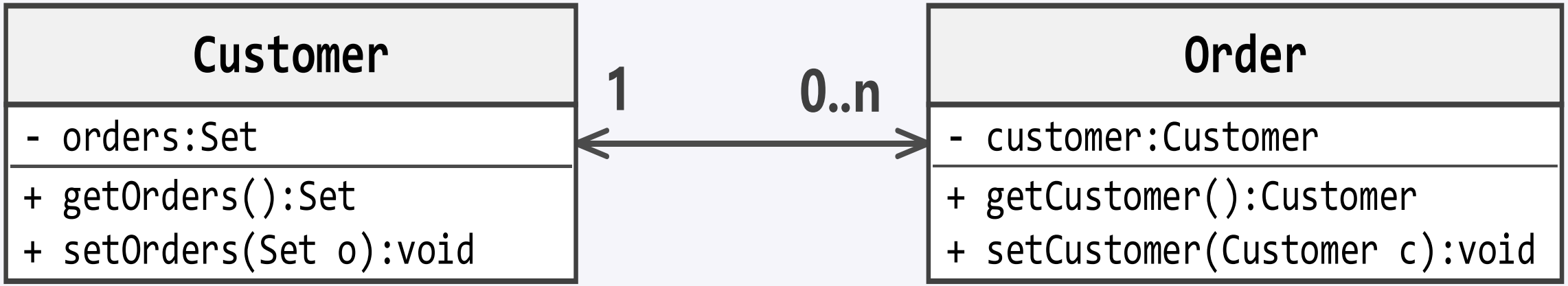


- 从Customer到Order的一对多单向关联。



域对象关联关系

- 从Customer到Order的一对多双向关联。



对象-关系映射

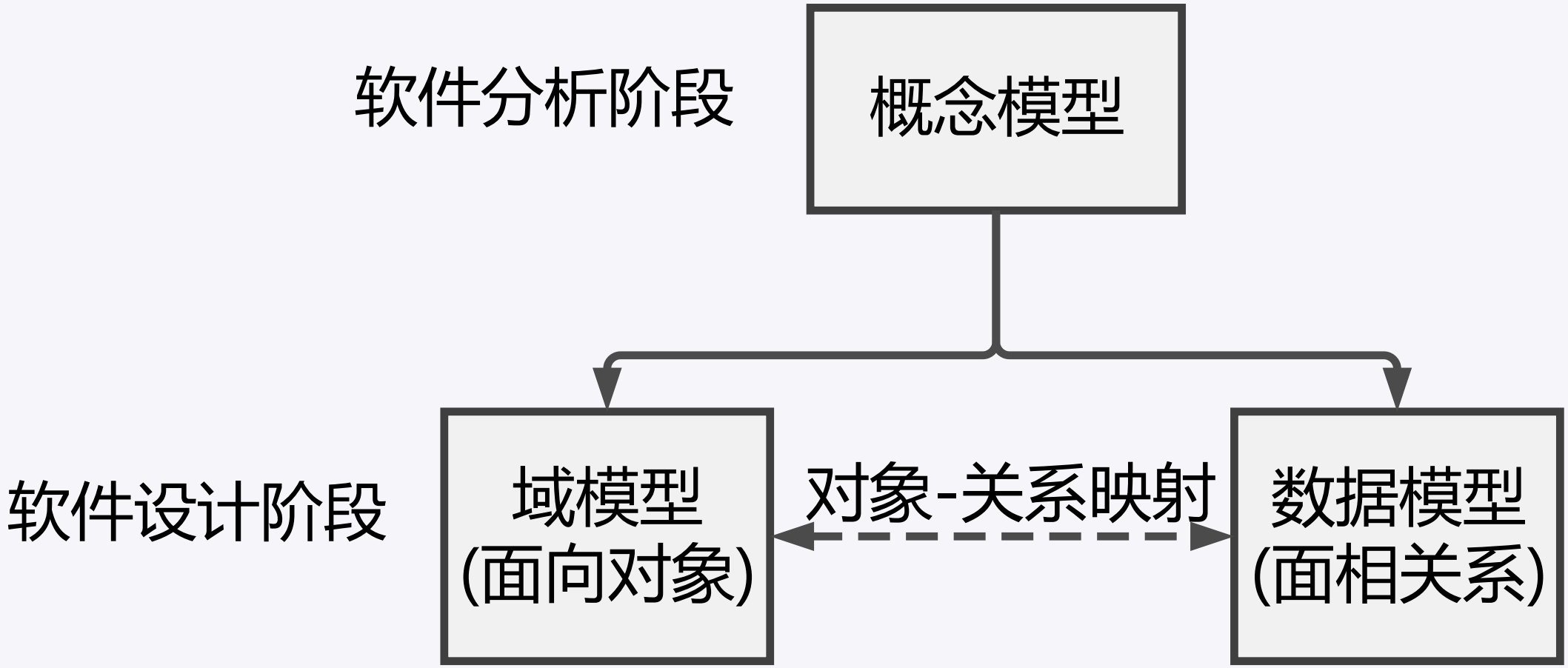
- 一般情况下，一个持久化类和一个表对应，类的每个实例对应表中的一条记录。

面向对象概念	面向关系概念
类	表
对象	记录
属性	列

域模型与关系模型之间存在许多不匹配之处

- 域模型中有继承关系，关系模型不能直接表示继承关系。
- 域模型中有多对多关联关系，关系模型通过连接表来表示多对多关联关系。
- 域模型中有双向关联关系，关系模型只有单向参照关系，而且总是many方参照one方。
- 域模型提倡精粒度模型，而关系模型提倡粗粒度模型。

