目录

**[新建数据库：](#_Toc20454_WPSOffice_Level1)** **[2](#_Toc20454_WPSOffice_Level1)**

**[1. 新建数据库](#_Toc5055_WPSOffice_Level1)** **[2](#_Toc5055_WPSOffice_Level1)**

**[2. 设置双主键 s\_studnetId s\_id（按住shift来设置）](#_Toc14207_WPSOffice_Level1)** **[2](#_Toc14207_WPSOffice_Level1)**

**[3. 表的设计](#_Toc18818_WPSOffice_Level1)** **[2](#_Toc18818_WPSOffice_Level1)**

**[创建工程：](#_Toc8481_WPSOffice_Level1)** **[3](#_Toc8481_WPSOffice_Level1)**

**[1. 创建duanxueqi web工程：](#_Toc8444_WPSOffice_Level1)** **[3](#_Toc8444_WPSOffice_Level1)**

**[2. 导入lib包](#_Toc15925_WPSOffice_Level1)** **[3](#_Toc15925_WPSOffice_Level1)**

**[3. 搭建ssh支持：](#_Toc15708_WPSOffice_Level1)** **[3](#_Toc15708_WPSOffice_Level1)**

[① 添加spring支持](#_Toc5055_WPSOffice_Level2) [3](#_Toc5055_WPSOffice_Level2)

[② 添加structs支持](#_Toc14207_WPSOffice_Level2) [3](#_Toc14207_WPSOffice_Level2)

[③ 导入mysql的架包：](#_Toc18818_WPSOffice_Level2) [4](#_Toc18818_WPSOffice_Level2)

[④ 添加数据库](#_Toc8481_WPSOffice_Level2) [4](#_Toc8481_WPSOffice_Level2)

[⑤ 添加Hibernate支持](#_Toc8444_WPSOffice_Level2) [4](#_Toc8444_WPSOffice_Level2)

[⑥ 建立spring struts联系](#_Toc15925_WPSOffice_Level2) [5](#_Toc15925_WPSOffice_Level2)

[⑦ 工程建立完成](#_Toc15708_WPSOffice_Level2) [6](#_Toc15708_WPSOffice_Level2)

**[4.对student表反向工程](#_Toc29662_WPSOffice_Level1)** **[6](#_Toc29662_WPSOffice_Level1)**

[（选择native自增）](#_Toc29662_WPSOffice_Level2) [7](#_Toc29662_WPSOffice_Level2)

**[5.连接数据库的时候还有一个关键一步，加上这句话，不然跑不起来，](#_Toc19239_WPSOffice_Level1)** **[7](#_Toc19239_WPSOffice_Level1)**

**[6.效果](#_Toc20688_WPSOffice_Level1)** **[7](#_Toc20688_WPSOffice_Level1)**

**[7.这里对student表进行点修改](#_Toc26999_WPSOffice_Level1)** **[8](#_Toc26999_WPSOffice_Level1)**

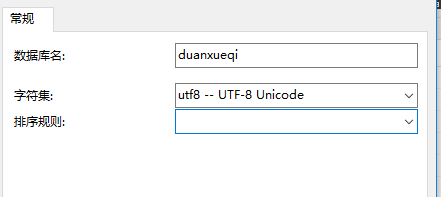
[(接着还有要改:)](#_Toc19239_WPSOffice_Level2) [8](#_Toc19239_WPSOffice_Level2)

**[8. 新建studentDao](#_Toc32273_WPSOffice_Level1)** **[8](#_Toc32273_WPSOffice_Level1)**

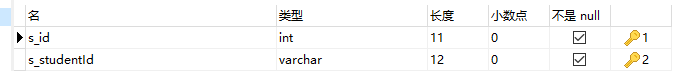
**[9. 新建studentDaoImp](#_Toc21059_WPSOffice_Level1)** **[8](#_Toc21059_WPSOffice_Level1)**

