# Hibernate的session获取方式

## Bean配置方式获取

Hibernate获取session的方式有2种，第一种通过数据源配置获取链接，是bean配置

|  |
| --- |
| <bean id=*"sessionFactory"* class=*"org.springframework.orm.hibernate3.LocalSessionFactoryBean"*>  <property name=*"dataSource"* >  <ref local=*"dataSource"*/>  </property>  <!-- 配置Hibernate的属性 -->  <property name=*"hibernateProperties"*>  <props>  <prop key=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</prop>  <prop key=*"hibernate.show\_sql"*>true</prop>  </props>  </property>  <!-- 指定HIbernate映射文件的路径 -->  <property name=*"mappingResources"*>  <list>  <value>com/integration/entity/User.hbm.xml</value>  </list>  </property>  </bean>  <bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>  <property name=*"driverClassName"*>  <value>com.mysql.jdbc.Driver</value>  </property>  <property name=*"url"*>  <value>jdbc:mysql://localhost:3306/test  </value>  </property>  <property name=*"username"*>  <value>root</value>  </property>  <property name=*"password"*>  <value>1995</value>  </property>  </bean>  <bean id=*"userDAO"*  class=*"com.integration.dao.UserDAOImpl"*  abstract=*"false"* lazy-init=*"default"* autowire=*"default"*>  <property name=*"sessionFactory"*>  <ref bean=*"sessionFactory"* />  </property>  </bean> |

配置顺序就是

配置数据源->sessionFactory->session,这种就是在web应用中用的多，因为web应用都是用数据源的，不会去自己申请数据库连接

## API方式获取

另外一种方式是直接调用hibernate的api获取sessionFactory

|  |
| --- |
| SessionFactory sessionFactory = **new** AnnotationConfiguration().configure().buildSessionFactory(); |

记得有这么回事就行了，术业有专攻，我们都是web应用，用的少的，瞄一眼就行了亲。

# 数据操作流程

## 说明

1. 获取session
2. 获取transaction
3. 增删改查操作
4. 提交，关闭session

|  |
| --- |
| SessionFactory sessionFactory = **new** AnnotationConfiguration().configure().buildSessionFactory();  Session session = sessionFactory.getCurrentSession();  session.beginTransaction();  Transaction tx = session.beginTransaction();  session.save(t);  tx.commit(); |

# 实体类

使用注解更加方便

|  |
| --- |
| **package** com.bjsxt.hibernate;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  @Entity  **public** **class** Teacher {  **private** **int** id;  **private** String name;  **private** String title;    @Id  @GeneratedValue(strategy=GenerationType.***IDENTITY***)  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getTitle() {  **return** title;  }  **public** **void** setTitle(String title) {  **this**.title = title;  }  } |

@entity

@Id

## 主键生成方式

1．assigned让程序员指定主键，不推荐   
2.sequence 要数据库支持sequence ,mysql不可以用这个   
3.increment自动递增，由hibernate管理主键在表里面找一个最大的id，然后在最大的基础上+1递增   
4.native让hibernate自己决定   
5.identity 由数据库管理主键，表要设置成autoincrement ，和increment类似，但是前者是由hibernate管理主键，后者是由数据库管理主键

一般设置成identity 算了

设置成AUTO是让程序管理主键   
@Id   
@GeneratedValue(strategy=GenerationType.AUTO)   
设置成identity就是数据库管理主键   
@Id   
@GeneratedValue(strategy=GenerationType.IDENTITY)

