**目 录**

[1绪 论](#_Toc16953)

[1.1 研究的背景及意义](#_Toc2517)

[1.1.1 选题的背景](#_Toc6708)

[1.1.2 国内外研究现状](#_Toc13554)

[1.1.3 研究的意义](#_Toc15574)

[1.2 系统目标](#_Toc10686)

[2 需求分析](#_Toc28280)

[2.1 业务需求](#_Toc29283)

[2.1.1主要业务流程](#_Toc21256)

[2.2 功能需求](#_Toc987)

[2.2.1 角色分析](#_Toc29008)

[2.2.2 业务功能](#_Toc149)

[2.3 非功能需求](#_Toc1420)

[2.3.1环境需求](#_Toc28992)

[2.3.2 性能需求](#_Toc17805)

[2.3.3 安全需求](#_Toc11486)

[3 总体设计](#_Toc2800)

[3.1系统设计的原则](#_Toc12949)

[3.2 系统体系结构设计](#_Toc15427)

[3.3 系统功能结构设计](#_Toc2772)

[4 数据库设计](#_Toc21508)

[4.1 概念结构设计](#_Toc12991)

[4.1.1设计思路](#_Toc7387)

[4.1.2 E-R图](#_Toc27663)

[4.1.3主要实体属性图](#_Toc13452)

[4.2 逻辑结构设计](#_Toc23741)

[4.2.1 设计思路](#_Toc13156)

[4.2.2 逻辑模型](#_Toc1328)

[4.3 物理结构设计](#_Toc1079)

[4.3.1 存取方式](#_Toc2255)

[4.3.2 存储结构](#_Toc6442)

[5 界面设计](#_Toc25633)

[5.1 界面关系图或工作流图](#_Toc19413)

[5.2 界面设计成果](#_Toc17630)

[5.2.1 主界面](#_Toc27530)

[5.2.2 子界面](#_Toc7071)

[6 详细设计](#_Toc21188)

[6.1 系统主要功能模块介绍](#_Toc31768)

[6.2功能模块设计](#_Toc25860)

[6.2.1系统用户管理](#_Toc18422)

[6.2.2项目管理](#_Toc8363)

[6.2.3投资方管理](#_Toc16859)

[6.2.4签约项目管理](#_Toc3218)

[6.2.5产业链管理](#_Toc20748)

[6.2.6新闻信息管理](#_Toc18283)

[6.2.7友情链接管理](#_Toc17364)

[6.2.8个人资料管理](#_Toc28472)

[7 编码](#_Toc3489)

[7.1 代码实现与核心算法](#_Toc2387)

[7.2 代码优化分析](#_Toc10088)

[8 测试](#_Toc6109)

[8.1 测试方案设计](#_Toc11221)

[8.1.1 测试策略](#_Toc12180)

[8.1.2 测试进度安排](#_Toc22098)

[8.1.3 测试资源](#_Toc14681)

[8.1.4 关键测试点](#_Toc747)

[8.2 测试用例构建](#_Toc9460)

[8.2.1 测试用例编写约定](#_Toc32398)

[8.2.2 测试用例设计](#_Toc16079)

[8.2.3 关键测试用例](#_Toc12876)

[8.2.4 测试用例维护](#_Toc10438)

[9 总结与展望](#_Toc18806)

[9.1 设计工作总结](#_Toc13885)

[9.2 未来工作展望](#_Toc12751)

[谢 辞](#_Toc11801)

[参考文献](#_Toc3628)

[附录A 外文翻译—原文部分](#_Toc18769)

[附录B 外文翻译—译文部分](#_Toc14135)

[附录C 软件使用说明书](#_Toc15152)

[附录D 主要源代码](#_Toc823)

1绪 论

## 1.1 研究的背景及意义

### 1.1.1 选题的背景

随着信息化社会的形成和科学技术不断的的发展，人们对信息网络的需求越来越高，同时对信息网络的要求也越来越多。这是难题，但是对互联网行业来说也是一个机遇。信息技术不断更新与发展，为招商引资提供了良好的发展环境，招商引资平台的规模不断扩大，在扩大的同时带来了很多新的问题，所以我们需要一个使用而又方便的系统对招商引资来进行管理。在互联网高度发展和应用的今天，人们对招商引资的需求越来越多，也正因此，招商引资平台应运而生，可以为招商引资有效的解决很多实际问题。

