**SSH员工信息管理项目**

[一、准备操作](#5395-1558866921284)

[1.1、配置spring](#9254-1558866875852)

[1.2、配置Hibernate](#8381-1558866876061)

[1.3、配置struts2](#OFFICE20A5765975D34F76A27A61638B5BD2E9)

[二、实现功能](#6354-1558954877128)

[2.1、功能一---查看员工信息](#3083-1558956250281)

[2.2、功能二--删除操作](#8446-1558956281835)

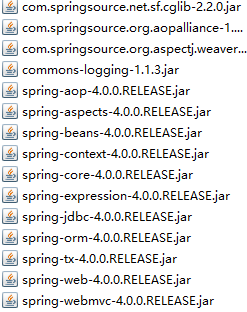
[2.3、功能三--添加员工](#8788-1558956281901)

[2.4、功能四--修改功能](#2929-1558956282033)

**一、准备操作**

**1.1、配置spring**

**添加jar包**



**配置web.xml文件**

**配置spring的核心配置文件**

**配置spring的核心配置文件**

**导入数据库资源文件**

**db.properties文件**

jdbc.user=root
jdbc.password=123456
jdbc.driverClass=com.mysql.jdbc.Driver
jdbc.jdbcUrl=jdbc:///SSH02

**导入文件**

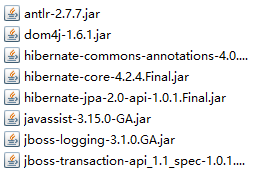
<!-- 导入资源文件 -->
<context:property-placeholder location="classpath:db,properties"/>

**配置c3p0数据源**

**配置sessionFactory**

**1.2、配置Hibernate**

**加入jar包**



**在类路径下编写hibernate核心配置文件**

**建立持久化类和其对应的xxx.hbm.xml文件**

**和spring进行整合**

**配置spring声明事务**

**配置hibernate事务管理器**

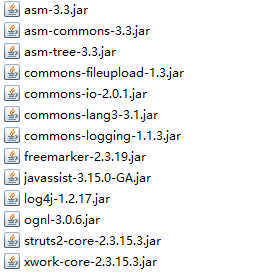
**配置事务属性**

**配置事务切入点**

<!-- 配置事务切入点 -->
<aop:config>
<aop:pointcut expression="exection(\* cn.amphobia.ssh02.service.\*.\*(..))" id="txPointcut"/>
<aop:advisor advice-ref="txAdvice" pointcut-ref="txPointcut"/>
</aop:config>

**1.3、配置struts2**

**加入jar包**



**在web.xml中配置Filter**

**配置struts2核心配置文件**

**整合spring**

**导入struts-spring的jar包**

clipboard.png

**在applicationContext-bean.xml中配置属性**

<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean id="employeeAction" class="cn.amphobia.ssh02.action.EmployeeAction" scope="prototype">
</bean>
</beans>

**二、实现功能**

**2.1、功能一---查看员工信息**

**创建EmployeeDao、EmployeeService查找所有员工信息**

**EmployeeDao**

public class EmployeeDao {
private SessionFactory sessionFactory;
public void setSessionFactory(SessionFactory sessionFactory) {
this.sessionFactory = sessionFactory;
}
public Session getSession() {
return this.sessionFactory.getCurrentSession();
}
/\*
\*
\* 查找所有员工信息
\*/
public List<Employee> getAll(){
String hql = "FROM Employee e LEFT OUTER JOIN FETCH e.department";
return getSession().createQuery(hql).list();
}
}

**EmployeeService**

public class EmployeeService {
private EmployeeDao employeeDao;
public void setEmployeeDao(EmployeeDao employeeDao) {
this.employeeDao = employeeDao;
}
/\*
\* 查找所有员工信息
\*/
public List<Employee> getAll(){
return employeeDao.getAll();
}
}

**创建EmployeeAction将员工信息提取出来，并发送到页面上**

public class EmployeeAction extends ActionSupport implements RequestAware {
private EmployeeService employeeService;
private Map<String, Object> request;
public void setEmployeeService(EmployeeService employeeService) {
this.employeeService = employeeService;
}
//重写setRequest方法，将获取到的值存放到request域中
public void setRequest(Map<String, Object> map) {
this.request = map;
}
/\*
\* 显示所有员工信息
\*
\*/
public String list() {
//将信息保存到request中，方便jsp页面获取
request.put("employees", employeeService.getAll());
return "list";
}
}

**配置相关文件**

**新建applicationContext-bean.xml将spring和struts联合**

**在struts.xml中配置页面信息**

**编写emp-list.jsp页面信息**

**结果显示**



**2.2、功能二--删除操作**

**编写EmployeeDao和EmployeeService的删除方法**

**EmployeeDao**

public void delete(Integer id) {
String hql = "DELETE FROM Employee e WHERE e.id = ?";
getSession().createQuery(hql).setInteger(0, id).executeUpdate();
}

