# 附件3 网上招聘系统详细设计

**1.导言**

**1.1 目的**

该文档的目的是描述《网上招聘系统》项目的详细设计，其主要内容包括：

·系统功能简介；

·系统详细设计简述；

·各个模块的三层划分；

·最小模块组件的伪代码。

本文档的预期的读者是：

·开发人员；

·项目管理人员；

·测试人员。

**1.2 范围**

该文档定义了系统的各个模块和模块接口，但未确定单位的具体实现，这部分内容将在程序实现中确定。

**1.3 编写说明**

HR：Human Resource（人力资源管理）的缩写。

JSP：Java Server Page（Java服务器页面）的缩写，一个脚本化的语言。

MVC：Model-View-Control（模式—视图—控制）的缩写，表示一个三层的结构体系。

**1.4 术语定义**

Struct：一种框架体系结构。

**1.5 引用标准**

[1]《企业文档格式标准》V1.1，北京长江软件有限公司

[2]《软件概要设计报告格式标准》V1.1，北京长江软件有限公司软件工程过程化组织

**1.6 参考资料**

[1]《实战struct》(美)Ted Husted

机械工业出版社出版

**1.7 版本更新信息**

本文档版更新记录如表C—1所列。

**表C-1 版本更新记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 |
| 000 | 2022.10.12 | 1.0 | 全部 | 初始发布版本 |

2.系统设计概述

根据《网上招聘系统》的概要设计，本系统按照功能分解，可以分为客户端子系统和管理端子系统。根据页面流的设计，管理端系统分为登录管理，题库管理、问卷管理，职位发布，简历管理、面试管理7个模块，它们的关系如图C-1所示，以下将对各个部分分别进行详细设计。

网上招聘系统

管理端系统

客户端系统

登录管理

用户管理

题库管理

问卷管理

职位发布

简历管理

面试管理

**图C-1 模块设计**

**3. 详细设计概述**

由于本系统采用了基于struct体系结构的设计，即MVC的三层设计模式，并采用面向对象的Java语言以及JSP的脚本语言实现。在整个开发过程中，尽可能采用复用的原则，例如采用标签库，统一数据库的基本操作，统一结果显示等。

本文档的详细设计主要地按照struct的MVC的三个层次分别编制视图层、控制层和模型层模块的伪代码，为下一步的编码提供基础。伪代码（Pseudocode）是一种算法描述语言。使用伪代码的目的是为了使被描述的算法可以容易地以任何一种编程语言实现。因此，伪代码必须结构清晰，代码简单，可主动性好，并且类似自然语言。

**4. 用户登录模块的详细设计**

用户登录管理主要实现了对管理端用户登录和验证和权限管理功能，用户登录管理的视图层、控制层和模型层三个层次的模块如表C-7所示。

表C-2 用户登录管理的三层模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 视图 | | | 控制器 | 模型 |
| JSP组件 | ActionFormBean | 客户标签 |
| index.jsp | LoginForm | Struct Html标签 | LoginAction | DBUtil |
| main.jsp |  | Struct Bean标签 |  | Permission |
| center.jsp |  | Struct Logic标签 |  | Userbean |
| top.jsp |  | 用户自定义CV标签库的标签 |  |  |
| left.jsp |  |  |  |  |