传统的招商引资的方法中有值得我们借鉴的地方，但也有不适合当代社会的方法。其中政府组团招商是经过政府相关人员的策划和筹备，政府领导集体出行，以招商会的形式到一些经济比较发达的城市进行招商，这种招商的缺点是开支大而且成功率低；文化招商是利用自己当地的文化特色，进行宣传，这种方法优点是具有自己的文化特色，可以引起一些特定的投资方的兴趣；以商招商是通过当时已经存在的成功企业，利用商人之间的交往，通过他们的介绍来获取投资，这种方法的优点是费用低而且容易成功，但是这种方法需要很长时间的积累，因为口碑是很难短时间就有的；旅游招商是利用当地的旅游资源在吸引游客观光的同时，吸引一些对此有兴趣的投资商，利用旅游实现推动旅游行业和吸引投资方的双重目的，这种方法的优点是不用主动出击，但是可以集中客户，缺点是所需费用较高而且所花费的时间比较长。在这些传统的招商方式中，我们既要看到它们的优点，也要看到他们的不足，从而使我们的平台更加具有先进性，优越性以及完备性。

### 1.1.2 国内外研究现状

随着互联网技术的不断发展，国内外出现了众多的招商引资平台。近几年来，招商引资平台技术没有什么重大的改变，同时传统的招商引资无法满足人们的现状也不会改变，在这个基础上，如何把招商引资平台设计的更加全面、更加优越、更加符合人们的需求就尤为重要了。招商引资平台的投入资金相对较低，中国人口人多，招商引资平台的需求巨大，市场容量与盈利性巨大的招商引资平台用户规模将不断呈增长的趋势。

在互联网蓬勃发展的今天，网络和我们的生活、学习、工作、娱乐越来越密不可分，一个网络新时代已经来临。利用开放的互联网技术开发出实时、先进、快捷、全方位的招商信息传播平台，使得信息在人与人之间的传播更为轻松，系统为用户提供服务，用户为平台提供项目和知名度，双方都能及时并且迅速快捷的得到信息。所以招商平台的设计必须可以解决大量信息的查询和管理，便于系统更好的管理和更改信息，同时也为用户提供了方便。

中国招商引资信息网是网群国际商协联盟有限公司旗下的集群式门户网站之一。中国招商引资信息网是以各国渠道为支撑的全球一体化招商引资、投资融资、产品营销的行业信息领军平台。中国招商引资信息网专注于品牌推广、人才引进、产品营销、技术转让、产权并购、产业转移、项目包装客商邀约、招投考察、会展支持等业务全球推广与项目对接的国际化平台。

### 1.1.3 研究的意义

本文设计并实现了招商引资平台系统。这个系统可以管理所有用户上传的项目，帮助用户让他们的项目可以被更多投资商看到，解决他们难以找到投资的难题，为用户带来便利。在这里面有很多投资方的信息，里面有他们的介绍以及他们的投资意向，既可以让他们更快的找到相关的投资项目，同时也可以让项目发布者更有针对性的寻找投资商。本系统分为前台和后台，前台有项目列表可以展示项目信息，投资方列表展示投资方信息，还有新闻信息可以让用户更快了解相关行内信息，有产业链信息展示。后台分为用户后台和管理员后台，用户后台可以发布个人的项目并对其进行管理，管理员后台可以对所有用户信息、项目信息、投资方信息、产业链信息、、友情链接信息、新闻信息进行管理，同时还能看到数据统计的信息等。通过本平台，可以更好的对招商项目进行管理，提高招商引资的效率。

## 1.2 系统目标

架构设计和开发技术：在本系统设计中，主要采用的体系结构是B/S结构。所谓B/S结构，也就是浏览器/服务器结构，在这种架构下，用户工作界面是通过浏览器来实现，部分[事务](https://baike.baidu.com/item/%E4%BA%8B%E5%8A%A1)逻辑在前端实现，主要事务逻辑在服务器端实现，形成所谓三层结构。为提高提高程序的规范性和代码的重用性本系统使用SSM开发框架，即 SpringMVC、Spring 与 MyBatis 三个框架。

