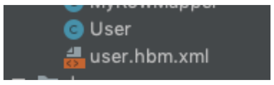
步骤：
1.创建实体类和映射文件
2.添加核心配置文件 hibernate.cfg.xml
3.配置spring配置文件，添加sessionFactory，添加DataSource，添加事务处理

首先将Hibernate所有包都添加到spring中来，然后创建实体对象和实体映射文件



```java
package bean;

public class User {

    private Integer uid;
    private String username;
    private String password;
    private String address;

     public Integer getUid() {
         return uid;
     }

     @Override
     public String toString() {
         return "User{" +
                 "uid=" + uid +
                 ", username='" + username + '\'' +
                 ", password='" + password + '\'' +
                 ", address='" + address + '\'' +
                 '}';
     }

     public void setUid(Integer uid) {
         this.uid = uid;
     }

     public String getUsername() {
         return username;
     }

     public void setUsername(String username) {
         this.username = username;
     }

     public String getPassword() {
         return password;
     }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}
```

映射文件

```xml
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD//EN"
        "http://www.hibernate.org/xsd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class table="user" name="bean.User">
        <id name="uid" column="uid">
            <generator class="native"></generator>
        </id>
         <property name="username" column="username"></property>
         <property name="password" column="password"></property>
         <property name="address" column="address"></property>
     </class>
 </hibernate-mapping>
```

添加核心配置文件 hibernate.cfg.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>

        <!--输出底层的sql语句-->
        <property name="hibernate.show_sql">true</property>
         <!--hibernatec创建表，update表示如果已经有表就创建，没有就更新-->
         <property name="hibernate.hbm2ddl.auto">update</property>
         <!--配置数据库方言，mysql中分院为limit，oracle中为rownum，让hibernate识别不同数据库特有的
         <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property

         <mapping resource="bean/user.hbm.xml"></mapping>
     </session-factory>
 </hibernate-configuration>
```

原本在核心配置文件中配置的数据库信息，放到了spring中的配置文件中配置，其实核心文件中的内容都可以放到spring配

置文件中配置。

原本我们使用Hibernate时，需要配置Configuration,来获得SessionFactory

```
configuration = new Configuration();
configuration.configure();

sessionFactory = configuration.buildSessionFactory();
```

现在我们直接使用spring来进行配置：
1.添加dataSource
2.添加核心配置文件

```

<!--    配置sessionFactory创建-->
    <bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFacto
<!--        指定数据库信息-->
        <property name="dataSource" ref="dataSource"></property>
<!--        指定核心文件的位置-->
        <property name="configLocations" value="classpath:hibernate.cfg.xml"></property>
    </bean>

     <!--    配置c3p0--->
 <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
     <!--        注入属性-->
     <property name="driverClass" value="com.mysql.cj.jdbc.Driver"></property>
     <property name="jdbcUrl" value="jdbc:mysql:///test"></property>
     <property name="user" value="root"></property>
     <property name="password" value="root"></property>
 </bean>
```

我们也可以不需要核心配置文件进行配置

```
 <!--    配置sessionFactory创建-->
    <bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFacto
        <!--        指定数据库信息-->
        <property name="dataSource" ref="dataSource"></property>

<!--        不需要核心配置文件-->
        <property name="mappingResources">
            <list>
                <value>bean/user.hbm.xml</value>
             </list>
         </property>
         <property name="hibernateProperties">
             <props>
                 <prop key="dialect">org.hibernate.dialect.MySQL8Dialect</prop>
                 <prop key="hibernate.show_sql">true</prop>
                 <prop key="hibernate.format_sql">true</prop>
             </props>
         </property>
        </bean>
```

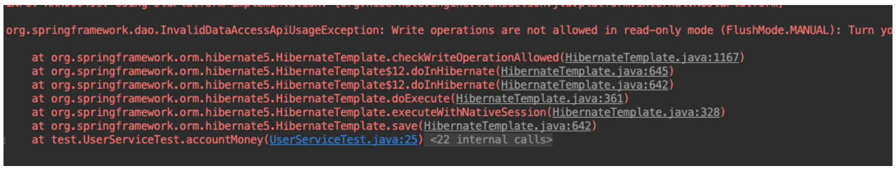
然后我们配置HibernateTemplate的使用

```xml
--    创建Hibernate模板对象-->
  <bean id="hibernateTemplate" class="org.springframework.orm.hibernate5.HibernateTemplate">
      <property name="sessionFactory" ref="sessionFactory"></property>
  </bean>

  <bean id="userDao" class="dao.UserDao">
      <property name="hibernateTemplate" ref="hibernateTemplate"></property>
  </bean>
```

配置完毕后，我们来测试使用

```java
     ApplicationContext context = new FileSystemXmlApplicationContext("/web/WEB-INF/applica
          UserDao userDao = (UserDao) context.getBean("userDao");

          User user = new User();
          user.setUsername("王五");
          user.setPassword("147566");
          user.setAddress("五道口");
          userDao.getHibernateTemplate().save(user);
```

此时保存，表示事务只读，说明我们要配置事务



```
org.springframework.dao.InvalidDataAccessApiUsageException: Write operations are not allowed in read-only mode (FlushMode.MANUAL): Turn yo

    at org.springframework.orm.hibernate5.HibernateTemplate.checkWriteOperationAllowed(HibernateTemplate.java:1167)
    at org.springframework.orm.hibernate5.HibernateTemplate$12.doInHibernate(HibernateTemplate.java:645)
    at org.springframework.orm.hibernate5.HibernateTemplate$12.doInHibernate(HibernateTemplate.java:642)
    at org.springframework.orm.hibernate5.HibernateTemplate.doExecute(HibernateTemplate.java:361)
    at org.springframework.orm.hibernate5.HibernateTemplate.executeWithNativeSession(HibernateTemplate.java:328)
    at org.springframework.orm.hibernate5.HibernateTemplate.save(HibernateTemplate.java:642)
    at test.UserServiceTest.accountMoney(UserServiceTest.java:25) <22 internal calls>
```

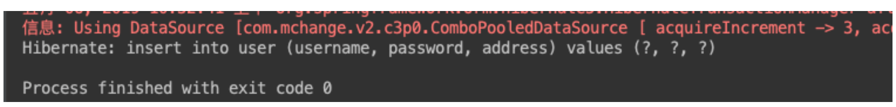
```xml
    配置事务管理器-->
n id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManag
<property name="sessionFactory" ref="sessionFactory"></property>
an>

    开启事务注解-->
annotation-driven transaction-manager="transactionManager"></tx:annotation-driven>

n id="userService" class="dao.UserService">
 <property name="userDao" ref="userDao"></property>
ean>
```

业务逻辑层

```java
//业务逻辑层
@Transactional
public class UserService {

    private UserDao userDao;

    public UserDao getUserDao() {
        return userDao;
    }

     public void setUserDao(UserDao userDao) {
         this.userDao = userDao;
     }

     public void save(User user){
         userDao.getHibernateTemplate().save(user);
     }
}
```

测试使用

```java
 @Test
   public void accountMoney() {

       ApplicationContext context = new FileSystemXmlApplicationContext("/web/WEB-INF/ap
       UserService userService = (UserService) context.getBean("userService");

       User user = new User();
       user.setUsername("王五");
       user.setPassword("147566");
        user.setAddress("五道口");
        userService.save(user);
    }
```



```
信息: Using DataSource [com.mchange.v2.c3p0.ComboPooledDataSource [ acquireIncrement -> 3, ac
Hibernate: insert into user (username, password, address) values (?, ?, ?)

Process finished with exit code 0
```

整合成功