



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

HIBERNATE

第二讲 Hibernate单实体映射



BIG
DATA

Java与大数据分析



Java与移动智能设备开发



- 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



- 1 单实体映射基础**
- 2 单实体的属性映射
- 3 单实体的对象标识符映射
- 4 使用注解映射单实体

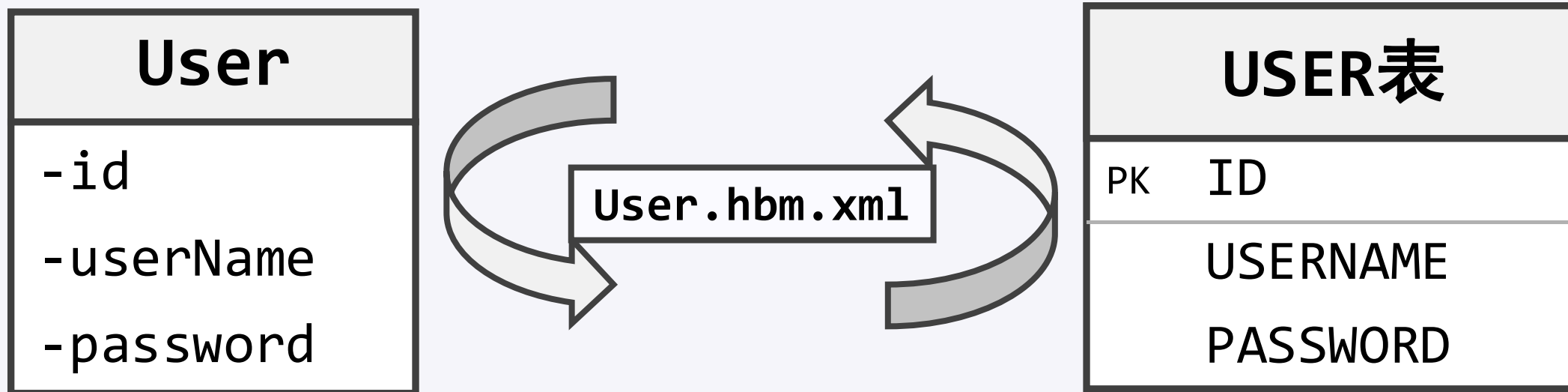


- 持久化类：指其实例需要被Hibernate持久化到数据库中的类即实体类。
 - private 类型属性；
 - public 类型的 setter 和 getter 方法；
 - public 或 protected 类型的无参数的构造方法。

创建持久化类的配置文件



- 描述持久化类与数据库表之间的对应关系。



持久化类的配置文件示例



■ User.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="User" table="USER">
    <id name="id" type="int" >
      <generator class="increment"/>
    </id>
    <property name="userName" type="java.lang.String" />
    <property name="password" type="java.lang.String" />
  </class>
</hibernate-mapping>
```

id 和 property
元素不能颠倒位置



■ **<class>** 元素用于指定类和表之间的映射。

➤ name - 设定类名(包含路径)；

➤ table - 设定表名，默认以类名作表名。

```
<class name="User" table="USER">
    ...
</class>
```

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



- **<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>
```




■ **<property>** 子元素设定类的其它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="userName" not-null="true"  
          type="java.lang.String">  
    <column name="NAME"/>  
</property>
```



1 单实体映射基础

2 单实体的属性映射

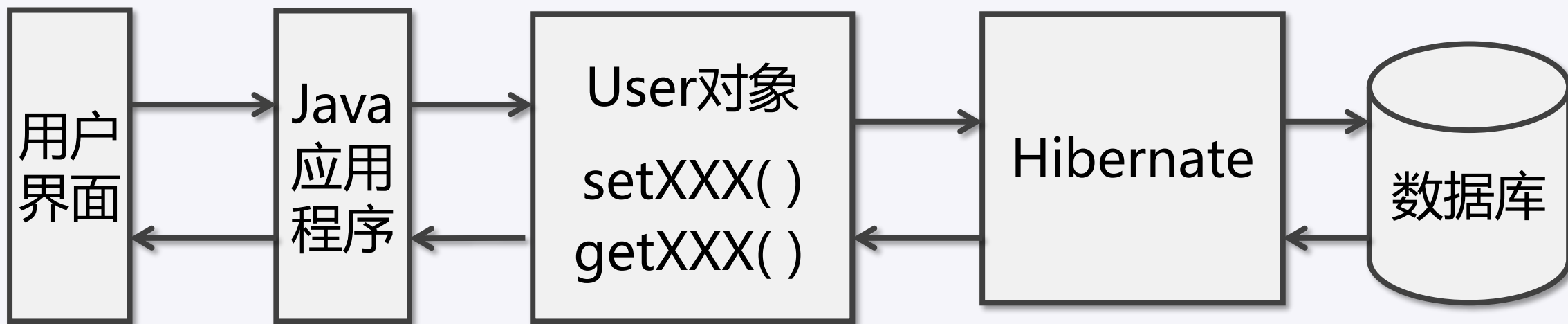
3 单实体的对象标识符映射

4 使用注解映射单实体

持久化类属性及访问方法



- 持久化类采用 JavaBean 风格，为被访问的属性创建 setter 和 getter 方法，这两个方法被称为持久化类的访问方法。
- setter 和 getter 方法优点：有效控制属性的访问权限。





- 在对象-关系映射文件中<property>元素的 access 属性用于指定Hibernate访问持久化类属性的方式。
 - property : 默认值，通过getter和setter方法访问属性值；
 - field : 通过Java反射机制直接访问属性值。



- 持久化类属性没有 setter 和 getter 方法的映射。
- 持久化类属性与数据库表字段不对称的映射。



- 缺少 setter 和 getter 方法的实体类在映射时，可将 `<property>` 元素的 `access` 属性设置为 `field`。
 - 例如：User 类的 `username` 属性没有 setter 和 getter 方法。

