Hibernate懒加载分析

懒加载是指程序推迟访问数据库，这样做可以保证有时候不必要的访问数据库，因为访问一次数据库是比较耗时的。

# load方法的懒加载

|  |
| --- |
| public class UserManager {    public static void main(String[] args) {  Users user = new Users();  user.setBirthday(new Date());    Name name = new Name();  name.setFirstName("guo");  name.setLastName("zhao");    user.setName(name);  addUser(user);    Users users = getUser(user.getId());  System.out.println(users.getName());    }      static Users getUser(int id){  Session session = HibernateUtil.getSession();  try {  Users users = (Users) session.load(Users.class, id);  return users;  } finally{  session.close();  }  }    static void addUser(Users users){  Session session = null;  Transaction tx = null;  try {  session = HibernateUtil.getSession();  tx = session.beginTransaction();    session.save(users);  tx.commit();  } catch (HibernateException e) {  if (tx!=null) {  tx.rollback();  }  throw e;  }finally{  if(session!=null){  session.close();  }  }  }  }  上面代码是先增加一个用户、然后再查询这个用户的name组合属性。运行上面的时候，该程序会抛出这样一个异常：  Exception in thread "main" org.hibernate.LazyInitializationException: could not initialize proxy - no Session这就是懒加载不能初始化异常。其原因就在于no session。在前面个持久化对象中已经说明：使用load方法时，该方法将具有延迟加载的功能，load方法并不会立即去访问数据库，它会返回一个代理对象，只有当你真正去访问这个对象的时候，它才会去访问数据库。  http://my.csdn.net/uploads/201207/10/1341888773_2703.jpg  通过上面的图，我们看出，hibernate根本就没有select语句，也就是说hibernate没有去访问数据库，所以这个时候你去访问它肯定是没有的。但为什么没有抛出空指针异常？没有抛出空指针异常，也就是说明User对象是存在的，那它是什么呢？通过输出user.getClass()可以看出是这样一个东西：  class com.hibernate.domain.Users\_$$\_javassist\_  这个user就是load返回的代理对象。但是这个对象并不是我们所要的。我们所要的是一个User实例对象。那么怎么解决这个问题呢？  第一种方法：在关闭session之前访问该对象  static Users getUser(int id){  Session session = HibernateUtil.getSession();  try {  Users users = (Users) session.load(Users.class, id);  users.getName();  return users;  } finally{  session.close();  }  }  不过这句话看起来会很奇怪。我们通常会采用如下的方式  static Users getUser(int id){  Session session = HibernateUtil.getSession();  try {  Users users = (Users) session.load(Users.class, id);  Hibernate.initialize(users);  return users;  } finally{  session.close();  }  }  利用hibernate的initialize()方法将这个代理对象给初始化。  注：在使用代理对象的getId()方法和getClass()方法的时候，并不会抛出不能初始化异常，因为这两个属性并不用查询数据库。 |

# 关联映射懒加载

在缺省的情况下，hibernate对于关联关系会采用懒加载的方式。也就是说1-1、1-N、N-1、N-N都存在懒加载的问题。

## one-to-one懒加载

一对一的懒加载并不常用，因为懒加载的目的是为了减少与数据库的交互，从而提高执行效率，而在一对一关系中，主表中的每一条数据只对应从表的一条数据库，就算都查询也不会增加多少交互的成本，而且主表不能有contrained=true，所以主表是不能懒加载的。但是从表可以有。

实现此种懒加载必须在从对象这边同时满足三个条件：

* lazy！=false（lazy的属性有三个选项分别为：no-proxy、false和proxy）
* Constrained = true ；
* fetch=select。

注意：当fetch设置为join时，懒加载就会失效。因为fetch的作用是抓取方式，他有两个值分别问select和join，默认值为select。即在设为join时，他会直接将从表信息以join方式查询到而不是再次使用select查询，这样导致了懒加载的失效。

## one-to-many懒加载

与one-to-one关联不同，对one-to-many而言，主表的每一条属性都会对应从表的多条数据，这个时候懒加载就显得非常有效了。比如一个部门里面有多个员工，如果没有懒加载，每查询这个部门的时候都会查询出多个员工，这会大大增加与数据库交互的成本。所以Hbernate默认的是加入懒加载的。这就是查询集合属性的时候返回的是一个PersistentIndexed\*类型对象的原因。该对象其实就是一个代理对象。当然，可以在映射文件中通过将lazy属性设为假来禁用。Hibernate默认对one-to-many就是使用的懒加载，但用户也可以取消懒加载操作：

* 设置lazy="false";
* 设置fetch="join".

实现此种懒加载必须在从对象这边同时满足两个个条件：

* lazy！=false（lazy的属性有三个选项分别为：no-proxy、false和proxy）
* fetch=select。

## many-to-one懒加载

此关联关系的懒加载和one-to-one的懒加载一样都是可要可不要的，因为对执行效率的提高都不是非常明显。虽然多对一与一对一关系方式相同，但是在Hibernate中多对一时，默认是进行懒加载的。另外有一点需要注意的是懒加载并不会区分集合属性里面是否有值，即使是没有值，他依然会使用懒加载。 实现此种懒加载必须在从对象这边同时满足两个个条件：

* lazy！=false（lazy的属性有三个选项分别为：no-proxy、false和proxy）
* fetch=select

## many-to-many懒加载

此关联关系的懒加载和one-to-many的懒加载一样对程序的执行效率的提高都是非常明显的。实现此种懒加载必须在从对象这边同时满足两个个条件：

* lazy！=false（lazy的属性有三个选项分别为：no-proxy、false和proxy）
* fetch=select