

1、添加操作 (create)

Hibernate 通过 session 的 save 方法来添加数据。

注：Hibernate自动建表是在 sessionFactory创建时期进行建表，一般都是用 static 修饰 sessionFactory，所以一般如果需要完成hibernate建表直接在封装 sessionFactory 的工具类中执行main即可

代码示例：

```
1      Transaction transaction = session.beginTransaction();
2
3      //curd操作
4      User user = new User();
5      user.setUsername("chenlong");
6  //      不要需要添加id属性
7      user.setPassword("123456");
8      user.setAddress("nankang");
9
10     session.save(user);
11     //提交事务
12     transaction.commit();
```

2、查询操作 (retrieve)

根据 id 查询：Hibernate 通过 session 的 get 方法来进行查询

session.get(Class<T>, id) 包含两个参数，第一个是查询返回数据的泛型，第二个是要查询数据的id

代码示例：

```
1      Transaction transaction = session.beginTransaction();
2
3      //curd操作
4      User user = session.get(User.class, 1);
5      System.out.println(user);
6
7
8      //提交事务
```

3、修改操作 (update)

Hibernate 通过 session 的 update 方法实现更新

实现步骤：

使用 get 方法，通过 id 查询 ----> 向返回的对象设置修改值 -----> 调用 session 中的 update 方法，更新数据

代码实现：

```
1      @Test
2      public void testUpdate() {
3          SessionFactory sessionFactory = HibernateUtils.getSessionFactory();
4          Session session = sessionFactory.openSession();
5          Transaction tx = session.beginTransaction();
6          try {
7              tx.begin();
8              User user = session.get(User.class, 1);
9              user.setUsername("Daiwei");
10         //      session.save(user);
11              session.update(user);
12              tx.commit();
13          } catch (Exception exception){
14              tx.rollback();
15              exception.printStackTrace();
16          } finally {
17              if(session != null) {
18                  session.close();
19              }
20          }
21      }
22  }
```

注：在这个更新的代码中，使用 save 方法也能实现，但是主要还是用 update 方法。

4、删除操作 (delete)

删除操作是通过 hibernate 调用 session 中的 delete 方法实现。。

实现步骤:

方法一: 使用 get 方法 获得要删除的对象---> 调用 session 的 delete 方法进行删除 (提倡)

方法二: new 一个数据库数据模型pojo---> 设置 pojo 的 id 值为要删除记录的id 值---> 调用 session 的 delete 方法进行删除

代码实现:

```
1 //方法
2 @Test
3 public void testDelete() {
4     SessionFactory sessionFactory = HibernateUtils.getSessionFactory();
5     Session session = sessionFactory.openSession();
6     Transaction tx = session.beginTransaction();
7     try{
8         tx.begin();
9         User user = session.get(User.class, 1);
10        session.delete(user);
11        tx.commit();
12    }catch (Exception e) {
13        tx.rollback();
14        e.printStackTrace();
15    }finally {
16        if(session != null) {
17            session.close();
18        }
19    }
20 }
```

```
1 //方法2
2 @Test
3 public void testDelete() {
4     SessionFactory sessionFactory = HibernateUtils.getSessionFactory();
5     Session session = sessionFactory.openSession();
6     Transaction tx = session.beginTransaction();
7     try{
8         tx.begin();
9         User user = new User();
10        user.setUid(2);
11        // User user = session.get(User.class, 1);
12        session.delete(user);
13        tx.commit();
14    }catch (Exception e) {
```

```

15         tx.rollback();
16         e.printStackTrace();
17     }finally {
18         if(session != null) {
19             session.close();
20         }
21     }
22 }

```

5、实体类对象状态（概念）

瞬时态：对象里面没有 id，对象与 session 也没有关联。

持久态：对象中有 id，且对象与 session 也有关联。

托管态：对象里面有 id，但对象与 session 没有关联。

6、saveOrUpdate方法

如果 saveOrUpdate 实体类对象的状态是瞬时态的话，saveOrUpdate 做的是 save 操作即插入操作；

```

1  @Test
2  public void saveOrUpdate() {
3      SessionFactory sessionFactory = HibernateUtils.getSessionFactory();
4      Session session = sessionFactory.openSession();
5      Transaction tx = session.beginTransaction();
6      try {
7          tx.begin();
8          User user = new User();
9          user.setUsername("DaiWei");
10         user.setAddress("GongQingCheng");
11         user.setPassword("123456");
12
13         session.saveOrUpdate(user);
14
15         tx.commit();
16     } catch (Exception e) {
17         // TODO: handle exception
18         tx.rollback();
19         e.printStackTrace();
20     } finally {
21         if(session != null) {

```

```
22         session.close();
23     }
24 }
25 }
```

如果 saveOrUpdate 实体类对象的状态是持久态 或 托管态的话，
saveOrUpdate 做的是 save 操作即更新操作；

```
1  @Test
2  public void saveOrUpdate() {
3      SessionFactory sessionFactory = HibernateUtils.getSessionFactory();
4      Session session = sessionFactory.openSession();
5      Transaction tx = session.beginTransaction();
6      try {
7          tx.begin();
8          // 瞬时态
9          // User user = new User();
10         // user.setUsername("DaiWei");
11         // user.setAddress("GongQingCheng");
12         // user.setPassword("123456");
13
14         // 持久态
15         // User user = session.get(User.class, 4);
16         // user.setPassword("345");
17         // 托管态
18         User user = new User();
19         user.setUid(4);
20         user.setPassword("123456");
21         session.saveOrUpdate(user);
22         tx.commit();
23
24     } catch (Exception e) {
25         // TODO: handle exception
26         tx.rollback();
27         e.printStackTrace();
28     } finally {
29         if(session != null) {
30             session.close();
31         }
32     }
33 }
```

