

Hibernate5笔记

hibernate的三态

hibernate由游离态，持久态，瞬时态。

游离态：session中没有，数据库中有此对象

持久态：session中有，数据库中也有

瞬时态：对象刚刚创建，session中没有，数据库中也没有。

补充（mysql是关系型数据库，存储与硬盘中。读写速度必定低于h2等内存数据库。除了关系型数据库还有nosql数据库，如redis,mongodb,直接存储对象）

持久化，序列化

持久化概念是指将内存中的数据以文件的形式存储在文件中。如，一个应用程序中的数据存储在数据库中，excel,txt等文件都可以作为数据的持久化存储文件。

序列化是指将对象以流的方式进行存储、传输。具有序列化id唯一可以进行反序列化。

ORM概念

orm是指对象关系映射技术（Object Relation Mapping）,并不只是hibernate所具有的特殊功能。是一种泛指的概念。hibernate是全自动的orm框架，类似的半自动化框架还有mybatis。jpa是Java Persistence API的简称中文名Java持久层API，是JDK 5.0注解或XML描述对象 - 关系表的映射关系，并将运行期的实体对象持久化到数据库中。

PA的总体思想和现有Hibernate、TopLink、JDO等ORM框架大体一致。总的来说，JPA包括以下3方面的技术：

ORM映射元数据

JPA支持XML和[JDK5.0](#)注解两种元数据的形式，元数据描述对象和表之间的映射关系，框架据此将实体[对象持久化](#)到数据库表中；

API

用来操作实体对象，执行CRUD操作，框架在后台替代我们完成所有的事情，开发者从繁琐的JDBC和SQL代码中解脱出来。

查询语言

这是持久化操作中很重要的一个方面，通过[面向对象](#)而非面向数据库的查询语言查询数据，避免程序的SQL语句紧密耦合。

EJB概念

EJB是sun的JavaEE服务器端[组件模型](#)，设计目标与核心应用是部署分布式应用程序。简单来说就是把已经编写好的程序（即：类）打包放在服务器上执行。凭借java跨平台的优势，用EJB技术部署的[分布式系统](#)可以不限于特定的平台。EJB (Enterprise [javaBean](#))是J2EE(javaEE)的一部分，定义了一个用于开发基于组件的企业多重应用程序的标准。其特点包括[网络服务](#)中心支持和核心开发工具(SDK)。在J2EE里，Enterprise Java Beans(EJB)称为Java 企业Bean，是Java的核心代码，分别是会话Bean（Session Bean），实体Bean（Entity Bean）和消息驱动Bean（MessageDriven Bean）。在EJB3.0推出以后，实体Bean被单独分了出来，形成了新的规范[JPA](#)。

Hibernate的get()与load()的区别

```

    @Test
    public void getAndLoad() {
        Configuration configuration=new Configuration().configure();
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction transaction = session.beginTransaction();
        GoodsEntity goodsEntity = session.get(GoodsEntity.class, id: 31);
        System.out.println("*****");
        System.out.println("==="+goodsEntity);
    }
}
HibernateTest > getAndLoad()
HibernateTest.getAndLoad
Tests passed: 1 of 1 test - 2 s 534 ms
```

```

goodsentit0_.id as id1_0_0_,
goodsentit0_.goods_name as goods_na2_0_0_,
goodsentit0_.goods_price as goods_pr3_0_0_
from
  goods goodsentit0_
where
  goodsentit0_.id=?
*****
==null
```

当使用get()查询的数据不存在时，会正常执行，展示null值。

```

    @Test
    public void getAndLoad() {
        Configuration configuration=new Configuration().configure();
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction transaction = session.beginTransaction();
        GoodsEntity goodsEntity = session.load(GoodsEntity.class, id: 31);
        System.out.println("*****");
        System.out.println("==="+goodsEntity);
    }
}
HibernateTest > getAndLoad()
HibernateTest.getAndLoad
Tests failed: 1 of 1 test - 2 s 584 ms
```

```

org.hibernate.ObjectNotFoundException: No row with the given identifier exists: [com.h
<6 internal calls>
  at com.hibernate.pojo.GoodsEntity$HibernateProxy$N94NiDhf.toString(Unknown Source)
  at java.lang.String.valueOf(String.java:2994)
  at java.lang.StringBuilder.append(StringBuilder.java:131)
  at HibernateTest.getAndLoad(HibernateTest.java:64) <22 internal calls>
```

rocess finished with exit code -1

当使用load()进行查询的数据不存在时，会抛出异常，因此，不建议使用load方法。

lazy的使用

```
57 public void getAndLoad() {
58     Configuration configuration=new Configuration().configure();
59     SessionFactory sessionFactory = configuration.buildSessionFactory();
60     Session session = sessionFactory.openSession();
61     Transaction transaction = session.beginTransaction();
62     GoodsEntity goodsEntity = session.get(GoodsEntity.class, id: 61);
63     System.out.println("*****");
64     System.out.println("===="+goodsEntity.getGoodsName());
65 }
```