****

**图C-2 登录系统顺序图**

**5. 客户端模块的详细设计**

客户端模块主要实现应聘者提交简历的功能，客户端模块的视图层、控制层和模型层三个层次的模块如表C-9所示。

表C-3 客户端的三层模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 视图 | | | 控制器 | 模型 |
| JSP组件 | ActionFormBean | 客户标签 |
| index.jsp | JobForm | Struct Html标签 | EnAction | DBUtil |
| joblist.jsp | JobDetailForm | Struct Bean标签 | JobDetailAction | CVDataBean |
| jobdetail.jsp | CVbasicinfoForm | Struct Logic标签 | ApplyJobAction |  |
| CVbasicinfo.jsp | CareerForm | 用户自定义CV标签库的标签 | AddbasicinfoAction |  |
| career.jsp | EducationForm |  | ProcesscareerAction |  |
| education.jsp | SkillsForm |  | AddcareerAction |  |
| skills.jsp | RelationForm |  | BackcareerAction |  |
| relation.jsp | AddtionForm |  | ProcesseducationAction |  |
| addtion.jsp | CVallForm |  | AddeducationAction |  |
| CVall.jsp | ExamForm |  | BackeducationAction |  |
| exam.jsp |  |  | ProcessskillsAction |  |
|  |  |  | AddskillsAction |  |
|  |  |  | BackskillsAction |  |
|  |  |  | ProcessrelationAction |  |
|  |  |  | AddrelationAction |  |
|  |  |  | BackrelationAction |  |
|  |  |  | AddaddtionAction |  |
|  |  |  | BackaddtionAction |  |
|  |  |  | CVallAction |  |
|  |  |  | BackCVallAction |  |
|  |  |  | SubmitAction |  |
|  |  |  | BacksubmitAction |  |

**5.1视图层**

客户管理视图层包括JSP组件、Form组件以及标签库等。共有11个JSP文件，共有10个Form文件，每个Form模块的伪代码详见光盘。

**5.2 控制层**

客户端管理控制层共有22个Action文件，

**5.3 模型层**

客户端管理控制层共有2个Bean文件，



**图C-3 面试管理的页面流程图**

**（其他模块分析见关盘中模型文件）**

**6. 复用的模块**

复用是本系统设计的一个重要原则，尤其是struct体系本身也遵循了复用的原理，本系统复用的部分主要包括标签库、数据库基本操作以及结果的显示等环节。

**6.1 客户标签库**

客户标签库主要包括：

**1）Struct标准标签库**。本系统中使用了Struct Html标签、Struct Bean标签和Struct Logic，它们是Struct标准标签库。

**2）自定义标签库。**CV标签是系统自定义的标签库，表V-10是CV标签库的标签模块。

表C-4 CV标签库的标签模块

|  |  |  |
| --- | --- | --- |
| 序号 | 标签 | 功能描述 |
| 1 | ValidateSessionTag | 身份验证 |
| 2 | PageBeanTag | 分页 |

下面是身份验证的伪代码说明。用户登录后进入的每个页面，为了安全的考虑，管理端都需要进行用户验证，以防止非法登录，验证每个页面的JSP都可以标签的方式进行复用。

/\*\*

\*@System:Online CV System

\*@Version:1.0

\*@Copyright@ 2022 by changjiang company, Inc. All Rights Reserved.

\*@Class:ValidateSessionTag

\*@Summary:对用户的身份验证

\*@Create:2022.03.28郭小明

\*Update:

\*/

Package tag;

import javax.servlet.http.HttpSession;

import javax.servlet.jsp.JspException;

import javax.servlet.jsp.tagext.TagSupport;

import org.apache.commons.logging.log;

import org.apache.commons.logging.logFactory;

/\*\*

\*ValidateSessionTag是客户端标签，以判断用户是否具有访问当前页面的权限

\*通过保持、判断一个session来实现

\*/

public final class ValidateSessionTag extends TagSupport

{

private String name="username";

private String page="/login.jsp";

private Log log=LogFactory.getLog(this.getClass().getName());

public int doEndTag()throws JspException

{

boolean valid=false;

HttpSession session=pageContext.getSession();

//判断session中是否有用户信息存在

//如果用户存在则不做任何处理

//用户不存在则跳转到指定的页面

}

public int doStartTag()throws JspException

{

return(SKIP\_BODY);

}

//获取用户信息

public String getName()

{

......

}

//获取用户当前访问的页面

public String getpage()

{

......

}

//如果用户没有当前页面的访问权限，则返回到指定的页面，设置这个指定页面

public void release()

{

......

}

//设置用户信息

public void setName(String name)

{

......

}

//设置当前页面

public void setPage(String page)

{

......

}

}

其他详见文件WSZP-PD-DD-Reuse-CVTag.doc,它对每个模块的伪代码都有比较详细的描述。

**6.2 数据库的基本操作**

所有的模块都需要进行数据库的操作，应该有一个通用的、高质量的数据库操作模块，其实所有的模型层的模块也都在调用这个模块。这个模块的名称是DBUtil。下在面是它的伪代码描述。

/\*\*

\*@System:Online CV System

\*@Version:1.0

\*@Copyright @ 2022 by changjiang company,Inc, All Rights Reserved.

