

JAVA资格认证:HibernateTemplate类使用Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文

[https://www.100test.com/kao\\_ti2020/559/2021\\_2022\\_JAVA\\_E8\\_B5\\_84\\_E6\\_A0\\_BC\\_c104\\_559601.htm](https://www.100test.com/kao_ti2020/559/2021_2022_JAVA_E8_B5_84_E6_A0_BC_c104_559601.htm) 目的：使用HibernateTemplate

执行execute ( new HibernateCallback ( ) ) 方法，

从HibernateCallback中得到session，在此session中做多个操作，并希望这些操作位于同一个事务中。如果你这样写 ( 1 )

```
public static void main(String ss[]) {
    CtxUtil.getBaseManager().getHibernateTemplate().execute(new
    HibernateCallback() { public Object doInHibernate(Session
    session) throws HibernateException, SQLException { // 保存stu1
    Student stu1 = new Student(). stu1.setName("aaaa").// 在数据库
    中,name字段不允许为null session.save(stu1). session.flush().//实
    际上，如果不是程序员"手痒"来调用这
    个flush(),HibernateTemplate中session的事务处理还是很方便的
    Student stu2 = new Student(). session.save(stu2).// 没有设置name
    字段，预期会报出例外 session.flush(). return null. } }). } 你期
    望spring在执行完execute回调后，在关闭session的时候提交事
    务，想法是很好的，但spring并不会这么做。让我们来看看在
    Hibernate的源代码中，session.beginTransaction ( ) 做了什么事
    。看如下代码 ( 2 ) : public Transaction beginTransaction()
    throws HibernateException { errorIfClosed(). if ( rootSession != null
    ) { // todo : should seriously consider not allowing a txn to begin
    from a child session // can always route the request to the root
    session log.warn( "Transaction started on non-root session" ). }
    Transaction result = getTransaction(). result.begin(). return result. }
```

这个方法中的result是一

个org.hibernate.transaction.JDBCTransaction实例，而方法中的getTransaction（）方法源代码为（3）：  
public Transaction  
getTransaction() throws HibernateException { if

```
(hibernateTransaction==null) {
```

```
log.error(owner.getFactory().getSettings()
```

```
.getTransactionFactory().getClass()). hibernateTransaction =
```

```
owner.getFactory().getSettings().getTransactionFactory()
```

```
.createTransaction( this, owner ). } return hibernateTransaction. } 再
```

次追踪，owner.getFactory（）。getSettings（）

.getTransactionFactory（）的createTransaction（）方法源代码

如下（4）：  
public Transaction createTransaction(JDBCContext  
jdbcContext, Context transactionContext) throws

```
HibernateException { return new JDBCTransaction( jdbcContext,  
transactionContext ). } 它返回了一个JDBCTransaction，没什么
```

特别的。在代码2中，执行了result.begin（），其实也就是

JDBCTransaction实例的begin（）方法，来看看（5）：

```
public void begin() throws HibernateException { if (begun) { return.
```

```
} if (commitFailed) { throw new TransactionException("cannot
```

```
re-start transaction after failed commit"). } log.debug("begin"). try {
```

```
toggleAutoCommit = jdbcContext.connection().getAutoCommit().
```

```
if (log.isDebugEnabled()) { log.debug("current autocommit status: "
```

```
toggleAutoCommit). } if (toggleAutoCommit) {
```

```
log.debug("disabling autocommit").
```

```
jdbcContext.connection().setAutoCommit(false).//把自动提交设
```

```
为了false } } catch (SQLException e) { log.error("JDBC begin
```

```
failed", e). throw new TransactionException("JDBC begin failed: ",
e). } callback = jdbcContext.registerCallbackIfNecessary(). begun =
true. committed = false. rolledBack = false. if (timeout > 0) {
jdbcContext.getConnectionManager().getBatcher().setTransaction
Timeout(timeout). } jdbcContext.afterTransactionBegin(this). }
```

100Test 下载频道开通 , 各类考试题目直接下载。详细请访问  
[www.100test.com](http://www.100test.com)