它们在三层架构中所处的位置是不同的，同样它们在三层架构中的功能各不相同。SpringMVC：作为 View 层的实现者，完成用户的请求接收功能。SpringMVC 的Controller作为整个应用的控制器，完成用户请求的转发及对用户的响应。 MyBatis：作为 Dao 层的实现者，完成对数据库的增、删、改、查功能。Spring：整个应用中所有对象的创建、初始化、销毁，及对象间关联关系的维护，均由 Spring 进行管理。

系统架构图如图1-1所示。

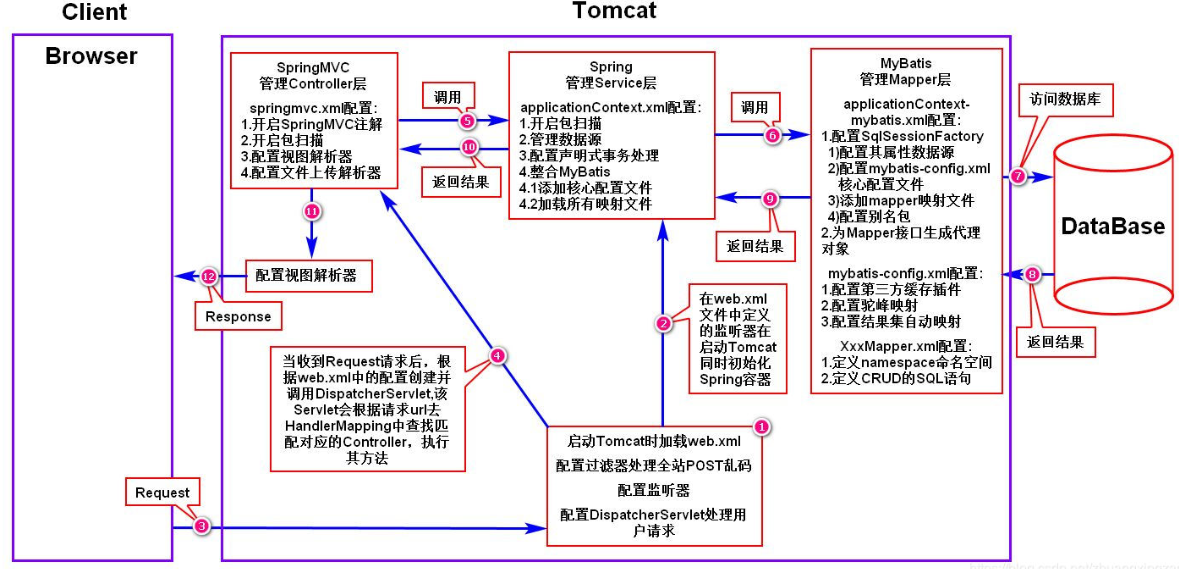


图1-1 系统架构图

在招商引资平台系统中，将用户分为普通用户和管理员两个不同的身份，而每个不同的身份在系统中扮演不同的角色，对应每个模块的操作也不同。

（1）管理员能对用户信息、项目信息、投资方信息、新闻信息、产业链信息、签约项目信息、友情链接信息进行增删改操作，同时可以查看数据统计的信息。

（2）普通用户可以发布项目，同时对自己的信息和项目进行修改，删除等操作，还可以查看项目信息、投资方信息、产业链信息、新闻信息、友情链接。

# 2 需求分析

## 2.1 业务需求

招商引资平台系统中主要包括用户信息管理模块、项目管理模块、投资方管理模块、产业链管理模块、签约项目管理模块、新闻信息管理模块、友情链接管理模块、数据统计模块八个主要模块。

