|  |  |
| --- | --- |
| **项目编号** | **200602006** |
| **文档编号** | **12** |
| **密 级** | **内部** |

**停车场管理系统详细设计**

**V1.0**

**软件工程实验**

评 审 日 期： 2019年11月3日

**目 录**

[1.1 目的 3](#_Toc26089376)

[1.2 范围 3](#_Toc26089377)

[1.3 缩写说明 3](#_Toc26089378)

[1.4 术语定义 3](#_Toc26089379)

[1.5引用标准 4](#_Toc26089380)

[1.6 参考资料 4](#_Toc26089381)

[1.7 版本更新信息 4](#_Toc26089382)

[2 系统设计概述 4](#_Toc26089383)

[3 详细设计概述 5](#_Toc26089384)

[4 客户端模块的详细设计 5](#_Toc26089385)

[4.1视图层 6](#_Toc26089386)

[4.2控制层 6](#_Toc26089387)

[4.3控制层 6](#_Toc26089388)

[５登录管理模块的详细设计 6](#_Toc26089389)

[5.1视图层 7](#_Toc26089390)

[5.2控制层 7](#_Toc26089391)

[5.3模型层 7](#_Toc26089392)

[5.4分析及功能实现 7](#_Toc26089393)

[6 用户管理模块的详细设计 9](#_Toc26089394)

[6.1视图层 9](#_Toc26089395)

[6.2控制层 9](#_Toc26089396)

[6.3模型层 9](#_Toc26089397)

[6.4分析及功能实现 10](#_Toc26089398)

[7停车场状态管理模块的详细设计 10](#_Toc26089399)

[7.1视图层 10](#_Toc26089400)

[7.2控制层 11](#_Toc26089401)

[7.3模型层 11](#_Toc26089402)

[7.4分析及功能实现 11](#_Toc26089403)

[8 车辆信息管理模块的详细设计 14](#_Toc26089404)

[8.1视图层 14](#_Toc26089405)

[8.2控制层 14](#_Toc26089406)

[8.3模型层 15](#_Toc26089407)

[8.4分析及功能实现 15](#_Toc26089408)

[9 配置文件 16](#_Toc26089409)

[9.1 Web.xml配置文件 16](#_Toc26089410)

[13.2 strcut-config*.*Xml配置文件 17](#_Toc26089411)

**1. 导言**

## 1.1 目的

该文档的目的是描述《停车场管理系统》项目的详细设计，其主要内容包括：

1. 系统功能简介
2. 系统详细设计简述
3. 各个模块的三层划分
4. 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，但未确定单元的具体实现，这部分内容将在实现中确定。

## 1.3 缩写说明

*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

[2] 《软件详细设计报告格式标准》 V1.1

## 1.6 参考资料

[1] 《实战ｓｔｒｕｃｔ》 〔美〕Ｔｅｄ　Ｈｕｓｔｅｄ

机械工业出版社

## 1.7 版本更新信息

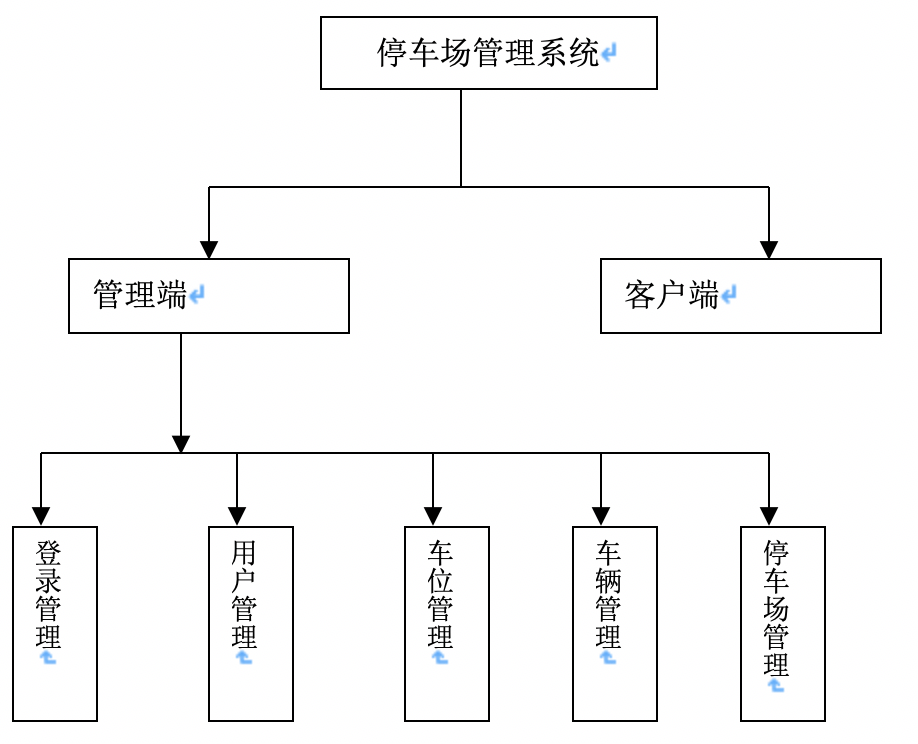
本文档版本更新记录如表Ｃ-1：

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 | 2019.11.1 | 1.0 | 全部 | 初始发布版本 |