模型之间的关系

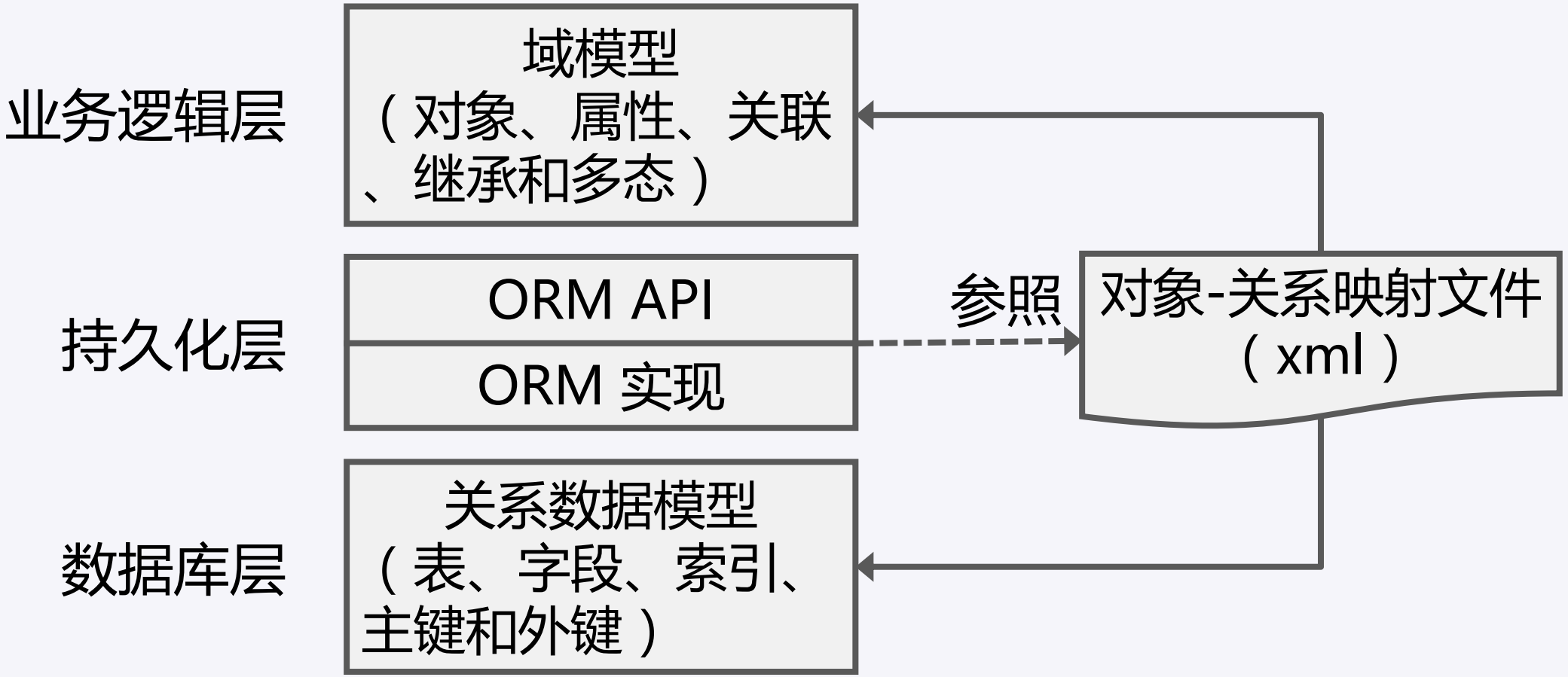


对象-关系映射

- 对象-关系映射（Object Relational Mapping，简称ORM），是随着面向对象的软件开发方法发展而产生的。用来把域模型表示的对象映射到关系数据模型对应的数据库结构中去。
- 通过ORM模式在操作实体对象的时候，就不需要再去和复杂的 SQL语句打交道，只需简单的操作实体对象的属性和方法，ORM 技术是在对象和关系之间提供了一条桥梁，对象型数据和数据库中的关系型的数据通过这个桥梁来相互转化。

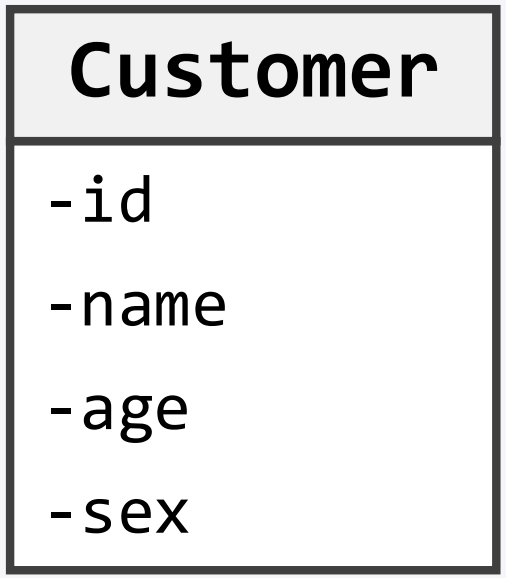
ORM框架

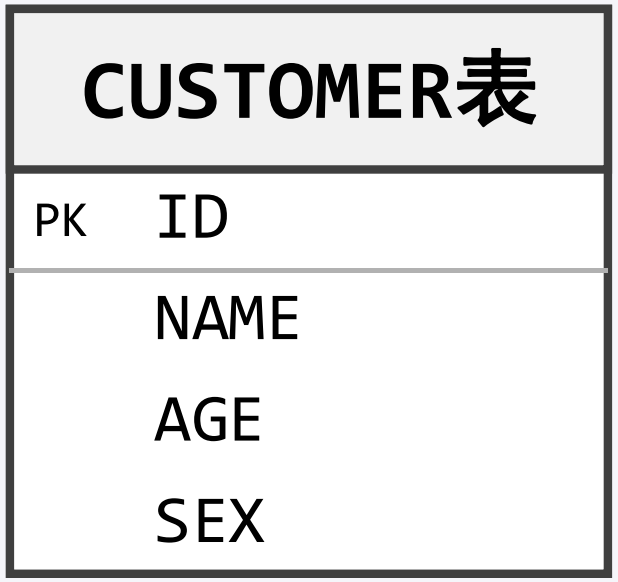
- 通过ORM框架实现
  - Hibernate 开源
  - Mybaits 开源



通过JDBC实现

- 直接通过JDBC API来持久化实体域对象。





通过JDBC实现

- 直接通过JDBC API来持久化实体域对象。

```
public interface PersistenceManger {
    public void saveCustomer(Customer customer);
    public void deleteCustomer(Customer customer);
    public void updateCustomer(Customer customer);
    public Customer loadCustomer(long id);
    public List findCustomerByName(String name);
    public List findCustomerByAge(int age);
    public List findCustomerByNameAndAge(String name,
                                         int age);
    public List findCustomer(String sqlstr);
}
```

通过JDBC实现的缺点

- 业务逻辑和关系数据绑定，如果关系数据模型发生变化，例如修改了表的结构，那么必须手工修改程序代码中所有相关的SQL语句，增加了软件维护的难度。
- 如果程序代码中的SQL语句包含语法错误，在编译时不能检查这种错误，在运行时才能发现这种错误，这增加了调试程序的难度。