1. 用户信息管理模块，所有人员可以在登录系统后查看个人信息、修改个人信息、修改密码。
2. 项目管理模块，普通用户可以发布自己的项目，也可以对自己的项目进行修改，删除等操作，管理员可以在登录系统后查看项目信息、删除项目信息、修改项目信息。
3. 投资方管理模块，管理员可以在登录系统后查看投资方信息、修改投资方信息、删除投资方信息、新增投资方信息。普通用户可以查看投资方信息。
4. 产业链管理模块，管理员可以在登录系统后查看产业链信息、修改产业链信息、删除产业链信息、新增产业链信息。普通用户可以查看产业链信息。
5. 签约项目管理模块，普通用户可以查看自己的签约项目、修改签约项目、删除签约项目。管理员可以在登录系统后查看签约项目、修改签约项目、删除签约项目。
6. 新闻信息管理模块，管理员可以在登录系统后查看新闻信息、修改新闻信息、删除新闻信息、新增新闻信息。用户可以查看新闻信息。
7. 友情链接管理模块，管理员可以在登录系统后查看友情链接信息、修改友情链接信息、删除友情链接信息、新增友情链接信息。普通用户可以查看并点击友情链接。
8. 数据统计模块，管理员可以在登录系统后查看数据统计的信息。

### 2.1.1主要业务流程

（1）个人资料管理活动图如图2-1所示。所有人员可以在登录系统之后对个人信息进行查看、修改操作。

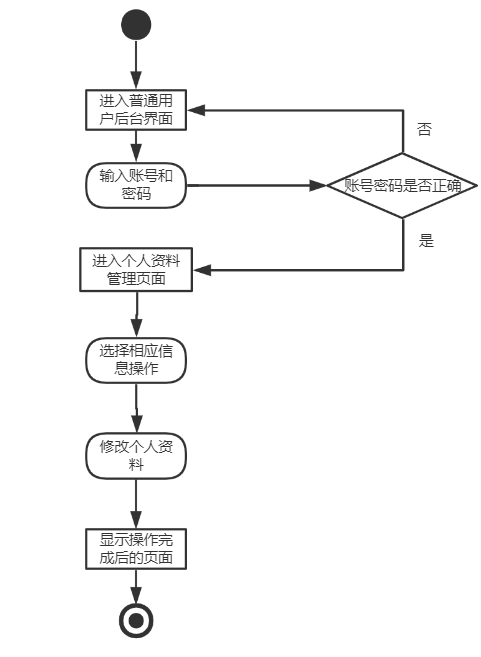


图2-1 个人资料管理活动图

（2）项目管理活动图如图2-2所示。普通用户可以新增自己的项目，也可以对自己的项目进行修改，删除等操作，管理员可以在登录系统后查看项目信息、删除项目信息、修改项目信息。