\*@Class:DB

\*@sUMMAYT:取得数据库连接并对数据库进行查询！增加记录，删除记录操作

\*@Create:2022.04.27李林

\*@Update:

\*/

package com.changjiangcompany.xtructs.javashare;

import java.sql.\*;

import java.util.\*;

import com.microsoft.jdbcx.sqlserver.SQLServerDataSource;

public class DB{

Private Statement stmt=null;

Private ResultSet rs=null;

public DBConnect()

{

//初始化连接

......

}

/\*\*

\*方法：executeQuery

\*描述：执行查询记录操作

\*输入参数：String strsql要执行操作的sql语句

\*返回：ArrayList ArrayRs查询的结果

\*异常：SQLException

\*/

public ArrayList executeQuery(SQLServerDataSource source,String strSql)throws SQLException

{

/\*

\*连接数据库

\*/

/\*

\*执行SQL的查询语句

\*/

......

/\*

\*将每条记录写入ArrayList中

\*/

......

}

/\*\*

\*方法：executeInsert

\*描述：执行插入查录操作

\*输入：String strsql要执行操作的sql语句

\*返回：boolean类型，插入操作是否正确执行

\*异常处理：SQLException

\*/

public boolean executeInsert(SQLServerDataSource source,String strSql)throws SQLException

{

/\*

\*连接数据库

\*/

/\*

\*执行SQL的查询语句

\*/

......

/\*

\*将每条记录写入ArrayList中

\*/

......

}

/\*\*

\*方法：executeUpdate

\*描述：执行更新操作

\*输入：String strsql要执行操作的sql语句

\*返回：int更新操作的记录数

\*异常处理：SQLException

\*/

public int executeUpdate(SQLServerDataSource source,String strSql)throws SQLException

{

/\*

\*连接数据库

\*/

/\*

\*执行SQL的修改语句

\*/

......

/\*

\*返回SQL修改语句执行的记录数

\*/

......

}

/\*\*

\*方法：executeDelete

\*描述：执行删除操作

\*输入：String strsql要执行操作的sql语句

\*返回：int删除操作的记录数

\*异常处理：SQLException

\*/

public int executeDelete(SQLServerDataSource source,String strSql)throws SQLException

{

/\*

\*连接数据库

\*/

/\*

\*执行SQL的删除语句

\*/

......

/\*

\*返回SQL删除语句记录数

\*/

}

**6.3 结果显示**

为了保证系统显示结果的一致性和统一性，系统中所有的结果显示要调用统一的结果显示页在，表C-5是结果显示一览表。

表C-5 结果显示页面一览表

|  |  |  |
| --- | --- | --- |
| 序号 | 模块名 | 功能说明 |
| 1 | SuccessResult | 成功登录 |
| 2 | WrongResult | 结果错误 |
| 3 | Common | 操作提示 |
| 4 | Top | 主页面的头部 |
| 5 | Bottom | 主页面的脚部 |
| 6 | Left | 主页面的左部 |

保个模块详见文件WSZP-PD-DD-Reuse-ResultShow.doc，它对每个模块的伪代码都有比较详细的描述。

**7. 配置文件**

Struct应用采用两个基于XML的配置文件来配置应用，这两个配置文件为web.xml和strcut-config.xml。web.xml适用于所有的Java Web应用，它是Web应用的发布描述文件，在Java Servlet规范中对它做了定义，对于struct应用，在web.xml文件中除了配置Java Web应用的常规信息，还应该配置和struct相关的特殊信息。strcut-config.xml是struct应用专有的配置文件，事实上，也可以根据需要将这个配置文件换为其他的文件名。

下面给出本项目中web.xml配置文件和strcut-config.xml配置文件的配置步骤和范围，在编码实施过程中，由开发经理指派专人负责所有文件的具体配置和协调。

**7.1 web.xml配置文件**

下面讲述web.xml配置文件的配置步骤和范围。

1）配置struct的ActionServlet：在web.xml中配置ActionServlet，主要包括声明ActionServlet（即配置<Servlet>元素）、运行时环境的初始化配置（即配置<Servlet>元素的<init-param>子元素）和指定ActionServlet可以处理哪些URL（即配置<Servlet-mapping>元素）。例如：