但是我发现，这2种配置都要求数据库表要设置成自动递增

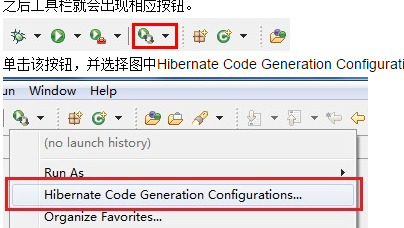
# 从表中逆向生成实体类

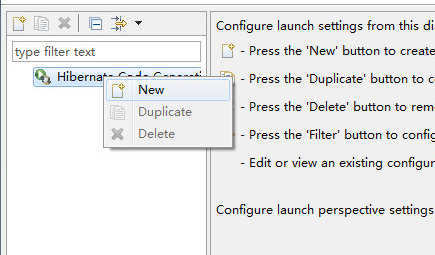
## Eclipse操作

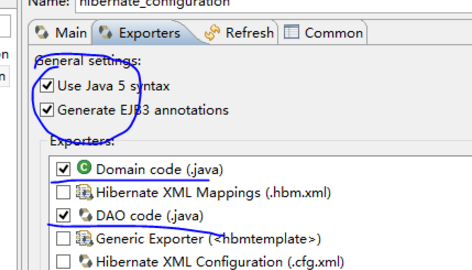
1. 安装hibernate插件
2. 打开视图hibernate configuration
3. 添加配置

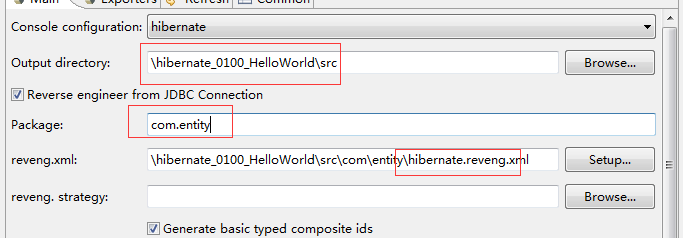


1. 打开hibernate perspective视图
2. 如图









输出目录不要选择到包名，选择src目录即可

Package才是选择包名的

Reveng.xml可以选择生成哪些表

# 增删改查

## 插入数据

|  |
| --- |
| Teacher t = **new** Teacher();  //t.setId(1);  t.setName("t1");  t.setTitle("middle");  SessionFactory sessionFactory = **new** AnnotationConfiguration().configure().buildSessionFactory();  Session session = sessionFactory.getCurrentSession();  session.beginTransaction();  Transaction tx = session.beginTransaction();  session.save(t);  tx.commit(); |

因为设置好了主键生成策略，所以不需要指定主键，执行一次就在表里面插入一条数据

## 删除数据

|  |
| --- |
| Teacher t = **new** Teacher();  t.setId(2);  session.beginTransaction();  Transaction tx = session.beginTransaction();  session.delete(t);  tx.commit(); |

## 更新数据

|  |
| --- |
| Teacher t = **new** Teacher();  t.setId(1);//update必须设置ID  t.setName("huang");  session.beginTransaction();  Transaction tx = session.beginTransaction();  session.update(t);  tx.commit(); |

## 查询数据

|  |
| --- |
| Transaction tx = session.beginTransaction();  Teacher t = (Teacher) session.get(Teacher.**class**, 1);//查找主键为1的System.***out***.println(t.getName()); |

# 实体数据的状态

## 说明

在Hibernate中，对象的存在状态有三种，分别是：Transient（瞬时状态）、Persistent（持久化状态）、Detached（脱管/游离状态）。这三种状态的区别在于：

**1.Transient：**处于瞬时状态时，对象只存在于JVM内存中，并没有和Hibernate中的Session关联，没有纳入到Hibernate的缓存管理中去，在数据库中也没有与对象对应的记录。如：新建一个对象时，该对象就处于瞬时状态。   
**2.Persistent：**处于持久化状态时，对象不仅在内存中占有空间，Hibernate缓存Session中也存在该对象，并且数据库表中有与该对象对应的记录，主键值确定。   
**3.Detached：**处于游离状态时，对象在内存中存在，在数据库中有与之对应的记录，但是不存在于Session缓存中

|  |
| --- |
| // 新建一个User类对象，此时user处于瞬时状态（Transient）  User user=new User();  user.setId(1);  user.setName("Luvjuin");  // 获取Session  Configuration cfg=new Configuration().configure();  SessionFactory sf=cfg.buildSessionFactory();  Session session=sf.openSession();  //开启事务  session.beginTransaction();  //保存user对象到数据库  //事务提交后，对象将会处于持久化状态（Persistent）  session.save(user);  //事务提交  session.getTransaction().commit();  //关闭Session和SessionFactroy  //此时user对象将处于游离状态（Detached）  session.close();  sf.close(); |