# 2 系统设计概述

根据《停车场管理系统》的概要设计，本系统按照功能角度分解，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，管理端系统分为用户登录、用户管理、车位管理、车辆管理、停车场管理5个模块，他们的关系如图Ｃ-１，以下将分小节对各个部分分别进行详细设计。



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，即采用ＭＶＣ的三层设计模式，采用面向对象的Ｊａｖａ语言以及ＪＳＰ的脚本语言。所以，基本采用面向对象的设计方法。在整个的开发过程中，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是按照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，可读性好，并且类似自然语言。

# 4 客户端模块的详细设计

客户端模块主要实现用户泊车的功能，客户端模块的Ｓｔｒｕｃｔ视图层、控制层和模型层三个层次的模块如表Ｃ－２所示。

表Ｃ－２：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  InputNum.jsp spot.jsp  path.jsp pay.jsp | InputNumForm SpotForm  PathForm  PayForm | Struct Html标签  Struct Bean标签  Struct Logic标签  用户自定义CV标签库的标签 | EnAction  SpotAction  PathAction  PayAction  BackSpotAction  BackPathAction  BackPayAction  BackExitAction | DBUtil  CVDataBean |

## 4.1视图层

客户端管理视图层包括ＪＳＰ组件、Form组件以及标签库等。共有6个ＪＳＰ文件，共有4个Form文件，标签库详见12.1描述。

## 4.2控制层

客户端管理控制层共有8个Action文件。

## 4.3控制层

客户端管理模型层共有2个Bean文件。

# ５登录管理模块的详细设计

登录管理模块负责管理端用户的登录。管理端用户都是通过登录界面进入管理端的，用户输入用户名和密码进入管理界面首页，提供了进入功能面板的接口，并根据用户的权限在首页中列出相应的操作功能。

表Ｃ－６：登录管理模块的三层模块

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

## 5.1视图层

登录管理视图层包括ＪＳＰ组件、Form组件以及标签库等。共有６个ＪＳＰ文件,共有1个Form文件。

## 5.2控制层

登录管理控制层共有1个Action文件。

## 5.3模型层

登录管理模型层共有3个Ｂｅａｎ文件。

## 5.4分析及功能实现



功能描述：登陆页面需要用户名、密码进行登录。登录的同时对输入的用户名和密码进行验证，首先我们要保证用户名和密码不能为空。其次是对数据库的验证，系统在数据库中搜索用户输入的用户名是否存在，若不存在，则提示出错。并且重新登录，系统不允许两个用户同时登录，该操作在一定程度上使系统的安全性有所提高。

登录主要代码位置：TestPark/src/DAL/login.java。主要封装了对登录数据的操，该类中有三个方法分别用于检查用户登录信息是否合法，根据用户编号和角色编号获取用户名和角色信息。

Public class Login {

publicboolean checkLogin(String user\_id,String user\_pwd)

{

String sqlCmd="select count(\*) from user where user\_id=? and user\_pwd=?";

Object[] objList=new Object[2];

objList[0]=user\_id;

objList[1]=user\_pwd;

String result=SQLUtil.*excuteScalar*(sqlCmd,objList).toString(); if(result.equals("1"))

{

returntrue;

}

else {

returnfalse;

}

}

public String getName(String user\_id)

{

String sqlCmd="select user\_name from user where user\_id='"+user\_id+"'";

String result=SQLUtil.*excuteScalar*(sqlCmd, null).toString();

return result;

}

public String getSysLevel(String user\_id)

{

String sqlCmd="select role\_id from user where user\_id='"+user\_id+"'";

String result=SQLUtil.*excuteScalar*(sqlCmd, null).toString();

return result;

}

}

# 6 用户管理模块的详细设计

在用户管理中系统管理员对用户进行增删改查，可以进行权限设置。具有不同权限的用户进入不同的主界面，界面左侧栏中的图标数有所不同，具体的图标与用户所具有的权限对应。在用户管理中可以增加或删除用户，编辑用户名，用户密码，修改用户权限等。

