三大框架：

Struts框架

1. params拦截器： 请求数据封装

2. 类型转换/数据处理

3. struts配置

4. 文件上传/下载/国际化处理

5. 数据效验/拦截器

6. Ognl表达式

7. 数据回显/模型驱动/防止表单重复提交

Hibernate框架

1. Api简介

2. 映射

多对一/一对多: 部门与员工

多对多/一对一

集合映射/组件映射/继承映射

3. Hibernate 缓存

4. 常用属性：inverse/lazy/cascade

5. 对象状态：临时/持久/游离

Spring框架

1） Spring IOC容器 【Inversion Of Control,即控制反转】

创建对象/处理对象依赖关系

2） Aop编程

3） Spring声明式事务管理

4） 框架整合

Spring与Struts

Spring与Hibernate

5）Spring对jdbc、hibernate操作的封装

JdbcTemplate (dataSource)/ HibernateTemplate(SessionFactory)

目标:

案例 【软件项目声明周期】

# 1. 需求分析

系统概述：

企业人事管理系统！

要求对员工信息进行维护;

后台系统先登陆，才能操作员工： 添加/修改/删除

没有登陆，只能查看列表，不能操作！

功能分类:

【管理员模块:】

注册/登陆

【员工模块】

1) 添加一个员工, 指定添加的部门

2) 对指定的员工信息修改

3) 删除选择员工

4)列表展示

# 2. 设计

## 2.1 系统设计 【系统架构师/技术经理】

主要做下面的事情,

1) 搭建系统框架结构

(基于mvc结构应用)

2) 确定项目的关键点/难点

3) 确定引用组件、公用类的版本

Struts2.3

Hibernate3.6

Spring3.2

## 2.2 数据库设计

管理员表： t\_admin

员工表： t\_employee

部门： t\_dept

# 3. 代码

## 步骤分析

编码顺序:

1) 设计数据库: hib\_demo

建表： t\_admin/t\_employee/t\_dept

2) 建立web项目、引入jar文件、准备环境

…..

3) 设计javvabean、写映射

Admin.java 封装管理员

Employee.java 员工

Dept.java 部门

Admin.hbm.xml

Employee.hbm.xml

Dept.hbm.xml

4) Dao设计接口

IAdminDao.java 管理员模块

void save(Admin admin);

Admin findByAdmin(Admin admin);

IDeptDao.java 部门模块

List<Dept> getAll();

Dept findById(int id);

IEmployeeDao.java 员工模块

Void save(Employee emp);

Void update(Employee emp);

Void delete(int id);

Employee findById(int id);

List<Employee> getAll();

List<Employee> getAll(String employeeName);

5) Dao接口实现

6)Service接口设计

IAdminService.java 管理员模块

void register(Admin admin);

Admin login(Admin admin);

7)Service接口实现

8) Action实现

EmployeeAction.java 员工模块

AdminAction.java 管理员模块

9)jsp页面

Index.jsp/list.jsp 首页列表

<http://localhost:8080/项目> 首页列表

优化部分：

10) 用户登陆拦截器

UserInterceptor.java 检查是否登陆

只有登陆才能操作； 否则只能查看

11) Dao 操作优化

BaseDao.java 所有dao的通用方法，dao都必须继承此类

(反射泛型)

