并发控制

悲观锁和乐观锁

- ✓ 锁主要是解决资源并发访问的一种机制
- ✓ Hibernate支持两种锁机制：
  - ➢ “悲观锁（Pessimistic Locking）
  - ➢ “乐观锁（OptimisticLocking）

悲观锁

- ✓ 悲观锁的实现，往往依靠数据库提供的锁机制（也只有数据库层提供的锁机制才能真正保证数据访问的排他性，否则，即使在本系统中实现了加锁机制，也无法保证外部系统不会修改数据）。
- ✓ Hibernate的加锁模式有：
  - ➢ LockMode.NONE： 无锁机制。
  - ➢ LockMode.WRITE：Hibernate在Insert和Update记录的时候会自动获取。
  - ➢ LockMode.READ： Hibernate在读取记录的时候会自动获取。
    以上这三种锁机制一般由Hibernate内部使用
  - ➢ LockMode.UPGRADE：相当于数据库查询for update子句
  - ➢ LockMode. UPGRADE_NOWAIT：相当于for update nowait子句
- ✓ Hibernate的悲观锁，也是基于数据库的锁机制实现

悲观锁示例

```
String hql = "from Dept dept";
Transaction trans = session.beginTransaction();
Query query = session.createQuery(hql);

//dept为表别名，LockMode.UPGRADE相当于查询for update语句
query.setLockMode("dept", LockMode.UPGRADE);

//query.setLockMode("dept", LockMode.UPGRADE_NOWAIT);

query.list();

//提交事务之后锁自动释放
trans.commit();
```

乐观锁

- ✓ 乐观锁，大多是基于数据版本（Version）记录机制实现。何谓数据版本？即为数据增加一个版本标识，在基于数据库表的版本解决方案中，一般是通过为数据库表增加一个“version”字段来实现。
- ✓ 读取出数据时，将此版本号一同读出，之后更新时，对此版本号加一。此时，将提交数据的版本数据与数据库表对应记录的当前版本信息进行比对，如果提交的数据版本号等于数据库表当前版本号，则予以更新，否则认为是过期数据，会抛出org.hibernate.StaleObjectStateException异常，由此判断数据已过期，再进行错误处理和提示

乐观锁示例

✓ 建表脚本

```
CREATE TABLE product
(
    prod_id    NUMBER(6) PRIMARY KEY,
    prod_name  VARCHAR2(100),
    prod_price NUMBER(8,2),
    version    NUMBER(10)
);
```

乐观锁示例

✓ PO类

```
public class Product implements java.io.Serializable {

  private Integer prodId;
  private String prodName;
  private Double prodPrice;
  private Integer version;
 ……
}
```

乐观锁示例

✓ 映射文件

```
<hibernate-mapping>
   <class name="com.icss.po.Product" table="PRODUCT" optimistic-lock="version">
      <id name="prodId" type="java.lang.Integer">
         <column name="PROD_ID" precision="6" scale="0" />
         <generator class="assigned" />
      </id>
      <!-- version标签必须紧挨着id标签 -->
      <version name="version" type="java.lang.Integer">
            <column name="VERSION" precision="10" scale="0" />
      </version>
      <property name="prodName" type="java.lang.String">
         <column name="PROD_NAME" length="100" />
      </property>
      <property name="prodPrice" type="java.lang.Double">
         <column name="PROD_PRICE" precision="8" />
      </property>
   </class>
</hibernate-mapping>
```

乐观锁示例

✓ 插入数据代码

```
Transaction trans = session.beginTransaction();
Product prod = new Product(1,"橘子",3.5);
session.save(prod);
trans.commit();
```

乐观锁示例

✓ 更新数据代码

```
Transaction trans = session.beginTransaction();
/*
 * 第四个参数是版本号，如果和表version列值不一致，则会抛出
 * org.hibernate.StaleObjectStateException异常
 * 由此判断数据已过期，再进行错误处理和提示
 */
Product prod = new Product(1,"橘子",3.6,1);
session.update(prod);
trans.commit();
```