表Ｃ－７：用户管理模块的三层模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 视图 | | | 控制器 | 模型 |
| JSP组件 | ActionFormBean | 客户标签 |
| main.jsp userlist.jsp  userdetail.jsp adduser.jsp | UserListForm UserDetailForm  ModifyUserForm AddUserForm  DelUserForm | Struct Html标签  Struct Bean标签  Struct Logic标签  用户自定义CV标签库的标签 | |  | | --- | | UserListAction | | UserDetailAction | | ModifyUserAction | | SubmitModifyAction | | AddUserAction | | SubmitAddAction | | DelUserAction | | DBUtil  Userbean |

## 6.1视图层

用户管理视图层包括JSP组件、Form组件以及标签库等。共有4个ＪＳＰ文件，共有5个Form文件。

## 6.2控制层

用户管理控制层共有7个Action文件。

## 6.3模型层

用户管理模型层共有3个Bean文件。

## 6.4分析及功能实现

功能描述：该模块是对系统信息的一个管理，添加角色输入编号和名称，后台数据库进行验证，若角色没有存在则添加数据成功，跳转到管理界面。管理角色信息可以根据用户的编号，角色名称进行查询操作、编辑、和删除操作。添加用户信息输入用户编号、名称、昵称、姓名、密码电话点击确定进行后台数据库的验证，若成功添加则跳转到管理界面。管理用户界面可根据用户编号、角色名称、用户名称、真实姓名进行查询、编辑和删除操作。



# 7停车场状态管理模块的详细设计

停车场状态管理是对车位信息库和车辆信息库的维护和管理，对状态库的车位信息进行的增、删、改的功能，及提供泊车库中各车位的详细信息。

停车场状态管理的Ｓｔｒｕｃｔ视图层、控制层和模型层三个层次的模块如表Ｃ－８所示。

表Ｃ－８：简历管理模块的三层模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 视图 | | | 控制器 | 模型 |
| JSP组件 | ActionFormBean | 客户标签 |
| inlist.jsp  spotlist.jsp  parkdetail.jsp  modifySpot.jsp  modifyPrice.jsp | showSpotForm  parkDetailForm  DelQuestionForm  modifyPriceForm | Struct Html标签  Struct Bean标签  Struct Logic标签  用户自定义CV标签库的标签 | showSpotAction  parkDetailAction  modifySpotAction  modifyPriceAction | DBUtil  Spotbean  Parkbean |

## 7.1视图层

停车场状态管理视图层包括JSP组件、Form组件以及标签库等。共有5个JSP文件，共有4个Form文件。

## 7.2控制层

停车场状态管理控制层共有4个Action文件。

## 7.3模型层

停车场状态管理模型层共有３个Bean文件。

## 7.4分析及功能实现

添加停车位信息如图所示：



管理停车位信息如图所示：

功能描述：该模块是对车位的一个管理。点击添加车位信息，输入编号选择是A区还是B区，添加成功则会跳转到管理界面。管理界面可根据车位ID、车位编号、所属区域、车位备注进行查询，还可对车位信息进行编辑和删除。

对车位数据的操作封装在：seat.java类中。该类中主要封装了分页和对车位信息的增、删、改、查等方法。首先获取车位信息列表，然后获取未分配的车位列表，根据查询条件获取获取分页后的信息列表，数据的更新、插入、删除，获取分页总数，根据查询条件获取分页总数。

publicclass Seat {

public List<Object> getEntity()

{

String sqlCmd="select \*from Seat";

return DBUtil.SQLUtil.*executeQuery*(sqlCmd, null);

}

public List<Object> getNoUseSeat()

{

String sqlCmd="SELECT \*FROM Seat WHERE seat\_id NOT IN(SELECT seat\_id FROM card)";

return DBUtil.SQLUtil.*executeQuery*(sqlCmd, null);

}

public List<Object> getEntity(int page)

{

int size=(page-1)\*15;

String sqlCmd="select \*from Seat limit "+size+",15";

return DBUtil.SQLUtil.*executeQuery*(sqlCmd, null);

}

public List<Object> getEntityByWhere(String sqlWhere,int page)

{

int size=(page-1)\*15;

String sqlCmd="select \*from Seat where "+sqlWhere+" limit "+ size+",15";

return DBUtil.SQLUtil.*executeQuery*(sqlCmd, null);

}

publicint deleteEntity(String seat\_id)

{

String sqlCmd="delete from Seat where seat\_id='"+seat\_id+"'";

return DBUtil.SQLUtil.*executeNonQuery*(sqlCmd, null);

}

public List<Object> getEntityById(String seat\_id)

{

String sqlCmd="select \*From Seat where seat\_id='"+seat\_id+"'";

return DBUtil.SQLUtil.*executeQuery*(sqlCmd, null);

}

publicint updateEntity(String seat\_id,String seat\_num,String seat\_section,String seat\_state,String seat\_tag)