```
<property name="userName" access="field" />
```



- 实体类属性与表字段不对称时，可以在 setter 和 getter 方法中加入程序逻辑。
 - 例如：User 类中没有 username 属性，而是改为了 firstName 和 lastName 两个属性。

User
-id
-firstName
-lastName
-password

USER表	
PK	ID
USERNAME	
PASSWORD	



```
public String getUser_name() {  
    return firstName + " " + lastName;  
}  
  
public void setUser_name(String name) {  
    String[] strName = userName.split(" ");  
    this.firstName = strName[0];  
    this.lastName = strName[1];  
}
```




■ 特殊需求

- 例如：User 需要订单总额属性，但数据库表中没有这个字段。
- 可在<property>元素的 formula 属性设置查询语句。

```
<property name="totalPrice"  
          formula="(select sum(o.price) from  
                    orders as o where o.userId=id)"/>
```



- Hibernate 在初始化阶段就会根据映射配置文件，为持久化类生成以下SQL语句：
 - INSERT SQL ;
 - UPDATE SQL ;
 - DELETE SQL ;
 - 根据ID检索持久化类实例 SQL。

控制持久化类的insert和update



映射属性	作用
<property>元素的insert 属性	insert语句中是否包含该属性，默认值为true
<property>元素的update 属性	update语句中是否包含该属性，默认值为true
<class>元素的mutable 属性	等价于所有property节点的update属性，默认值为true
<class>元素的dynamic-insert 属性	值为true等价于所有的property元素dynamic-insert属性为true
<class>元素的dynamic-update 属性	值为true等价于所有的property元素dynamic-update属性为true



- 1 单实体映射基础
- 2 单实体的属性映射
- 3 单实体的对象标识符映射**
- 4 使用注解映射单实体



- 关系型数据库中区分不同记录。
 - 数据库中用主键来标识记录并保证记录的唯一性。
 - 主键必须满足的条件：
 - 不允许null；
 - 每条记录必须有唯一的主键值，主键值不能重复；
 - 每条记录的主键值不能改变。
 - 主键分类：
 - 业务（自然）主键：具有实际意义；
 - 代理主键：没有实际意义。



■ Java程序中区分不同对象。

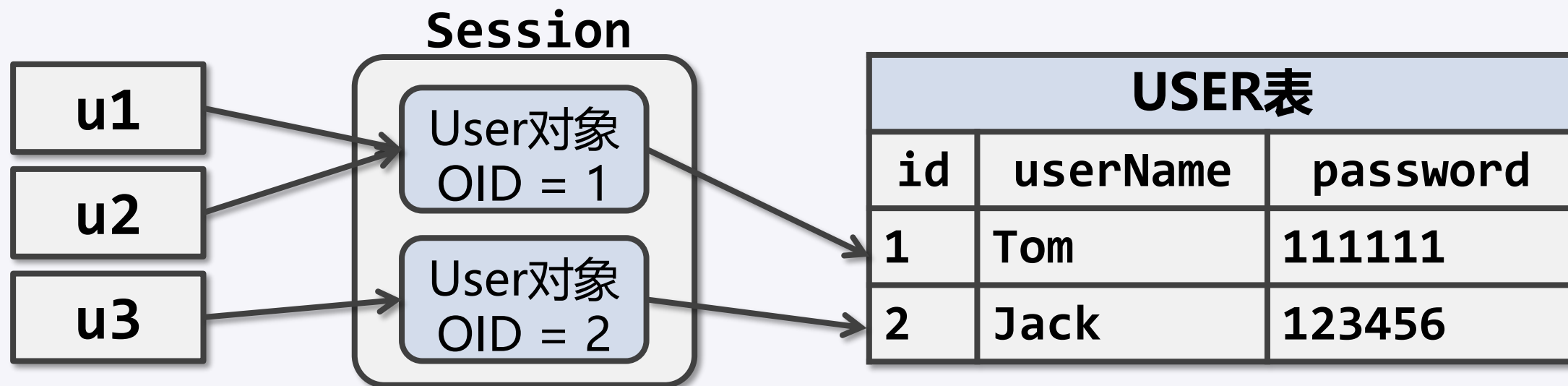
- Java语言中通过内存地址区分不同对象；
- 两种比较引用变量方法；
 - "==" 比较两个变量引用的内存地址是否相同；
 - equals 比较两个变量引用的对象的值是否相同。
- 用户自定义的类也可以覆盖Object的equals方法实现对对象按值进行比较。



- Hibernate 采用对象标识符（OID）区分对象。
 - OID 是关系数据库中主键（通常是代理主键）在 Java 对象模型中的等价物；
 - Hibernate 采用 OID 来维持java对象和数据库表中对应关系。