## 新建数据库：

1. 新建数据库



1. 设置双主键 s\_studnetId s\_id（按住shift来设置）



1. 表的设计

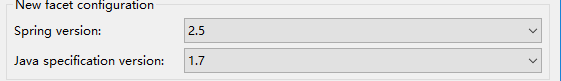
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 描述 | 类型和长度 | 主键 | 可空 | 默认值 |
| 1 | s\_id | 学生id | int | 是 | 否 | 无（自增） |
| 2 | s\_studentId | 学生学号 | varchar(12) | 是 | 否 | 无 |
| 3 | s\_name | 学生姓名 | varchar(255) | 否 | 是 | 无 |
| 4 | s\_sex | 性别（0位男生，1位女生） | tinyint | 否 | 是 | 无 |
| 5 | s\_school | 学校 | varchar(255) | 否 | 是 | 无 |
| 6 | s\_college | 所在学院 | varchar(255) | 否 | 是 | 无 |
| 7 | s\_profession | 所在专业 | varchar(255) | 否 | 是 | 无 |
| 8 | s\_class | 所在班级 | varchar(255) | 否 | 是 | 无 |
| 9 | s\_classtype | 所在班级类型 | varchar(255) | 否 | 是 | 无 |
| 10 | s\_jobstatus | 班级内担任职务 | varchar(255) | 否 | 是 | 无 |
| 11 | s\_tutor | 学生导师 | varchar(255) | 否 | 是 | 无 |
| 12 | s\_phone | 学生手机号 | varchar(15) | 否 | 是 | 无 |
| 13 | s\_idcard | 学生身份证号 | varchar(18) | 否 | 是 | 无 |
| 14 | s\_password | 密码 | varchar(255) | 否 | 是 | 无 |
| 15 | s\_createTime | 创建时间 | datetime | 否 | 是 | 无 |
| 16 | s\_changeTime | 最近一次修改信息时间 | datetime | 否 | 是 | 无 |
| 17 | s\_picture | 学生照片 | varchar(255) | 否 | 是 | 无 |
| 18 | s\_token | token号 | varchar(255) | 否 | 是 | 无 |
| 19 | s\_tokenendtime | token保存到期时间 | datetime | 否 | 是 | 无 |