{

String sqlCmd="Update Seat set seat\_num='" + seat\_num + "',seat\_section='" + seat\_section + "',seat\_state='" + seat\_state + "',seat\_tag='" + seat\_tag + "' where seat\_id='"+seat\_id+"'";

return SQLUtil.*executeNonQuery*(sqlCmd, null);

}

publicint insertEntity(String seat\_id,String seat\_num,String seat\_section,String seat\_state,String seat\_tag)

{

String sqlCmd="Insert into Seat values('" + seat\_id + "','" + seat\_num + "','" + seat\_section + "','" + seat\_state + "','"+seat\_tag+"')";

return SQLUtil.*executeNonQuery*(sqlCmd, null);

}

publicboolean checkExist(String seat\_id)

{

String sqlCmd="select count(\*) from Seat where seat\_id='"+seat\_id+"'";

if(1==Integer.*parseInt*(SQLUtil.*excuteScalar*(sqlCmd, null).toString()) )

{

returntrue;

}

returnfalse;

}

public Object getPageCount()

{

String sqlCmd="SELECT CEIL( COUNT(\*)/15.0) FROM Seat ";

return SQLUtil.*excuteScalar*(sqlCmd, null);

}

public Object getPageCountByWhere(String sqlWhere)

{

String sqlCmd="SELECT CEIL( COUNT(\*)/15.0) FROM Seat where "+sqlWhere;

return SQLUtil.*excuteScalar*(sqlCmd, null);

}

}

# 8 车辆信息管理模块的详细设计

车辆信息管理的Ｓｔｒｕｃｔ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－９：车辆信息管理管理的三层模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 视图 | | | 控制器 | 模型 |
| JSP组件 | ActionFormBean | 客户标签 |
| InputNum.jsp  parklist.jsp  parkdetail.jsp | InputNumForm  ExamDetailForm  AddExamForm  deleteForm | Struct Html标签  Struct Bean标签  Struct Logic标签  用户自定义CV标签库的标签 | InputNumAction  parklist Action  parkdetailAction  deleteAction | DBUtil  Carbean |

## 8.1视图层

车辆信息管理视图层包括JSP组件、Form组件以及标签库等。共有3个ＪＳＰ文件，共有4个Form文件。

## 8.2控制层

车辆信息管理控制层共有4个Action文件。

## 8.3模型层

车辆信息管理模型层共有3个Bean文件。

## 8.4分析及功能实现







# 9 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，他是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，也可以根据需要给这个配置文件换为其它的文件名。

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。

## 9.1 Web.xml配置文件

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>的子元素）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>

1. 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>

1. 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>

1. 配置标签库

Strcut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,那么必须在配置元素<taglib>中配置它们,它有两个子元素,<taglib-url>和<taglib-location>.例如:

<taglib>

<taglib-url>/WEB-INF/struts-bean.tld</taglib-url>

<taglib-location>/WEB-INF/struts-bean.tld </taglib-location>

</taglib>

## 13.2 strcut-config*.*Xml配置文件

Struts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. Struts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.

<struts-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，数据源负责建立和特定数据库的连接。<data-sources>元素包含零个、或者多个<data-source>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.BasicDataSource">

<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>

1. <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-bean>子元素，每个<form-bean>元素又包含多个属性。例如：

<form-beans>

<form-bean name="AddJobForm" type="org.apache.struts.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-bean>

<form-bean name="JobDetailForm" type="org.apache.struts.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="Requirements" type="java.lang.String" />

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

</form-bean>

</form-beans>

1. <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>

1. <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<forward>元素组成。<forward>元素用于将一个逻辑名映射到特定的URL，这样Action类或者JSP文件无需指定实际的URL，减弱了控制组件和视图组件之间的耦合。例如：

<global-forwards>

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

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

<forward name="tojobsearch" path="/position/jspsearchdetail.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/addjob1.jsp" />

<forward name="toreturn" path="/position/addjob1.jsp" />

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

<forward name="toaddjobexam" path="/position/addjob1.jsp" />

<forward name="todeljobexam" path="/position/addjob1.jsp" />

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

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

<forward name="toupdate" path="/position/addjob1.jsp" />

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

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

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

<forward name="tomodifyjobexampre" 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" />

</global-forwards>

1. <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.struts.action.JobDetailAction"></action>

<action attribute="JobSearchForm" input="/position/joblist.jsp" name="JobSearchForm" path="/position/jobsearch" scope="request" type="com.changjiangcompany.struts.action.JobSearchAction"></action>

</action-mappings>

1. <controller>元素

<controller>元素用于配置ActionServlet。例如

<controller processorClass="share.MyRequestProcessor" />

1. <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.struts.ApplicationResources" />

1. <plug-in>元素

<plug-in>元素用于配置Struts插件，如果没有插件可以不写