# 增加雇员的功能

# 1.新建一个EmployeeAction，用于跳转

## 1.2配置action 在spring容器和 Struts 的config文件中

### 1.2.1Struts 的config

<!-- 配置employeeaction -->

<action

path=*"/employee"*

name=*"employeeForm"*

parameter=*"flag"*>

<forward name=*"goAddEmpUi"* path=*"/WEB-INF/addEmpUi.jsp"* />

<forward name=*"goListEmp"* path=*"/WEB-INF/listEmp.jsp"*/>

<forward name=*"goUpdEmpUi"* path=*"/WEB-INF/updEmpUi.jsp"*/>

</action>

### 1.2.2spring容器

<bean name=*"/employee"* scope=*"prototype"* class=*"com.hlj.web.action.EmployeeAction"*/>

## 1.3测试，成功

# 2.实现增加雇员的功能

## 2.1在实现因为要使用departmentService，employeeService所以我们必须对他们进行注入

**public** ActionForward addEmp(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response)

**throws** Exception {

EmployeeForm employeeForm=(EmployeeForm)form;

System.*out*.println(employeeForm.getDepartmentId()+" "+employeeForm.getEmail());

Employee employee=**new** Employee();

employee.setEmail(employeeForm.getEmail());

employee.setGrade(Integer.*parseInt*(employeeForm.getGrade()));

employee.setHiredate(**new** java.util.Date());

employee.setName(employeeForm.getName());

employee.setPwd(employeeForm.getPwd());

employee.setSalary(Float.*parseFloat*(employeeForm.getSalary())); employee.setDepartment(departmentService.findById(Integer.*parseInt*(employeeForm.getDepartmentId())));

System.*out*.println(employee.getDepartment()+"这个雇员的部门是他");

try {

employeeService.addEmployList(employee);

} catch (Exception e) {

return mapping.findForward("opererr");

}

**return** mapping.findForward("operok");

}

## 2.2对service进行注入，只有set方法就可以了，但是resource必须是有两个

@Transactional

**public** **class** EmployeeAction **extends** DispatchAction {

@Resource

**private** DepartmentServiceInter departmentService;

@Resource

**private** EmployeeServiceInter employeeService;

**public** **void** setEmployeeService(EmployeeServiceInter employeeService) {

**this**.employeeService = employeeService;

}

**public** **void** setDepartmentService(DepartmentServiceInter departmentService) {

**this**.departmentService = departmentService;

}

## 2.3这样添加雇员就成功了

# 3.对上面的代码进行更好的处理，需要重新整理

## 3.1建立基础接口声明一些常用的方法

**public** **interface** BasicServiceInter {

**public** Object findById(Class clazz,java.io.Serializable id);

**public** List executeQuery(String hql,Object []parameters);

**public** List executeQueryByPage(String hql,Object []parameters,**int** pageNow,**int** PagesSize);

**public** **void** add(Object obj);

**public** List executeUpdate(String hql,Object []parameters);

**public** Object uniqueQuery(String hql,Object []parameters);

**public** **int** **queryPageCount**(String hql,Object [] parameters,**int** pageSize);

**public** **void** delById(Class clazz,java.io.Serializable id);

**public** **void** update(Object object);

}

## 3.2实现接口

@Transactional

**public** **abstract** **class** BasicService **implements** BasicServiceInter {

@Resource

**private** SessionFactory sessionFactory;

**public** **void** setSessionFactory(SessionFactory sessionFactory) {

**this**.sessionFactory = sessionFactory;

}

**public** **void** add(Object obj) {

**this**.sessionFactory.getCurrentSession().save(obj);

}

//ͳһ��ѯ����(hql)

**public** List executeQuery(String hql, Object[] parameters) {

// **TODO** Auto-generated method stub

Query query=**this**.sessionFactory.getCurrentSession().createQuery(hql);

//ע��?ֵ

**if**(parameters!=**null** && parameters.length>0){

**for**(**int** i=0;i<parameters.length;i++){

query.setParameter(i, parameters[i]);

}

}

**return** query.list();

}

//��ҳ

**public** List executeQueryByPage(String hql, Object[] parameters,

**int** pageNow, **int** PagesSize) {

// **TODO** Auto-generated method stub

Query query=**this**.sessionFactory.getCurrentSession().createQuery(hql);

**if**(parameters!=**null** && parameters.length>0){

**for**(**int** i=0;i<parameters.length;i++){

query.setParameter(i, parameters[i]);

}

}

//���ַ�ҳ

**return** query.setFirstResult((pageNow-1)\*PagesSize).setMaxResults(PagesSize).list();

}

//ͳһ���޸ĺ�ɾ��

**public** List executeUpdate(String hql, Object[] parameters) {

// **TODO** Auto-generated method stub

**return** **null**;

}

**public** Object findById(Class clazz, Serializable id) {

// **TODO** Auto-generated method stub

**return** **this**.sessionFactory.getCurrentSession().get(clazz, id);

}

**public** Object uniqueQuery(String hql, Object[] parameters) {

// **TODO** Auto-generated method stub

Query query=**this**.sessionFactory.getCurrentSession().createQuery(hql);

//��?��ֵ

**if**(parameters!=**null** && parameters.length>0){

**for**(**int** i=0;i<parameters.length;i++){

query.setParameter(i, parameters[i]);

}

}

**return** query.uniqueResult();

}

**public** **int** queryPageCount(String hql, Object[] parameters, **int** pageSize) {

// **TODO** Auto-generated method stub

//��ȡrowCount

/\* List list=this.executeQuery(hql, parameters);

Iterator iteator=list.iterator();

if(iteator.hasNext()){

}\*/

Object obj=**this**.uniqueQuery(hql, parameters);

//System.out.println("obj value= "+ obj);//obj������rowConunt

**int** rowCount=Integer.*parseInt*(obj.toString());

**return** (rowCount-1)/pageSize+1;

}

**public** **void** delById(Class clazz,Serializable id) {

// **TODO** Auto-generated method stub

Session session=**this**.sessionFactory.getCurrentSession();

session.delete(**this**.findById(clazz, id));

}

**public** **void** update(Object object) {

// **TODO** Auto-generated method stub

**this**.sessionFactory.getCurrentSession().update(object);

}

}

# 4.开始更改从前的项目

//开始事物管理

//@Transactional

**public** **class** DepartmentService **extends** BasicService **implements** DepartmentServiceInter {

/\*

\* 下面这些有了基础接口之后就都不需要了

\*

\* @Resource

SessionFactory sessionFactory;

public SessionFactory getSessionFactory() {

return sessionFactory;

}

public void setSessionFactory(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

# 4.重量级解释：

## 4.1spring 中什么也不用变化，因为我们在 基础接口中添加了，之后，我们只要是继承他 那么他的所有的属行都会被使用

@Transactional

**public** **abstract** **class** BasicService **implements** BasicServiceInter {

@Resource