## 实现步骤代码：

|  |
| --- |
| **1) 设计数据库: hib\_demo** |
| CREATE TABLE t\_admin(  id INT PRIMARY KEY AUTO\_INCREMENT,  adminName VARCHAR(20),  pwd VARCHAR(20)  )  表： t\_dept/ t\_employee |
| **2) 建立web项目、引入jar文件、准备环境**  【struts相关jar】  commons-fileupload-1.2.2.jar  commons-io-2.0.1.jar  commons-lang3-3.1.jar  freemarker-2.3.19.jar  javassist-3.11.0.GA.jar  ognl-3.0.5.jar  struts2-core-2.3.4.1.jar  xwork-core-2.3.4.1.jar  【hibernate 相关 jar】  antlr-2.7.6.jar  commons-collections-3.1.jar  dom4j-1.6.1.jar  hibernate3.jar  hibernate-jpa-2.0-api-1.0.0.Final.jar  javassist-3.12.0.GA.jar  jta-1.1.jar  slf4j-api-1.6.1.jar  【Spring-core】  commons-logging-1.1.3.jar  spring-beans-3.2.5.RELEASE.jar  spring-context-3.2.5.RELEASE.jar  spring-core-3.2.5.RELEASE.jar  spring-expression-3.2.5.RELEASE.jar  【Spring-web】  spring-web-3.2.5.RELEASE.jar  struts2-spring-plugin-2.3.4.1.jar  【Spring-Aop】  aopalliance.jar  aspectjrt.jar  aspectjweaver.jar  spring-aop-3.2.5.RELEASE.jar  【Spring-orm】  c3p0-0.9.1.2.jar  mysql-connector-java-5.1.12-bin.jar  spring-orm-3.2.5.RELEASE.jar  spring-tx-3.2.5.RELEASE.jar  spring-jdbc-3.2.5.RELEASE.jar |
| **3) 设计javvabean、写映射** |
| **public** **class** Admin {  **private** **int** id;  **private** String adminName;  **private** String pwd;  } |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">  <hibernate-mapping package=*"cn.itcast.entity"*>  <class name=*"Admin"* table=*"t\_admin"*>  <id name=*"id"*>  <generator class=*"native"*></generator>  </id>  <property name=*"adminName"* length=*"20"*></property>  <property name=*"pwd"* length=*"20"*></property>  </class>  </hibernate-mapping> |
| **public** **class** Dept {  **private** **int** id;  **private** String name;  } |
| <hibernate-mapping package=*"cn.itcast.entity"*>  <class name=*"Dept"* table=*"t\_dept"*>  <id name=*"id"* column=*"deptId"*>  <generator class=*"native"*></generator>  </id>  <property name=*"name"* column=*"deptName"*></property>  </class>  </hibernate-mapping> |
|  |
| **public** **class** Employee {  **private** **int** id;  **private** String empName;  **private** **double** salary;  **private** Dept dept;  } |
| <hibernate-mapping package=*"cn.itcast.entity"*>  <class name=*"Employee"* table=*"t\_employee"*>  <id name=*"id"* column=*"empId"*>  <generator class=*"native"*></generator>  </id>  <property name=*"empName"*></property>  <property name=*"salary"*></property>  <!-- 多对一 -->  <many-to-one name=*"dept"* column=*"dept\_id"* class=*"Dept"*></many-to-one>  </class>  </hibernate-mapping> |
|  |
| **4) Dao设计接口** |
| **public** **interface** IAdminDao {  /\*\*  \* 保存  \* **@param** admin 管理员对象  \*/  **void** save(Admin admin);  /\*\*  \* 根据管理员信息查询  \* **@param** admin 管理员对象  \* **@return** 返回查询后的结果  \*/  Admin findByAdmin(Admin admin);  } |
| **public** **interface** IDeptDao {  /\*\*  \* 查询全部  \* **@return** 返回全部信息  \*/  List<Dept> getAll();  /\*\*  \* 根据主键查询  \* **@param** id 主键  \* **@return** 返回查询后的结果  \*/  Dept findById(**int** id);  } |
| **public** **interface** IEmployeeDao {  /\*\*  \* 保存员工  \* **@param** emp  \*/  **void** save(Employee emp);  /\*\*  \* 跟新员工信息  \* **@param** emp  \*/  **void** update(Employee emp);  /\*\*  \* 根据主键删除  \* **@param** id  \*/  **void** delete(**int** id);  /\*\*  \* 根据主键查询  \* **@param** id  \* **@return**  \*/  Employee findById(**int** id);  /\*\*  \* 查询全部  \* **@return**  \*/  List<Employee> getAll();  /\*\*  \* 根据员工名称条件查询  \* **@param** employeeName  \* **@return**  \*/  List<Employee> getAll(String employeeName);  } |
| Dao实现、service省略 |
| ….. |
|  |

作业：

1) 员工CRUD操作

2) 根据条件查询员工

3) 新建一个管理员