通过JDBC实现的缺点

- 持久化层产生大量冗余代码，如下3个查询方法，程序代码都很相似，仅仅是 SQL SELECT 语句中的查询条件不一样。

```
public List findCustomerByName(String name);
public List findCustomerByAge(int age);
public List findCustomerByNameAndAge(String name, int age);
```

```
public List findCustomer(String sqlstr);
```

findCustomer()方法是供业务逻辑层调用，使得业务逻辑层需要了解数据库的访问细节，无法达到软件分层的效果。

目录

1 分层体系结构与持久化

2 软件的模型及ORM

3 **Hibernate介绍**

4 第一个Hibernate程序

Hibernate概念

- Hibernate是什么？
  - 在分层体系结构中Hibernate位于持久层，是完成对象持久化的持久层框架；
  - Hibernate是连接Java应用程序和关系型数据库的框架，能够建立对象模型和关系数据模型之间的映射，是一种自动ORM框架；
  - Hibernate是对JDBC API的封装，是JDBC轻量级封装框架。

Hibernate概念

- Hibernate能带给我们什么？
  - Hibernate实现了ORM，使Java程序员可以方便的运用面向对象的编程思想来操纵关系型数据库；
  - Hibernate是对JDBC的封装，增强了代码的重用性，简化了代码，提高了编程效率；
  - Hibernate是对JDBC的轻量级封装，必要时Java程序员可以绕过Hibernate直接访问JDBC API；
  - Hibernate不仅可以应用在独立的Java程序中，还可以应用在Java Web项目中，可以和多种Web服务器集成，并支持多种数据库平台。