七月 27, 2019 11:32:30 上午 org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version 4.2
Hibernate:
select
goodsentit0_.id as id1_0_0_,
goodsentit0_.goods_name as goods_na2_0_0_,
goodsentit0_.goods_price as goods_pr3_0_0_
from
goods goodsentit0_
where
goodsentit0_.id=?

====营养快线

使用get()方法获取数据时, lazy默认为false, 因此在箭头处直接执行sql语句进行查询

```
7 public void getAndLoad() {
8     Configuration configuration=new Configuration().configure();
9     SessionFactory sessionFactory = configuration.buildSessionFactory();
10    Session session = sessionFactory.openSession();
11    Transaction transaction = session.beginTransaction();
12    GoodsEntity goodsEntity = session.load(GoodsEntity.class, id: 61);
13    System.out.println("*****");
14    System.out.println("===="+goodsEntity.getGoodsName());
15 }
```

七月 27, 2019 11:38:29 上午 org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version 4.2

Hibernate:
select
goodsentit0_.id as id1_0_0_,
goodsentit0_.goods_name as goods_na2_0_0_,
goodsentit0_.goods_price as goods_pr3_0_0_
from
goods goodsentit0_
where
goodsentit0_.id=?
====营养快线

使用load()方法进行获取时, lazy默认为true, 会在使用时才执行sql查询, 进箭头处才会执行sql。

lazy属性时hibernate的一种优化策略, 在必要时会节省数据库连接资源的开销。

lazy有三个属性: true、false、extra

【true】:默认取值，它的意思是只有在调用这个集合获取里面的元素对象时，才发出查询语句，加载其集合元素的数据

【false】:取消懒加载特性，即在加载对象的同时，就发出第二条查询语句加载其关联集合的数据

【extra】:一种比较聪明的懒加载策略，即调用集合的size/contains等方法的时候，hibernate并不会去加载整个集合的数据，而是发出一条聪明的SQL语句，以便获得需要的值，只有在真正需要用到这些集合元素对象数据的时候，才去发出查询语句加载所有对象的数据。

关联关系的几种方式

1.many-to-one

我们在多的一方配置单向的

```
<!--多的一方配置的属性与数据库的外键进行关系映射-->
<!--name为实体中的关系，column为数据库中的参考外键-->
<many-to-one name="people" column="pid"></many-to-one>
```

然后再代码中进行保存测试

```
People people=new People();
people.setName("小王");
people.setYear(201);
GoodsEntity goodsEntity=new GoodsEntity();
goodsEntity.setGoodsName("瓜子");
goodsEntity.setGoodsPrice(33);
goodsEntity.setPeople(people);
session.save(goodsEntity);
transaction.commit();
```

按照逻辑应当先保存people，当不先进行people的保存时，运行程序会抛出如下异常

```
java.lang.IllegalStateException: org.hibernate.TransientObjectException: object references an
unsaved transient instance - save the transient instance before flushing:
com.hibernate.pojo.People

Caused by: org.hibernate.TransientObjectException: object references an unsaved transient
instance - save the transient instance before flushing: com.hibernate.pojo.People
    at
org.hibernate.engine.internal.ForeignKeys.getEntityIdentifierIfNotUnsaved(ForeignKeys.java:350)
    at org.hibernate.type.EntityType.getIdentifier(EntityType.java:495)
    at org.hibernate.type.ManyToOneType.isDirty(ManyToOneType.java:332)
    at org.hibernate.type.ManyToOneType.isDirty(ManyToOneType.java:343)
    at org.hibernate.type.TypeHelper.findDirty(TypeHelper.java:315)

    at org.hibernate.internal.SessionImpl.doFlush(SessionImpl.java:1454)
    ... 31 more
```

出现此种情况，我们再置文件中写上cascade属性设置为级联便可

```
<!--多的一方配置的属性与数据库的外键进行关系映射-->
```

```
<many-to-one name="people" column="pid" cascade="save-update"></many-to-one>
```