<servlet>

<servlet-name>action</servlet-name>

<servlet-class>org.apache.structs.action.ActionServlet</servlet-class>

<init-param>

<param-name>config</param-name>

<param-value>/WEB-INF/struct-config.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>action</servlet-name>

<url-pattern>\*.do</url-pattern>

<servlet-mapping>

2）配置欢迎文件清单：当客户访问Web应用时，如果没有指定具体的文件名，Web容器可以自动调用Web应用的欢迎文件。<welcome-file-list>元素用来设置欢迎文件清单。例如：

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

<welcome-file>login.jsp</welcome-file>

</welcome-file-list>

3）配置错误处理：在系统运行过程中如果错误发生而Struct框架不能处理这种错误时，就将错误抛给Web容器，为了避免让用户看到原始的错误信息，可以配置<error-page>元素及其子元素<error-code>和<exception-type>等。例如：

<error-page>

<error-code>404</error-code>

< location >/common/404.jsp</location>

</error-page>

<error-page>

<exception-type>java.io.IOException</exception-type>

<location>/common/system\_ioerror.jsp</location>

</error-page>

4)配置标签库：Struct框架提供了一些实用的客户化标签库，如果在应用中使用了标签库，那么必须在配置<taglib>元素中配置，它有两个子元素，<taglib-url>和<taglib-location>。例如：

<taglib>

<taglb-url>/WEB-INF/structs-bean.tld</taglib-url>

<taglib-location>/WEB-INF/structs-bean.tld</taqlib-location>

</taglib>

13.2 struct-config.xml配置文件

Struct框架在启动的时候会读入其配置文件，根据它来创建和配置各种组件。Struct配置文件使得开发者可以灵活地组装和配置各个组件，提高了应用软件的可扩展性和灵活性。

<structs-config>有8个子元素:

·<data-sources>;

·<form-beans>;

·<global-exceptions>;

·<global-forwards>;

·<action-mappings>;

·<controller>;

·<message-resources>;

·<plug-in>.

在配置文件中，必须按照这个先后顺序来配置各个子元素。

1）<data-sources>元素：<data-sources>元素用来配置应用所需要的数据源，数据源负责建立与特定数据库的连接。<data-sources>元素包含零个或者多个<data-sources>子元素，它配置特定的数据源，可以包含多个<set-property>。例如：

<data-sources>

<data-source key=“sqlserver”type=“org.apache.commons.dbcp.jdbc.sqlserver.SQLServerDriver”/>

<set-property property=“driverClassName”value=“com.microsoft.jdbc.sqlserver.SQLServerDriver”/>

< set-property property=“url”value=“jdbc:Microsoft:sqlserver://127.0.0.1:1433;DatabaseName=onlineCV”/>

< set-property property=“MaxACTIVE”VALUE=“5/”>

< set-property property=“username”value=“sa”/>

< set-property property=“password”value=“xiaobo633601”/>

< set-property property=“autoCommit”value=“true”/>

</data-source>

</data-sources>

2）<form-beans>元素：<form-beans>元素用来配置多个ActionForm Bean。<form-beans>元素包含零个或者多个<form-beans>子元素，每个<form-beans>元素又包含多个属性。例如：

<form-beans>

<form-bean name=“AddJobForm”type=“org.apache.structs.action.DynaActionForm”>

<form-property name=“JobName”type=“java.lang.String”/>

<form-property name=“pubDate”type=“java.lang.String”/>

<form-property name=“EndDate”type=“java.lang.String”/>

<form-property name=“Requirements”type=“java.lang.String”/>

<form-property name=“Description”type=“java.lang.String”/>

<form-property name=“JobNum”type=“java.lang.Integer”/>

<form-beans>

<form-bean name=“JobDetailForm”type=“org.apache.structs.action.DynaActionForm”>

<form-property name=“JobID”type=“java.lang.Integer”/>

<form-property name=“JobName”type=“java.lang.String”/>

<form-property name=“PubDate”type=“java.lang.String”/>

<form-property name=“EndDate”type=“java.lang.String”/>

<form-property name=“Description”type=“java.lang.String”/>

<form-property name=“Requiremente”type=“java.lang.String”/>

<form-property name=“JobNum”type=“java.lang.Integer”/>

<form-bean>

<form-beans>

3）<global-exceptions>元素：<global-exceptions>元素用于配置异常处理，<global-exceptions>元素可以包含零个或者多个<exception>元素。例如：