Hibernate发展

- Hibernate发展。

Gavin King

2001年11月
Hibernate1

2003年6月
Hibernate2

2005年3月
Hibernate3

2011年12月
Hibernate4

2015年8月
Hibernate5

2017年4月
Hibernate5.2.10

目录

1 分层体系结构与持久化

2 软件的模型及ORM

3 Hibernate介绍

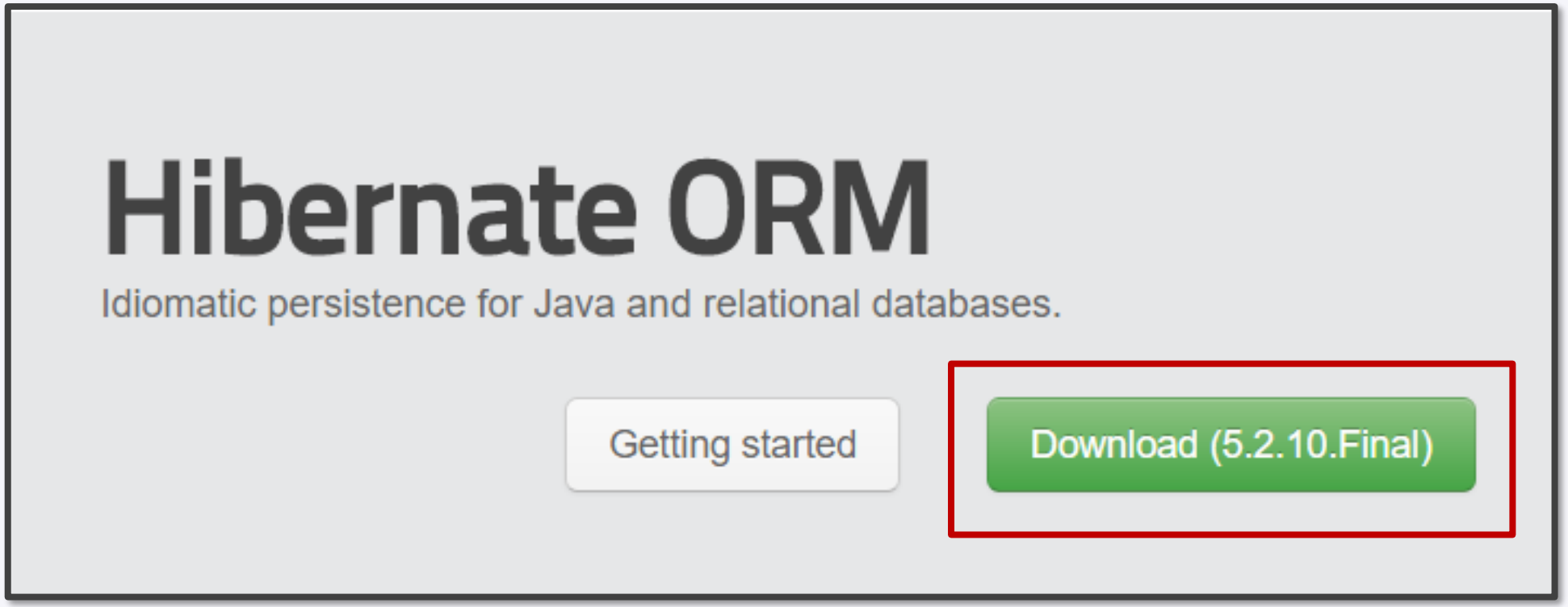
4 **第一个Hibernate程序**

创建Hibernate项目的步骤

0. 安装Eclipse、Mysql；
1. 下载Hibernate，并解压缩；
2. 使用Eclipse创建新的项目；
3. 引入Hibernate及其依赖库（jar包）；
4. 引入Mysql数据库驱动包；
5. 编写Hibernate配置文件；
6. 创建Java持久化类XXX.java；
7. 编写持久化类的映射配置文件XXX.hbm.xml；
8. 使用Hibernate API 完成对象的持久化。