默认会先执行主表的插入操作，然后进行从表的插入操作。

2.one-to-money

再一的一方中给实体类添加set集合，再一的配置文件中添加如下标签

```
<set name="goodsEntities"><!-- 实体类中的属性名称-->
```

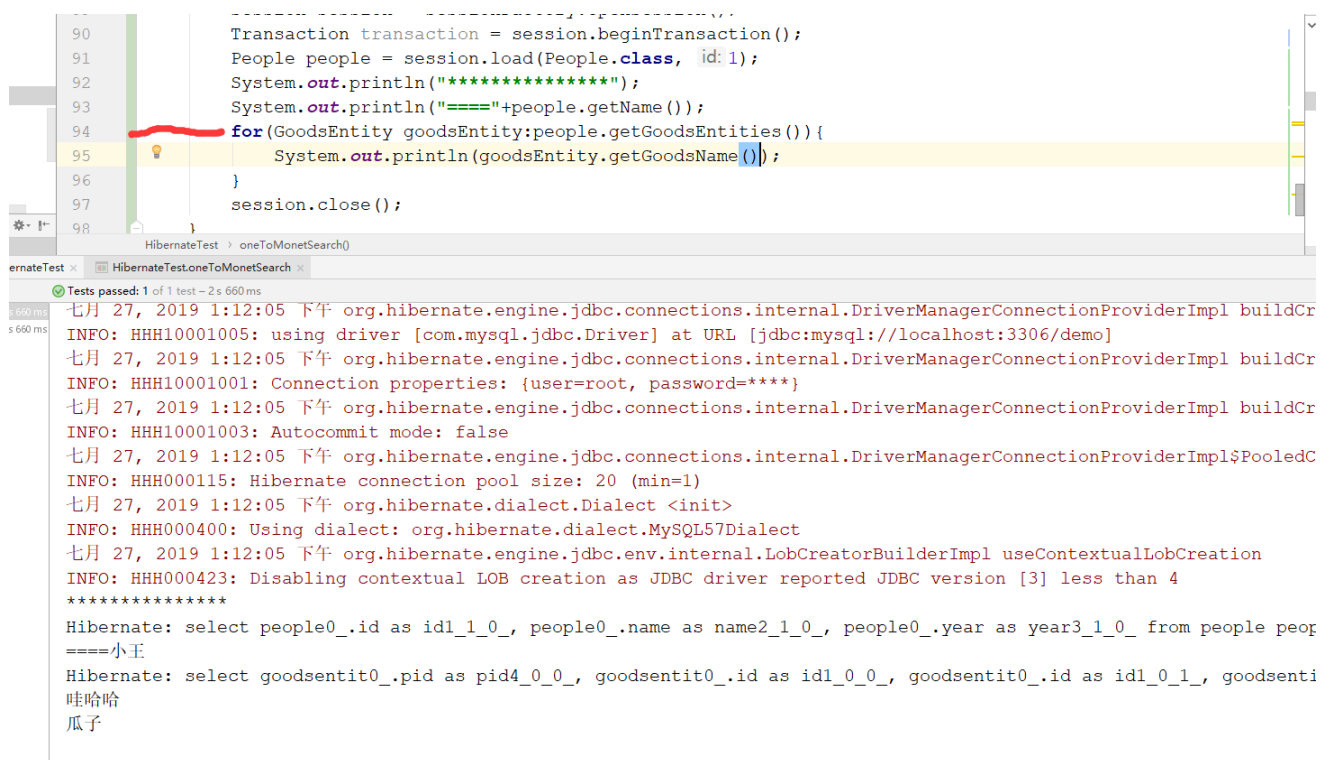
```
<key>
```

```
<column name="pid"></column><!--关联的外键的列名-->
```

```
</key>
```

```
<one-to-many class="com.hibernate.pojo.GoodsEntity"/><!--关联的实体类（多的一方）-->
```

```
</set>
```



```
90 Transaction transaction = session.beginTransaction();
91 People people = session.load(People.class, id: 1);
92 System.out.println("*****");
93 System.out.println("====="+people.getName());
94 for(GoodsEntity goodsEntity:people.getGoodsEntities()){
95     System.out.println(goodsEntity.getGoodsName());
96 }
97 session.close();
98 }
```