```
User u1= (User) session.get(User.class, new Integer(1));
User u2= (User) session.get(User.class, new Integer(1));
User u3= (User) session.get(User.class, new Integer(2));
System.out.println(u1 == u2); //true
System.out.println(u1 == u3); //false
```





- OID 与表中代理主键对应，OID 也是整数类型，Hibernate 允许在持久化类中把OID定义为以下三种类型：
 - Short
 - Integer
 - Long
- 为了保证 OID 的唯一性，通常由 Hibernate 或底层数据库给 OID 赋值。



- 在对象-关系映射配置文件中<class>元素的<id>子元素用来设置 OID。
 - <generator>子元素用来指定OID的生成器。

```
<class name="User" table="USER">  
  <id name="id" type="int" >  
    <generator class="identity" />  
  </id>  
  .....  
</class>
```



- Hibernate 自带了很多种标识符生成器：
 - **increment** 采用 Hibernate 数值递增的方式；
 - **identity** 采用数据库提供的自增长方式；
 - **assigned** 主键由应用逻辑产生；
 - **sequence** 采用数据库提供的序列方式；
 - **hilo** 通过hi/lo算法 // Hibernate 5.0 以后不支持；
 - **seqhilo** 通过hi/lo算法；
 - **native** 自动选择合适的标识符生成器；
 - **uuid.hex** 通过uuid算法。



■ **increment** 标识符：该机制是 Hibernate 以递增的方式为 OID赋值。

- 不依赖于底层数据库系统，适合所有数据库;
- 适合单独的 Hibernate 应用使用，不适合在集群情况下使用。

```
<id name="id" type="int" >  
    <generator class="increment" />  
</id>
```



- **identity** 标识符：该机制依赖于底层数据库，需要数据库支持自动增长字段。

➤ 例如：MySQL、MSSQL、DB2、Sybase等。

```
<id name="id" type="int" >  
    <generator class="identity" />  
</id>
```



- **assigned** 标识符：该机制是由外部程序负责生成 OID，Hibernate 不负责维护主键生成，与Hibernate和底层数据库都无关。
 - 例如：Student 类没有定义 ID，而是以学号 studentNo 作为业务主键。

```
<id name="studentNo" type="string" >  
    <generator class="assigned" />  
</id>
```



- 1 单实体映射基础
- 2 单实体的属性映射
- 3 单实体的对象标识符映射
- 4 使用注解映射单实体**



- `@Entity` : 声明一个实体类。
- `@Table(name="table_name")` : 为实体类指定对应的数据库表。
- `@Id` : 声明实体类的OID属性。
- `@GeneratedValue(generator="increment_generator")`
: 声明OID的生成策略。
- `@GenericGenerator(name="increment_generator",
strategy="increment")` : 使用Hibernate提供的生成策略。



- `@Column(name="columnName")` : 将属性映射到列。
 - `name="columnName"` 字段名称 ;
 - `unique=false` 是否在该字段上设置唯一约束 ;
 - `nullable=true` 字段是否能为空 ;
 - `insertable=true` 控制 insert 语句 ;
 - `updatable=true` 控制 update 语句 ;
 - `length=255` 指定字段长度。



- `@Access(AccessType.PROPERTY)` :
 - 通过 getter 和 setter 方法访问实体类的属性；
 - 需要在 getter 方法上定义字段的属性。
- `@Access(AccessType.FIELD)` :
 - 直接访问实体类的属性，可以不定义 getter 和 setter 方法；
 - 需要在变量上定义字段的属性。



- @Formula : 将属性映射到SQL语句。

```
@Formula(value = "(select sum(o.price) from  
orders as o where o.userid=id)")
```

- @DynamicInsert : 动态生成 INSERT 语句。
- @DynamicUpdate : 动态生成 UPDATE 语句。



- 单实体映射基础
- 单实体的属性映射
- 单实体的对象标识符映射
- 使用注解映射单实体

练习





THANK YOU