Hibernate下载

- Hibernate官网：http://hibernate.org/orm/



Eclipse中创建项目并引入jar

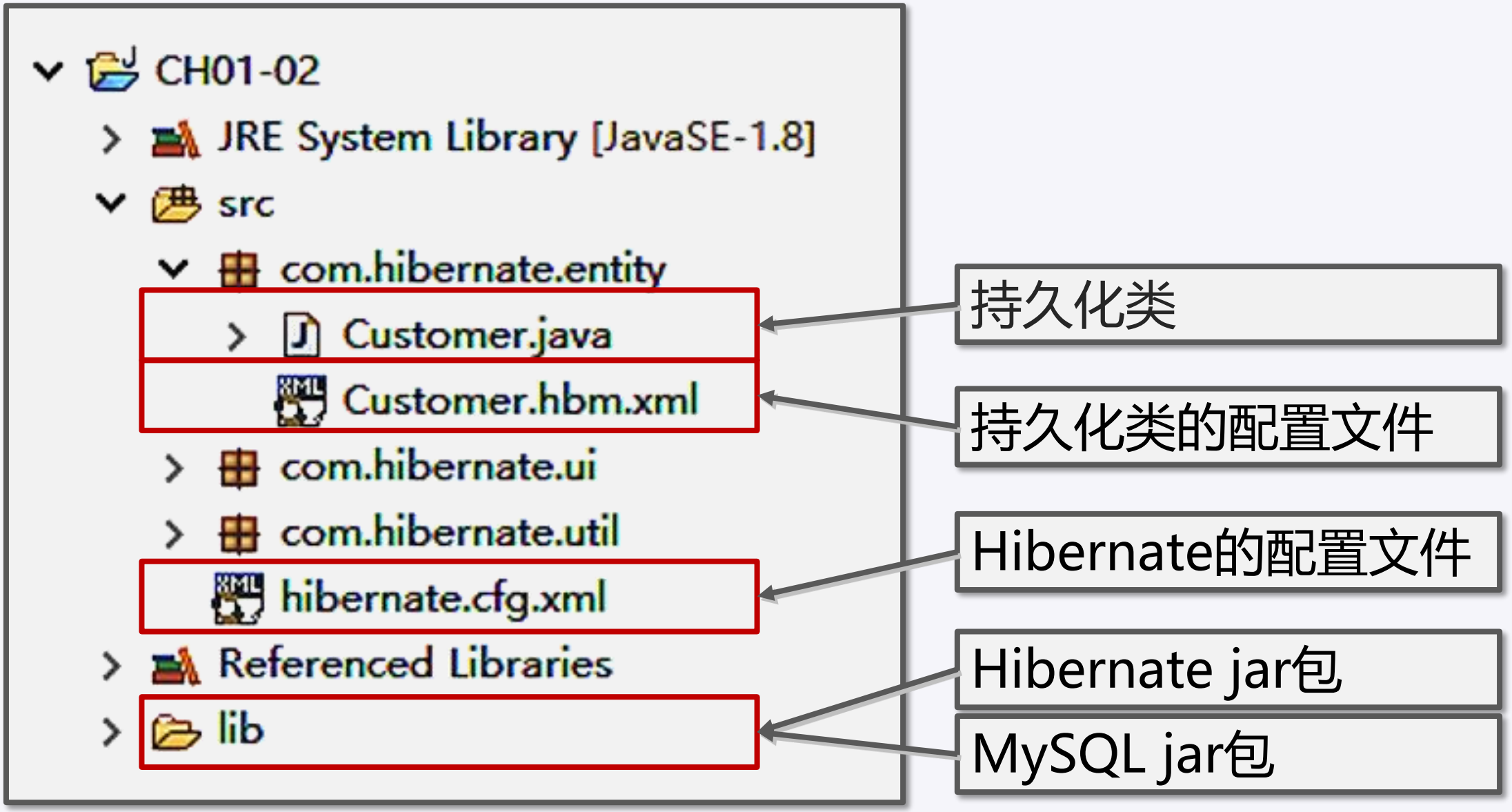
MySQL

- mysql-connector-java-5.1.42-bin.jar

Hibernate

- antlr-2.7.7.jar
- classmate-1.3.0.jar
- dom4j-1.6.1.jar
- hibernate-commons-annotations-5.0.1.Final.jar
- hibernate-core-5.2.10.Final.jar
- hibernate-jpa-2.1-api-1.0.0.Final.jar
- jandex-2.0.3.Final.jar
- javassist-3.20.0-GA.jar
- jboss-logging-3.3.0.Final.jar
- jboss-transaction-api_1.2_spec-1.0.1.Final.jar

项目目录结构说明


CH01-02
JRE System Library [JavaSE-1.8]
src
com.hibernate.entity
Customer.java
Customer.hbm.xml
com.hibernate.ui
com.hibernate.util
hibernate.cfg.xml
Referenced Libraries
lib
持久化类
持久化类的配置文件
Hibernate的配置文件
Hibernate jar包
MySQL jar包

Hibernate配置文件

- Hibernate需要从配置文件中读取数据库配置信息，配置文件一般位于项目根路径。
- Hibernate配置文件两种方式：
  - hibernate.properties（键=值方式）

    默认名字为：hibernate.propeties