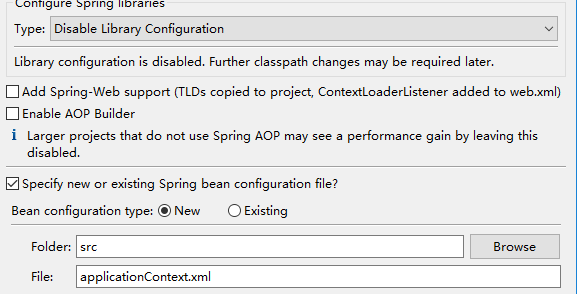
## 创建工程：

1. 创建duanxueqi web工程：

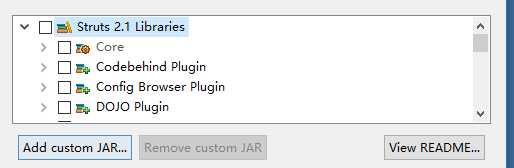


1. 导入lib包
2. 搭建ssh支持：
3. 添加spring支持



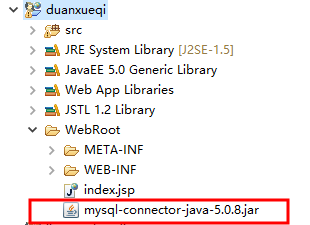


1. 添加structs支持

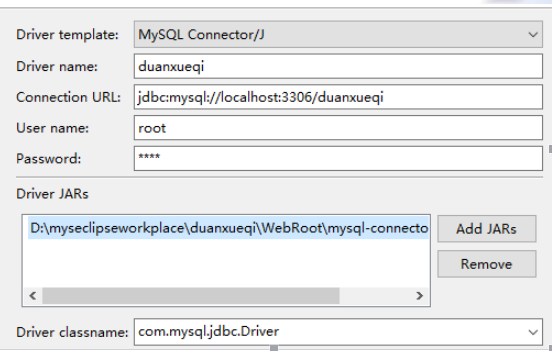


勾勾都去掉

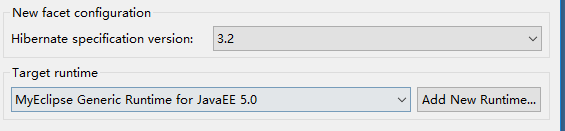
1. 导入mysql的架包：

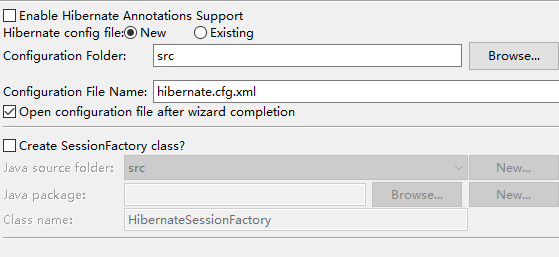


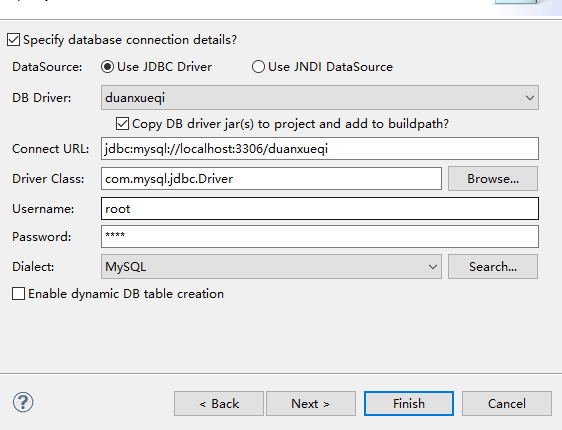
1. 添加数据库

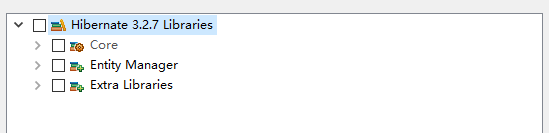


1. 添加Hibernate支持







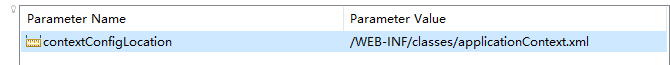


1. 建立spring struts联系
2. 在src下新建一个文件及并添加这项数据：struts.objectFactory=spring

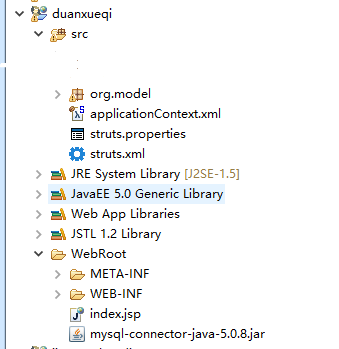


1. 在web.xml配置

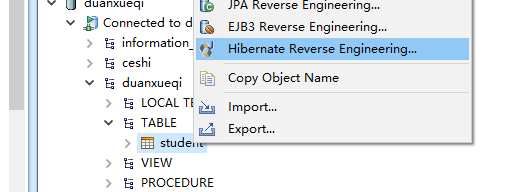
添加：org.springframework.web.context.ContextLoaderListener