<global-exceptions>

<exception>

Key=“error.invalidlogin”

Path=“com/relogin.jsp”

Scope=“request”

<Type=“netstore.framework.exceptions.InvalidLoginException”/>

<global-exceptions>

4）<global-forwards>元素：<global-forwards>元素用来声明全局的转发关系，实现Web组件之间的相互转发。<global-forwards>元素由零个或者多个< forwards>元素组成。< forwards>元素用于将一个逻辑名映射到特定的URL，这样Action类或者JSP文件无须指定实际的URL，减弱了控制组件和视图组件之间的耦合。例如：

<gloal-forwards>

<forward name=“toaddjobpre”path=“position/addjob.jsp”/>

<forward name=“tojobdetail”path=“position/jobdetail.jsp”/>

<forward name=“tojobsearch”path=“position/jspsearchetail.jsp”/>

<forward name=“toaddjob”path=“position/joblist.do”/>

<forward name=“towrong”path=“position/wrong.jsp”/>

<forward name=“tojoblist”path=“position/joblist.jsp”/>

<forward name=“tomodifyjob”path=“position/modifyjob.jsp”/>

<forward name=“toaddjobvalueexampre”path=“position/selectexam.jsp”/>

<forward name=“toaddjobvalueexam”path=“position/addjob.jsp”/>

<forward name=“toreturn”path=“position/addjobl.jsp”/>

<forward name=“toaddjobexampre”path=“position/selectjobexam.jsp”/>

<forward name=“toaddjobexam”path=“position/addjobl.jsp”/>

<forward name=“toaddjobexam”path=“position/addjobl.jsp”/>

<forward name=“toaddfulljob”path=“position/joblist.do”/>

<forward name=“tomodifyjobsubmit”path=“position/joblist.do”/>

<forward name=“toupdate”path=“position/addjobl.jsp”/>

<forward name=“todeljob”path=“position/joblist.do”/>

<forward name=“toupdatemodifyjsp”path=“position/modifyjobsubmit.jsp”/>

<forward name=“tomodifyjobvalueexampre”path=“position/modifyselectvalueexam.jsp”/>

<forward name=“toomodifyjobexampre”path=“position/modifyselectexam.jsp”/>

<forward name=“tomodifyjobvalueexam”path=“position/modifyjobsubmit.jsp”/>

<forward name=“tomodifyjobexam”path=“position/modifyjobsubmit.jsp”/>

<forward name=“tomodifyupdatejobexam”path=“position/modifyjobsubmit.jsp”/>

<forward name=“tomodifydeljobexam”path=“position/modifyjobsubmit.jsp”/>

<gloal-forwards>

5）<action-mappings>元素：<action-mappings>元素包含零个或者多个<action >元素，<action >元素描述了从特定的请求路径到相应的Action类的映射。<action >元素可以包含多个<exception>和<forward>子元素，它们分别配置局部的异常处理及请求转发，仅被当前的Action所访问。例如：

<action-mappings>

<action attribute=“JobDetailForm”name=“JobDetailForm”path=“/position/jobdetail”scope=“request”type=“com.changjiangcompany.structs.action.JobDetailAction”></action>

<action attribute=“JobSearchForm”input=“position/joblist.jsp”name=“JobSearchForm”

path=“position/jobsearch” scope=“request”type=“com.changjiangcompany.structs.action.

JobSearchAction”></action>

<action-mappings>

6）<contoller>元素：<contoller>元素用于配置ActionServlet。例如：

<controller processorClass=“share.MyRequestProcessor/”>

7）<message-resources>元素：<message-resources>元素用于配置Resource Bundle，Resource Bundle用于存放本地化消息文本。例如：

<message-resources parameter=“com.yourcompany.structs.ApplicationResources”/>

8）<plug-in>元素：<plug-in>元素用于配置structs插件，如果没有插件可以不写。