用户信息、职位和分组模块：

### 3.1.2 dao层

定义从BaseDao继承的接口GroupDao，代码如下

|  |
| --- |
| **public** **interface** GroupDao **extends** GenericTreeDao<Group, Long> {  Group findByName (String name);  } |

定义从BaseDao继承的接口RoleDao，代码如下

|  |
| --- |
| **public** **interface** RoleDao **extends** GenericDao<Role, Long> {  Role findByName(String name);  } |

定义从BaseDao继承的接口UserDao ，代码如下

|  |
| --- |
| **public** **interface** UserDao **extends** GenericDao<User, Long> {  User findByUsername (String username);  } |

父接口GenericDao代码如下，从对GenericDao接口的注释“Generic DAO (Data Access Object) with common methods to CRUD POJOs.” 可以看出GenericDao已经封装了对普通JavaBeans的CRUD操作。

|  |
| --- |
| **public** **interface** BaseDao <T **extends** BaseEntity, PK **extends** Serializable>  **extends** JpaRepository<T, PK>, JpaSpecificationExecutor<T> {  } |

其中JpaRepository和JpaSpecificationExecutor的API链接为<http://static.springsource.org/spring-data/data-jpa/docs/current/api/>，二者是[Spring Data](http://www.springsource.org/spring-data) Project下的接口。

### 3.1.3 service层

GroupManager

|  |
| --- |
| **public** **interface** GroupManager **extends** GenericManager<Group, Long> {  /\*\*  \*  \* Check the Group by the group's name  \* **@param** name Group name  \* **@return**  \*/  Group getbyName(String name);  } |

RoleManager

|  |
| --- |
| **public** **interface** RoleManager **extends** GenericManager<Role, Long> {  /\*\*  \*  \* Check the Role by the role's name  \* **@param** name Role name  \* **@return**  \*/  Role getbyName(String name);  } |

UserManager

|  |
| --- |
| **public** **interface** UserManager **extends** GenericManager<User, Long> {  /\*\*  \*  \* Check All Users  \* Check the User by the user's username  \* **@param** username User username  \* **@return**  \*/  List<User> findAll();  User findbyUsername(String username);  } |

GroupManagerImpl

|  |
| --- |
| /\*\*  \* group managerimpl class  \* GroupManagerImpl extends GenericManagerImpl and it's GroupManager interface  \* **@param** GroupDao Dao  \* **@author** lsz  \*  \*/  @Service("groupManager")  @Transactional  **public** **class** GroupManagerImpl **extends** GenericManagerImpl<Group, Long> **implements** GroupManager {  GroupDao groupDao;  **private** Logger logger = LoggerFactory.*getLogger*(GroupManagerImpl.**class**);  @Autowired  /\*\*  \* group managerimpl findByname method  \* **@return** object for the group's name  \* **@author** lsz  \*/  @Override  **public** Group getbyName(String name) {  // **TODO** Auto-generated method stub  **return** groupDao.findByName(name);  } |

RoleManagerImpl

|  |
| --- |
| /\*\*  \* role managerimpl class  \* RoleManagerImpl extends GenericManagerImpl and it's RoleManager interface  \* **@param** Role class  \* **@author** lsz  \*  \*/  @Component  **public** **class** RoleManagerImpl **extends** GenericManagerImpl<Role, Long> **implements** RoleManager {  **private** Logger logger = LoggerFactory.*getLogger*(RoleManagerImpl.**class**);  @Autowired  **private** RoleDao roleDao;  /\*\*  \* role managerimpl findByname method  \* **@return** object for the role's name  \* **@author** lsz  \*/  @Override  **public** Role getbyName(String name) {  // **TODO** Auto-generated method stub  **return** roleDao.findByName(name);  } |

UserManagerImpl

|  |
| --- |
| /\*\*  \* user managerimpl class  \* UserManagerImpl extends GenericManagerImpl and it's UserManager interface  \* **@param** User class  \* **@author** lsz  \*  \*/  @Component  **public** **class** UserManagerImpl **extends** GenericManagerImpl<User, Long> **implements** UserManager {  UserDao userDao;  @Autowired  **public** **void** setUserDao(UserDao userDao) {  **this**.userDao = userDao;  **this**.dao = **this**.userDao;  }  @Override  **public** User findbyUsername(String username) {  // **TODO** Auto-generated method stub  **return** **null**;  } |

**注解说明：**

各注解的具体意义见表3-3

**表3-3**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **注解名** | **注解（或属性）意义** | **备注** |
| 1 | @Service | 定义一个Bean |  |
| 2 | @Transactional | 配置事务，每一个业务方法开始时都会打开一个事务 |  |
| 3 | @Autowired | 按byType自动注入 |  |

### 3.1.4 controller层

controller层是与用户交互的一层，里面封装了具体的业务操作，代码如下

|  |
| --- |
| **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** edu.zut.cs.javaee.epms.admin.domain.User;  **import** edu.zut.cs.javaee.epms.admin.service.UserManager;  **import** edu.zut.cs.javaee.epms.base.web.spring.controller.GenericController;  @Controller  @RequestMapping("/user/user") //requestMapping  **public** **class** UserController **extends** GenericController<User, Long, UserManager> {  UserManager userManager;  @RequestMapping(method = RequestMethod.***GET***, value = "/index.html")  **public** String index() {  **return** "user/index";  }  @Autowired  **public** **void** setUserManager(UserManager userManager) {  **this**.userManager = userManager;  **this**.manager = **this**.userManager;  }  } |

**注解说明：**

各注解的具体意义见表3-4

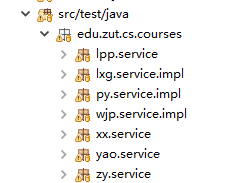
**表3-4**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **注解名** | **注解（或属性）意义** | **备注** |
| 1 | @ Controller | 注册Controller |  |
| 2 | @RequestMapping | 注册访问时的路径 |  |

## 5.2 test测试文件夹下

### 准备工作

我们构建项目的方式是多模块项目，每一个模块都会有一个test的目录。在每一个模块的test源文件夹下，如图3-3所示是service层的测试，尝试建立如下的包层次结构，test文件夹下存放的是测试src源代码的测试代码。



**图3-3**

### 3.2.1 dao层

**public** **class** DepartmentDaoTest **extends** GenericDaoTestCase<Long, Student2, Student2Dao> {

@Autowired

DepartmentDao department;

@Test

**public** **final** **void** testFindById() {

System.***out***.println(department.findByManager("lpp"));

}

}

**代码说明：**

继承基类的DepartmentDaoTest 类，对GenericDaoTestCase进行测试。

### 3.1.2 service层

public class DepartmentServiceTest extends BaseServiceTestCase<Long, Department, DepartmentService>{

public DepartmentServiceTest() {

super(Department.class);

// TODO Auto-generated constructor stub

}

@Autowired

DepartmentService departmentService;

@Test

public void FindByIdTest() {

Department d = departmentService.findByManager("lpp");

System.out.println(d.getName());

}

}

**代码说明：**

DepartmentServiceTest继承基类的BaseServiceTestCase类，对src中的类DepartmentService进行测试。

### 3.1.3 controller层

**说明：**

本项目使用jsp技术。Controller层并无测试方法。