  - hibernate.cfg.xml

Hibernate配置文件

- hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/MyDB</property>
  <property name="hibernate.connection.username">root</property>
  <property name="hibernate.connection.password">123456</property>
  <property name="hibernate.show_sql">true</property>
  <property name="hibernate.format_sql">true</property>
  <mapping resource="com/hibernate/entity/Customer.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

创建持久化类

- 持久化类：指其实例需要被Hibernate持久化到数据库中的类即实体类
  - private 类型属性；
  - public 类型的 setter 和 getter 方法；
  - public 或 protected 类型的无参数的构造方法。

创建持久化类

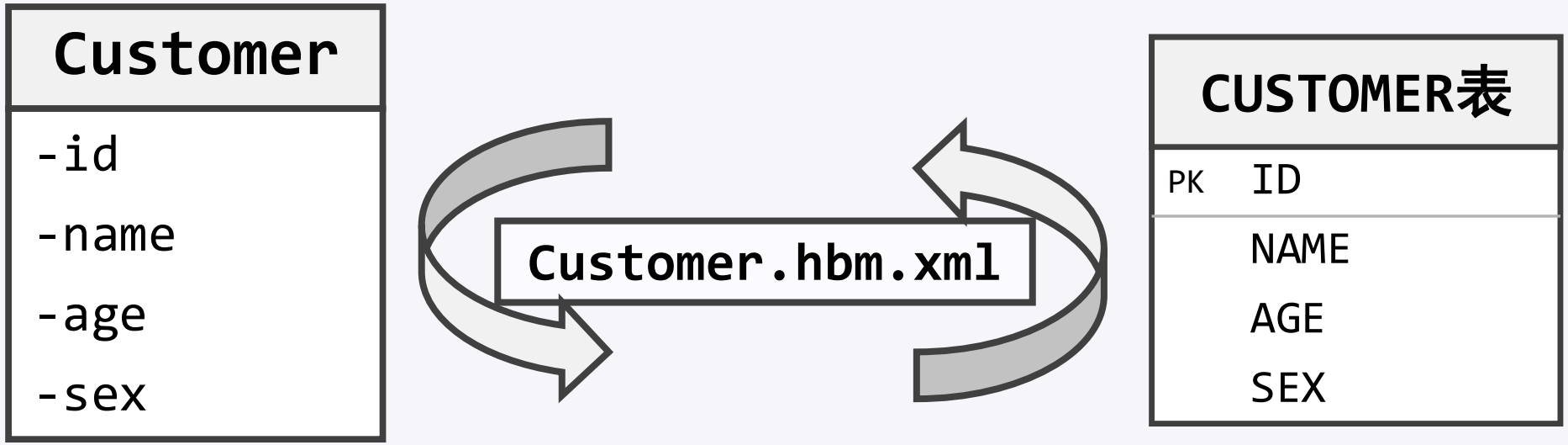
```
public class Customer {
    private int id;
    private String name;
    ...

    public int getId() {...}
    public void setId(int id) {....}
    public String getName() {...}
    public void setName(String name) {...}
    ...
}
```