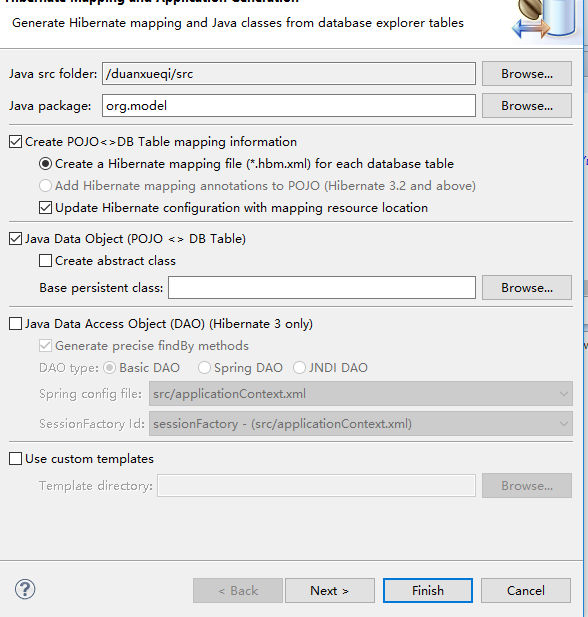
添加：

1. 工程建立完成

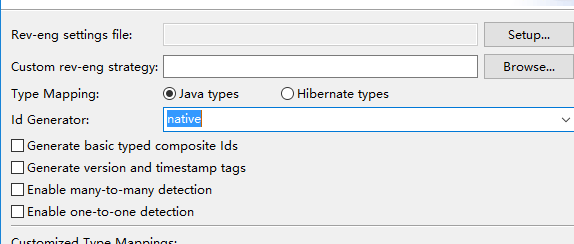


4.对student表反向工程

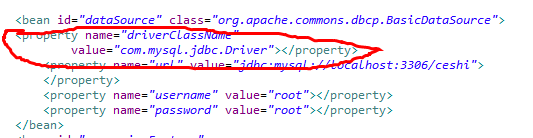




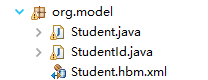
（选择native自增）



5.连接数据库的时候还有一个关键一步，加上这句话，不然跑不起来，



6.效果



7.这里对student表进行点修改

改成boolean型

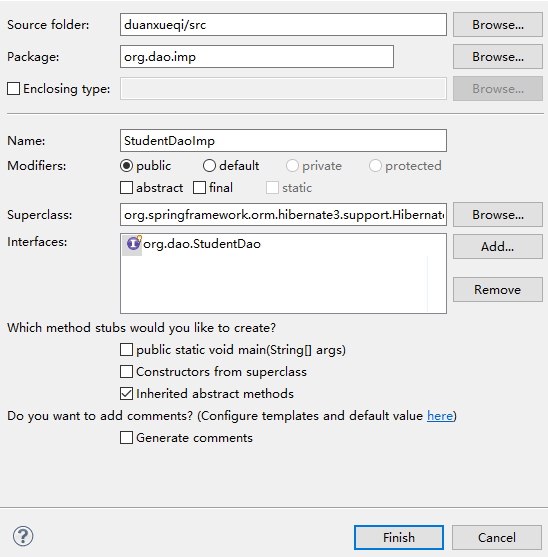
(接着还有要改:)



1. 新建studentDao

|  |
| --- |
| **public** **interface** StudentDao {  //1.获得所有学院  List<Student> allStudentCollege();  //2.获得所有年级  List<Student> allStudentGrade();  //3.获得所有专业  List<Student>allStudentProfession();  //4.获得所有班级  List<Student>allStudentClass();  //5.获得指定学院、年级、专业、班级的学生的Map对象列表(用于返回json串)  List<Map<String,Object>>allStudentMap(String stuCollege,String stuGrade,String stuProfession,String stuClass);  } |

1. 新建studentDaoImp



|  |
| --- |
| **public** **class** StudentDaoImp **extends** HibernateDaoSupport **implements** StudentDao {  **public** List<Student> allStudentCollege() {  List<Student> list=**this**.getHibernateTemplate().find("from Student s group by s.SCollege");  **return** (list.size()==0)?**null**:list;  }  **public** List<Student> allStudentGrade() {  List<Student> list=**this**.getHibernateTemplate().find("from Student s group by s.SGrade");  **return** (list.size()==0)?**null**:list;  }  **public** List<Student> allStudentProfession() {  List<Student> list=**this**.getHibernateTemplate().find("from Student s group by SProfession");  **return** (list.size()==0)?**null**:list;  }  **public** List<Student> allStudentClass() {  List<Student> list=**this**.getHibernateTemplate().find("from Student s group by SClass");  **return** (list.size()==0)?**null**:list;  }  **public** List<Map<String, Object>> allStudentMap(String stuCollege,  String stuGrade, String stuProfession, String stuClass) {  List<Student> slist;  //所有学院  **if**(stuCollege.equals("all")){  //所有学院、所有年级  **if**(stuGrade.equals("all")){  slist=**this**.getHibernateTemplate().find("from Student");  }**else**{  //所有学年，指定年级  slist=**this**.getHibernateTemplate().find("from Student where SGrade=?",stuGrade);  }  }  **else**{  //指定学院、所有专业  **if**(stuProfession.equals("all")){  //指定学院、所有专业、所有年级  **if**(stuGrade.equals("all")){  slist=**this**.getHibernateTemplate().find("from Student where SCollege=?",stuCollege);  }**else**{  //指定学院、所有专业、指定年级  String[] list={stuCollege,stuProfession};  slist=**this**.getHibernateTemplate().find("from Student where SCollege=? and SGrade=?",list);  }  }  **else**{  //指定学院、所有专业、所有班级  **if**(stuClass.equals("all")){  //指定 学院、指定专业、所有年级  **if**(stuGrade.equals("all")){  slist=**this**.getHibernateTemplate().find("from Student where SProfession=?",stuProfession);  }  **else**{  //指定学院/指定专业。指定年级  String[] list={stuProfession,stuGrade};  slist=**this**.getHibernateTemplate().find("from Student where SProfession=? and SGrade=? ",stuProfession);  }  }**else**{  //指定学院、指定专业、指定班级  slist=**this**.getHibernateTemplate().find("from Student where SClass=?",stuClass);  }  }  }  **return** **null**;  }  } |