**EmployeeService**

public void delete(Integer id) {
employeeDao.delete(id);
}

**在EmployeeAction中调用删除方法**

**在页面上用ajax编写删除的操作**

<script type="text/javascript">
$(function(){
//1. 点击 delete 时, 弹出 确定是要删除 xx 的信息吗 ? 若确定, 执行删除, 若不确定, 则取消
$(".delete").click(function(){
var lastName = $(this).next(":hidden").val();
var flag = confirm("确定要删除" + lastName + "的信息吗?");
if(flag){
var $tr = $(this).parent().parent();
//删除, 使用 ajax 的方式
var url = this.href;
var args = {"time":new Date()};
$.post(url, args, function(data){
//若 data 的返回值为 1, 则提示 删除成功, 且把当前行删除
if(data == "1"){
alert("删除成功!");
$tr.remove();
}else{
//若 data 的返回值不是 1, 提示删除失败.
alert("删除失败!");
}
});
}
//取消超链接的默认行为
return false;
});
})
</script>

**在EmployeeAction添加功能**

private InputStream inputStream;
public InputStream getInputStream() {
return inputStream;
}
public String delete() {
try {
employeeService.delete(id);
inputStream = new ByteArrayInputStream("1".getBytes("UTF-8"));
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
try {
inputStream = new ByteArrayInputStream("0".getBytes("UTF-8"));
} catch (UnsupportedEncodingException e1) {
e1.printStackTrace();
}
}
return "delete";
}

**在struct-xml配置文件**

**2.3、功能三--添加员工**

**编写员工录入的页面**

**编写DepartmentDao、DepartmentService功能**

**EmployeeDao**

public List<Department> getAll(){
String hql = "from Department";
return getSession().createQuery(hql).list();
}

**EmployeeService**

public List<Department> getAll(){
return departmentDao.getAll();
}

**编写EmployeeAction功能**

**继承ModelDriven,Preparable，并重写方法**

private InputStream inputStream;
public InputStream getInputStream() {
return inputStream;
}
/\*
\* 添加员工信息时显示的表单
\*/
public String input() {
request.put("departments", departmentService.getAll());
return INPUT;
}
public void prepareSave() {
model = new Employee();
}
public void prepare() throws Exception {}
private Employee model;
public Employee getModel() {
return model;
}

**注意：要配置拦截器**

**struct.xml**

**编写文件SSHDateConverter.java**

private DateFormat dateFormat;
{
dateFormat = new SimpleDateFormat("yyyy-MM-dd");
}
@Override
public Object convertFromString(Map context, String[] values, Class toClass) {
if(toClass == Date.class){
try {
return dateFormat.parse(values[0]);
} catch (ParseException e) {
e.printStackTrace();
}
}
return null;
}
@Override
public String convertToString(Map context, Object o) {
if(o instanceof Date){
return dateFormat.format((Date)o);
}
return null;
}
}

**编写文件xwork-conversion.properties**

java.util.Date=cn.amphobia.ssh02.converters.SSHDateConverter

**编写页面代码**

**编写实际功能**

**EmployeeDao**

public void saveOrUpdate(Employee employee) {
getSession().saveOrUpdate(employee);
}

**EmployeeService**

/\*
\* 添加成员
\*/
public void saveOrUpdate(Employee employee) {
employeeDao.saveOrUpdate(employee);
}

**EmployeeAction**

public String save() {
model.setCreateTime(new Date());
employeeService.saveOrUpdate(model);
System.out.println(model);
return SUCCESS;
}

**使用jQuery来判断用户名是否能用**

**配置ajax**

**配置判断用户名是否存在**

**EmployeeDao**

public Employee getEmployeeByLastName(String lastName) {
String hql = "FROM Employee e WHERE e.lastName = ?";
Query query = getSession().createQuery(hql).setString(0, lastName);
Employee employee = (Employee) query.uniqueResult();
System.out.println(employee.getDepartment().getClass().getName());
return employee;
}

**EmployeeService**

public boolean lastNameIsValid(String lastName) {
return employeeDao.getEmployeeByLastName(lastName) == null;
}

**EmployeeAction**

private String lastName;
public void setLastName(String lastName) {
this.lastName = lastName;
}
public String validateLastName() throws UnsupportedEncodingException {
if(employeeService.lastNameIsValid(lastName)) {
inputStream = new ByteArrayInputStream("1".getBytes("UTF-8"));
}else{
inputStream = new ByteArrayInputStream("0".getBytes("UTF-8"));
}
return "ajax-success";
}

**2.4、功能四--修改功能**

**在EmployDao、EmployeeService添加查询id的操作**

**EmployeeDao**

public Employee get(Integer id) {
return (Employee) getSession().get(Employee.class, id);
}

**EmployeeService**

public Employee get(Integer id) {
return employeeDao.get(id);
}

**在EmployeeAction中添加根据id查询对象的功能**

public String save() {
if(id == null) {
model.setCreateTime(new Date());
}
employeeService.saveOrUpdate(model);
// System.out.println(model);
return SUCCESS;
}
public void prepareInput() {
if(id != null) {
model = employeeService.get(id);
}
}
//根据id来判断save方法的model是new的还是从数据库获取的
public void prepareSave() {
if(id == null) {
model = new Employee();
}else {
model = employeeService.get(id);
}
}

**修改页面，设置用户名不可修改**