Customer
-id
-name
-age
-sex

创建持久化类的配置文件

- 创建持久化类的配置文件。
  - 描述持久化类与数据库表之间的对应关系。



创建持久化类的配置文件

- Student.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.hibernate.entity">
    <class name="Customer" table="CUSTOMER">
        <id name="id" type="int" >
            ...
        </id>
        <property name="name" type="java.lang.String">
            ...
        </property>
        ...
    </class>
</hibernate-mapping>
```

id 和 property 元素不能颠倒位置

创建持久化类的配置文件

- <class>元素用于指定类和表之间的映射。
  - name属性设定类名(包含路径)；
  - table属性设定表名，默认以类名作表名。

```
<class name="Customer" table="CUSTOMER">
    ...
</class>
```

- <class>元素包含一个<id>子元素及多个<property>子元素。

创建持久化类的配置文件

- **id** 子元素设定持久化类的OID和表的主键的映射关系。
  - column – 指定表字段的名称；
  - generator – 元素指定OID的生成器。

```
<id name="id" type="int">
    <column name="ID"/>
    <generator class="native"/>
</id>
```

创建持久化类的配置文件

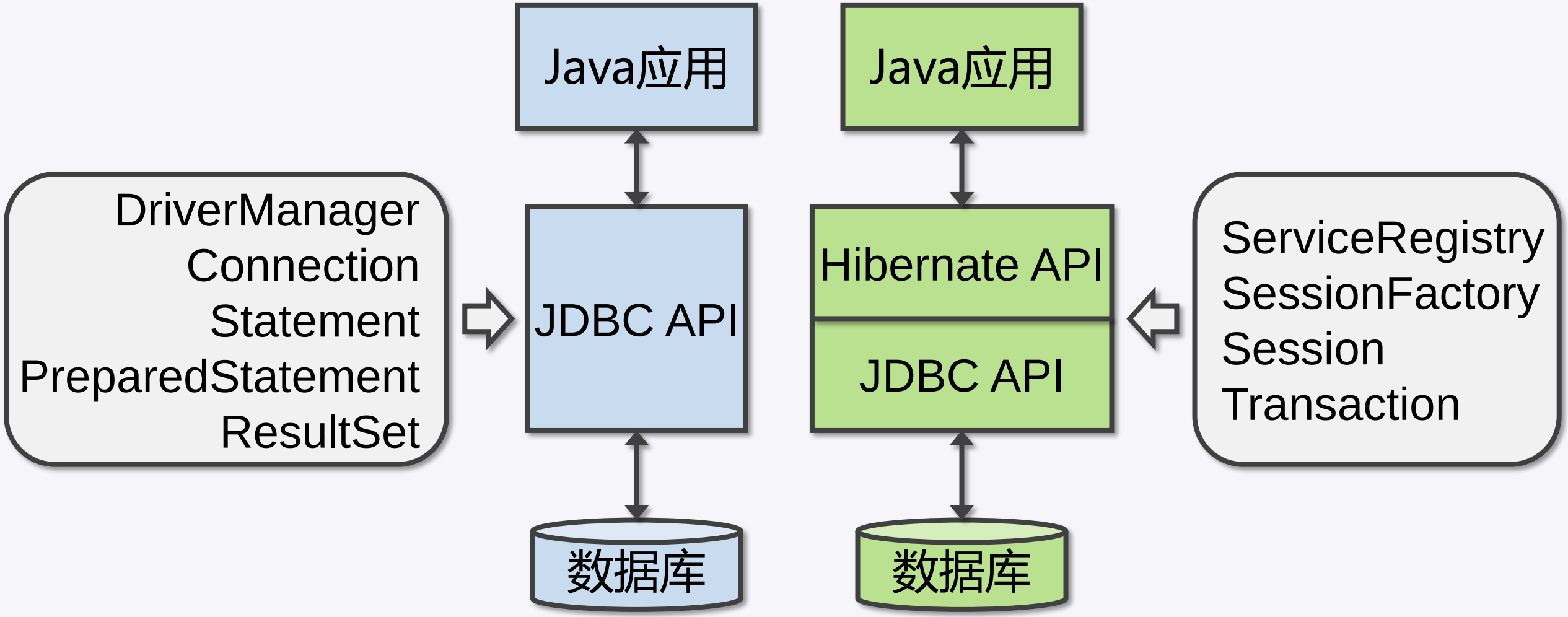
- **property** 子元素设定类的其他属性和表的字段的映射关系。
  - name – 对应类的属性名称；
  - type – 指定属性的类型；
  - column – 指定表字段的名称；
  - not-null – 指定属性是否允许为空。