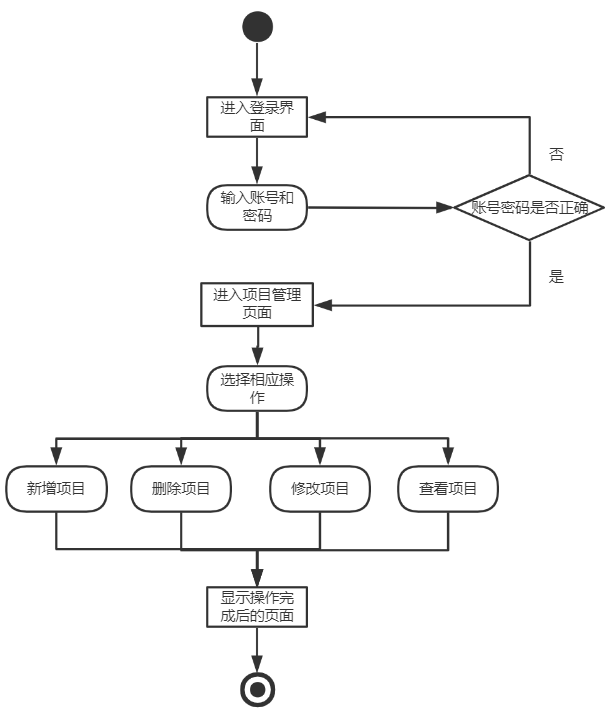


图2-2 项目管理活动图

（3）投资方管理活动图如图2-3所示。管理员可以在登录系统后查看投资方信息、修改投资方信息、删除投资方信息、新增投资方信息。

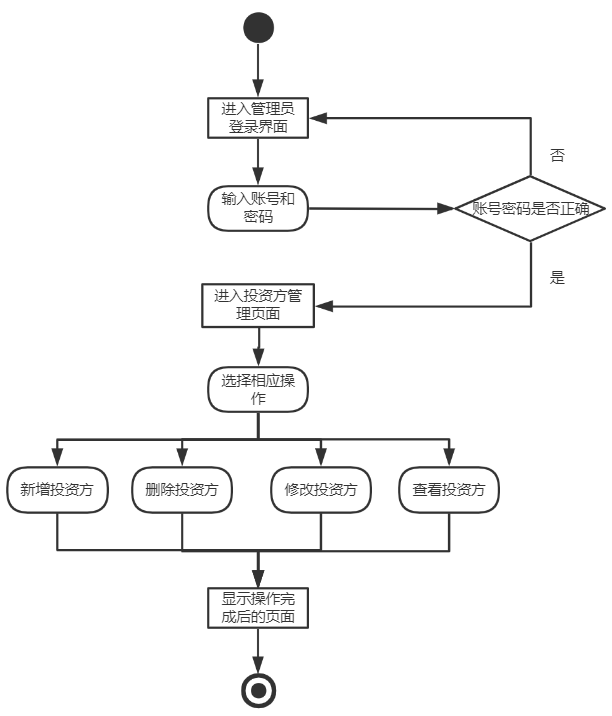


图2-3 投资方管理活动图

（4）产业链管理活动图如图2-4所示。管理员可以在登录系统后查看产业链信息、修改产业链信息、删除产业链信息、新增产业链信息。

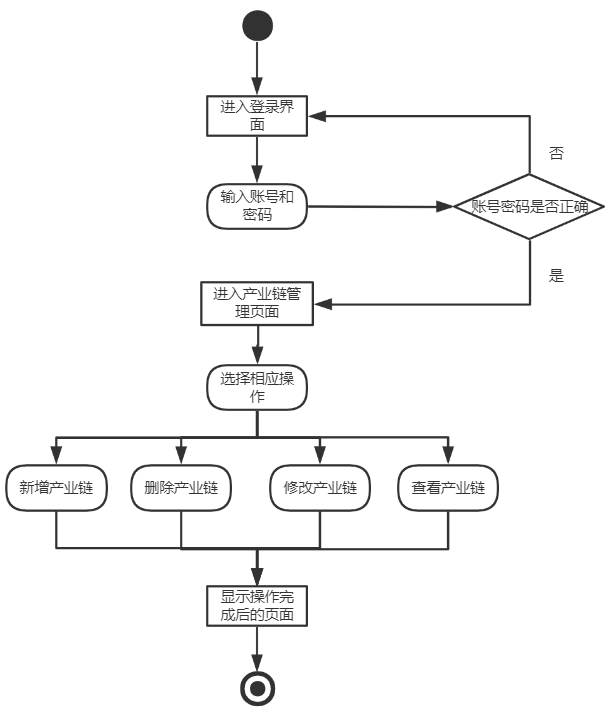


图2-4 产业链管理活动图

（5）签约项目管理活动图如图2-5所示。普通用户可以查看自己的签约项目、修改签约项目、删除签约项目。管理员可以在登录系统后查看签约项目、修改签约项目、删除签约项目。