Tests passed: 1 of 1 test - 2 s 660 ms

七月 27, 2019 1:12:05 下午 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCr
INFO: HHH10001005: using driver [com.mysql.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/demo]

七月 27, 2019 1:12:05 下午 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCr
INFO: HHH10001001: Connection properties: {user=root, password=****}

七月 27, 2019 1:12:05 下午 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCr
INFO: HHH10001003: Autocommit mode: false

七月 27, 2019 1:12:05 下午 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl\$PooledC
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)

七月 27, 2019 1:12:05 下午 org.hibernate.dialect.Dialect <init>

INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL57Dialect

七月 27, 2019 1:12:05 下午 org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl useContextualLobCreation

INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3] less than 4

Hibernate: select people0_.id as id1_1_0_, people0_.name as name2_1_0_, people0_.year as year3_1_0_ from people peop
====小王

Hibernate: select goodsentit0_.pid as pid4_0_0_, goodsentit0_.id as id1_0_0_, goodsentit0_.id as id1_0_1_, goodsenti
哇哈哈

瓜子

看程序再关联查询中的结果，可知，在红线处打印了第二条sql，关联查询时采用的时懒加载。可以在set标签上配置lazy="false"，来关闭懒加载的属性。

在执行一对多的一的一方的保存时

```
98 //
99 }
100 GoodsEntity goodsEntity=new GoodsEntity();
101 goodsEntity.setGoodsName("电冰箱");
102 goodsEntity.setGoodsPrice(2000);
103 GoodsEntity goodsEntity1=new GoodsEntity();
104 goodsEntity1.setGoodsName("电视机");
105 goodsEntity1.setGoodsPrice(2000);
106 People people=new People();
107 people.setName("奉先");
108 people.setYear(401);
109 Set<GoodsEntity> goodsEntities = people.getGoodsEntities();
110 goodsEntities.add(goodsEntity);
111 goodsEntities.add(goodsEntity1);
112 people.setGoodsEntities(goodsEntities);
113 session.save(people);
114 tran.commit();
115 session.close();
116 }
```

hibernateTest > oneToMonetSearch()

hibernateTest x hibernateTest.oneToMonetSearch x

Tests passed: 1 of 1 test - 2 s 606 ms

七月 27, 2019 1:38:19 下午 org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL57Dialect
七月 27, 2019 1:38:19 下午 org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl useContextualLobCreation
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3] less than 4
Hibernate: insert into people (name, year) values (?, ?)
Hibernate: insert into goods (goods_name, goods_price, pid) values (?, ?, ?)
Hibernate: insert into goods (goods_name, goods_price, pid) values (?, ?, ?)
Hibernate: update goods set pid=? where id=?
Hibernate: update goods set pid=? where id=?

Process finished with exit code 0

hibernate一共打印了五条数据，先执行主表的插入，然后执行从表的插入，再从从表的外键中建立主从表的关系维护，执行update语句。（此处主表数据一条，从表数据两条）

注：以上均为单向维护。

3.OneToOne

一对一的关系的映射有两种在数据库级别上边。

1：从表的主键参考主表的主键，主表的主键既做主键又做外键

```
create table user(
    id int(11) NOT NULL AUTO_INCREMENT,
    name varchar(20) DEFAULT NULL,
    accpass varchar(20) DEFAULT NULL,
    PRIMARY KEY (id)
);
create table reader (
    `accid` int(11) NOT NULL,
    `username` varchar(20) DEFAULT NULL,
    `birthday` date DEFAULT NULL,
    `email` varchar(50) DEFAULT NULL,
    PRIMARY KEY (`accid`),
    CONSTRAINT `detail_ibfk_1` FOREIGN KEY (`accid`) REFERENCES `account` (`id`)
)
```

2：从表的主键参考主表的外键，主表的外键是一个普通的键

配置文件中主表与从表的配置文件都需要配置

```
<one-to-one name="实体所对应的属性的名称" class="另一张关系表的classpath下的绝对路径"/>
```

例：主表配置

```
<generator class="native"></generator><!--采用数据库的本地的策略-->
<one-to-one name="detailByAccid" class="com.hibernate.pojo.Detail"/>
```

从表配置

```
<id name="accid">
    <column name="accid" sql-type="int(11)"/>
    <generator class="foreign"><!--外键的配置，此处需要配置关联的属性-->
        <param name="property">accountByAccid</param>
    </generator>
</id>
<one-to-one name="accountByAccid" class="com.hibernate.pojo.Account"/>
```