```
<property name="name" not-null="true"
          type="java.lang.String">
    <column name="NAME"/>
</property>
```

创建Hibernate项目的步骤

1. 下载Hibernate，并解压缩；
2. 使用Eclipse创建新的项目；
3. 引入Hibernate及其依赖库（jar包）；
4. 引入mysql数据库驱动包；
5. 编写Hibernate配置文件；
6. 创建Java持久化类XXX.java；
7. 编写持久化类的映射配置文件XXX.hbm.xml；
8. 使用Hibernate API 完成对象的持久化。

通过Hibernate API 操纵数据库



ServiceRegistry接口

- ServiceRegistry 注册服务的创建。

```
StandardServiceRegistry registry
     = new StandardServiceRegistryBuilder()
          .configure().build();
// 从 hibernate.cfg.xml 读取配置信息
```

SessionFactory接口

- SessionFactory 会话工厂的作用。
  - 缓存Hibernate配置信息和映射元数据信息；
  - 负责创建Session实例。
- SessionFactory的创建。

```
SessionFactory sessionFactory
    = new MetadataSources(registry).buildMetadata()
        .buildSessionFactory();
```

- SessionFactory是线程安全的，多个应用线程间进行共享，一般整个应用只有唯一的一个SessionFactory实例。

SessionFactory接口

```
final StandardServiceRegistry registry
    = new StandardServiceRegistryBuilder()
        .configure() // 从hibernate.cfg.xml读取配置信息
        .build();    // 创建StandardServiceRegistry
SessionFactory sessionFactory = null;
try {
    sessionFactory = new MetadataSources(registry)
        .buildMetadata().buildSessionFactory();
} catch (Exception e) {
    // 创建失败手动释放
    StandardServiceRegistryBuilder.destroy(registry);
}
```

Session接口

- Session是Hibernate持久操作的基础核心。
  - Hibernate Session 与Hibernate相当于JDBC Connection与JDBC；
  - 它代表与数据库之间的一次连续操作；
  - Session负责执行访问数据库的操作，比如保存、更新、删除、加载和查询，也称为持久化管理器。
- Session的创建。

```
Session session = sessionFactory.openSession();
```

- Session是一个轻量级对象，是非线程安全的，通常和一个数据库事务绑定。

Session接口

```
Session session = sessionFactory.openSession();
// 开始一次事务
Transaction tx = session.beginTransaction();

// 创建需要持久化的对象
Customer c1 = new Customer();
c1.setName("zhangsan");
c1.setSex(1);
c1.setAge(20);
session.save(c1); // 将对象持久化
tx.commit();      // 提交事务
session.close();  // 关闭会话
```

本章小结

- 分层体系结构与持久化
- 软件的模型及ORM
- Hibernate是什么？
- Hibernate项目的创建过程
  - 总体配置文件、持久化类的配置文件
  - Hibernate API的使用

练习

- 创建项目，搭建 Hibernate 开发环境。

THANK YOU