Hibernate 疑难异常及处理

博客分类：

- 疑难杂症

HibernateDAOBeanSpringMySQL

1、a different object with the same identifier value was already associated with the session。

错误原因：在hibernate中同一个session里面有了两个相同标识但是是不同实体。

解决方法一：session.clean()

PS：如果在clean操作后面又进行了saveOrUpdate(object)等改变数据状态的操作，有可能会报出"Found two representations of same collection"异常。

解决方法二：session.refresh(object)

PS：当object不是数据库中已有数据的对象的时候，不能使用session.refresh(object)因为该方法是从hibernate的session中去重新取object，如果session中没有这个对象，则会报错所以当你使用saveOrUpdate(object)之前还需要判断一下。

解决方法三：session.merge(object)

PS：Hibernate里面自带的方法，推荐使用。

2、Found two representations of same collection

错误原因：见1。

解决方法：session.merge(object)

以上两中异常经常出现在一对多映射和多对多映射中。

3、net.sf.hibernate.TransientObjectException: object references an unsaved transient instance - save the transient instance before flushing: BBusinessman

从这个bug的字面上，应该是BBusinessman的某个属性是一个实体，在这个实体没有保存之前就保存Businessman对象，导致的错误。所以我就一直看Businessman的属性中到底哪个是实体，结果发现三个，分别City，Type, Status，而且这三个实体都是dao.load(Id，Session)从数据库中获得的，不是已经有Id的，或者是null，而不存在没有保存的实体。

于是我又怀疑，是否是同一个事务需要一个session，如果不使用此session，是不是这个原因，于是我把事务中方法里所有dao.getSession()这样不太明确的地方都是用方法传入的session,这样session都是同一个了，但是仍旧有此错误。

这样，BBusinessman的各个属性都是没有问题的。那么是否是没有保存BBusinessman之前，而且BBusinessman又被当作某个实体的属性，首先保存此实体，然后去保存BBusinessman，这样是否会导致这个错误。结果我发现，正是这个问题。

==========错误的写法：==========

```
Session session = dao.getSession();
Transaction tx = session.beginTransaction();
Bo.setBman( form ,man,session);
Bo.saveChangeTable( man,session); // 把man当作属性赋给ChangeTable,并保存ChangeTable. 就是这出的错，
dao.save(man,session);  // 保存man
```

tx.commit();
==================== 应该这样写：===============
Session session = dao.getSession();
Transaction tx = session.beginTransaction();
Bo.setBman(form ,man,session);
dao.save(man,session); // 保存man
Bo.saveChangeTable(man,session); // 把man当作属性赋给ChangeTable,并保存ChangeTable. 就是这出的错，
tx.commit();
这样，问题就解决了。

4、Write operations are not allowed in read-only mode (FlushMode.NEVER) - turn your Session into FlushMode.AUTO or remove 'readOnly' marker from transaction definition 错误解决
错误代码:
org.springframework.dao.InvalidDataAccessApiUsageException: Write operations are not allowed in read-only mode (FlushMode.NEVER) - turn your Session into FlushMode.AUTO or
remove 'readOnly' marker from transaction definition
错误原因:
OpenSessionInViewFilter在getSession的时候,会把获取回来的session的flush mode 设为FlushMode.NEVER。然后把该sessionFactory绑定到TransactionSynchronizationManager，使request的整个过程都使用同一个session，在请求过后再接除该sessionFactory的绑定，最后closeSessionIfNecessary根据该session是否已和transaction绑定来决定是否关闭session。在这个过程中，若HibernateTemplate 发现自当前session有不是readOnly的transaction，就会获取到FlushMode.AUTO Session，使方法拥有写权限。
也即是，如果有不是readOnly的transaction就可以由Flush.NEVER转为Flush.AUTO,拥有insert,update,delete操作权限，如果没有transaction，并且没有另外人为地设flush model的话，则doFilter的整个过程都是Flush.NEVER。所以受transaction保护的方法有写权限，没受保护的则没有。
参考文章:
http://calvin.blog.javascud.org/post/46.htm
解决办法:
采用spring的事务声明,使方法受transaction控制

```
<bean id="baseTransaction"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean"
        abstract="true">
      <property name="transactionManager" ref="transactionManager"/>
      <property name="proxyTargetClass" value="true"/>
      <property name="transactionAttributes">
        <props>
          <prop key="get*">PROPAGATION_REQUIRED,readOnly</prop>
          <prop key="find*">PROPAGATION_REQUIRED,readOnly</prop>
          <prop key="load*">PROPAGATION_REQUIRED,readOnly</prop>
          <prop key="save*">PROPAGATION_REQUIRED</prop>
```

```
            <prop key="add*">PROPAGATION_REQUIRED</prop>
            <prop key="update*">PROPAGATION_REQUIRED</prop>
            <prop key="remove*">PROPAGATION_REQUIRED</prop>
        </props>
    </property>
</bean>
<bean id="userService" parent="baseTransaction">
    <property name="target">
        <bean class="com.phopesoft.security.service.impl.UserServiceImpl"/>
    </property>
</bean>
```

5、关于Hibernate的 Batch update returned unexpected row count from update异常

ERROR [http-8080-Processor22] (BatchingBatcher.java:60) - Exception executing batch: org.hibernate.StaleStateException: Batch update returned unexpected row count from update: 0 actual row count: 0 expected: 1

1).使用的是hibernate的saveOrUpdate方法保存实例。saveOrUpdate方法要求ID为null时才执行SAVE，在其它情况下执行UPDATE。在保存实例的时候是新增，但你的ID不为null，所以使用的是UPDATE，但是数据库里没有主键相关的值，所以出现异常。

==========

异常：

在插入时:

org.hibernate.StaleStateException: Batch update returned unexpected row count from update: 0 actual row count: 0 expected: 1

解决方法：

如果是自增主键?

有的数据库是可以修改自增主键例如:mysql,有的数据库是不允许修改自增主键的例如postgresql

不要设置自增主键的值

2)

在Hibernate映射一对多，多对一，多对多的时候新增常常会出现这个异常，代码如下：

```
public void saveFunctionCell(FunctionCell functionCell, Integer pid) {
  System.out.println("现在进行新增操作");
  FunctionCell fc = new FunctionCell();
    try {
   BeanUtils.copyProperties(fc, functionCell);
  } catch (IllegalAccessException e) {
   e.printStackTrace();
  } catch (InvocationTargetException e) {
   e.printStackTrace();
```

```
 }
 fc.setFuncCellID(null);
 // 获得父权限
 FunctionCell pfc = functionCellDao.findFunctionCellByID(pid);
 fc.setParentFunctionCell(pfc);
 functionCellDao.saveFunctionCell(fc);
}
```

注意特别标识出来的这个地方，BeanUtils拷贝Bean属性的时候，它会将你的Integer类型全部设置成0，在这里设置一个空，这样就不会抛出错误了。