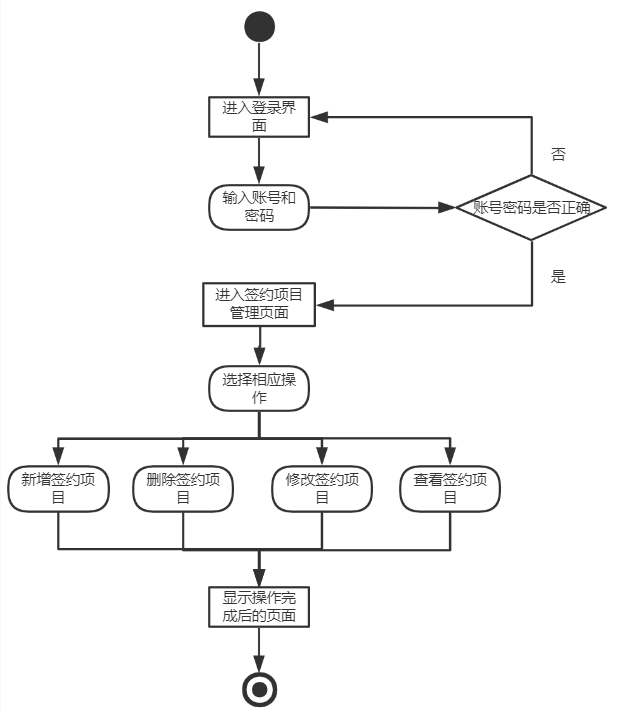


图2-5 签约项目管理活动图

（6）新闻信息管理活动图如图2-6所示。管理员可以在登录系统后查看新闻信息、修改新闻信息、删除新闻信息、新增新闻信息。

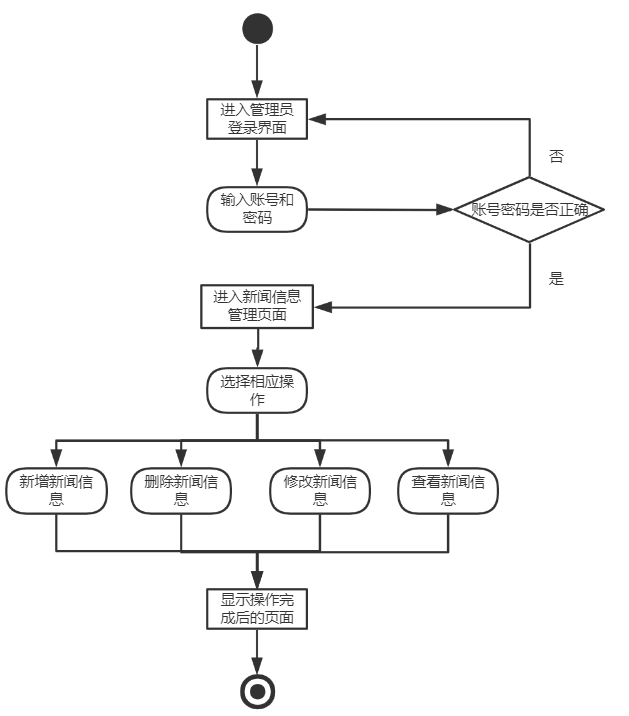


图2-6 新闻信息管理活动图

（7）数据统计管理活动图如图2-7所示。管理员可以在登录系统后查看数据统计的信息。

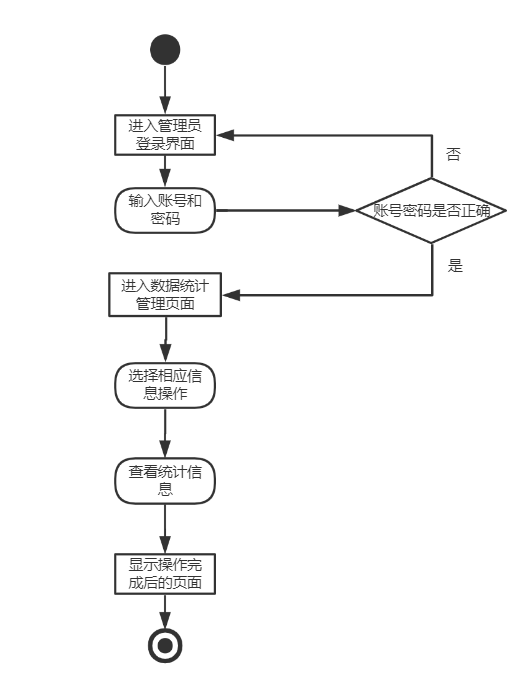


图2-7 数据统计管理活动图

## 2.2 功能需求

### 2.2.1 角色分析

以下从招商引资平台系统的实际需求分析，系统涉及到以下角色如表2-1所示。

表2-1 系统角色表

|  |  |
| --- | --- |
| 角色 | 职责或功能 |
| 普通用户 | 主要是浏览页面，查看相关信息，可以发布项目，对个人信息和项目进行相关操作。 |
| 管理员 | 主要是对用户信息、项目信息、投资方信息、产业链信息、签约项目信息、新闻信息进行相关操作，还可以查看数据统计结果。 |

### 2.2.2 业务功能

以下从实际情况角度出发，给出了系统的总体用例图，招商引资平台中普通用户的主要功能包括对个人资料管理、项目管理、项目签约管理。管理员的主要功能包括对系统用户管理、新闻数据管理、友情链接管理、项目管理、项目签约管理、投资方管理、产业链管理、统计管理等，系统总体用例图如图2-8所示。

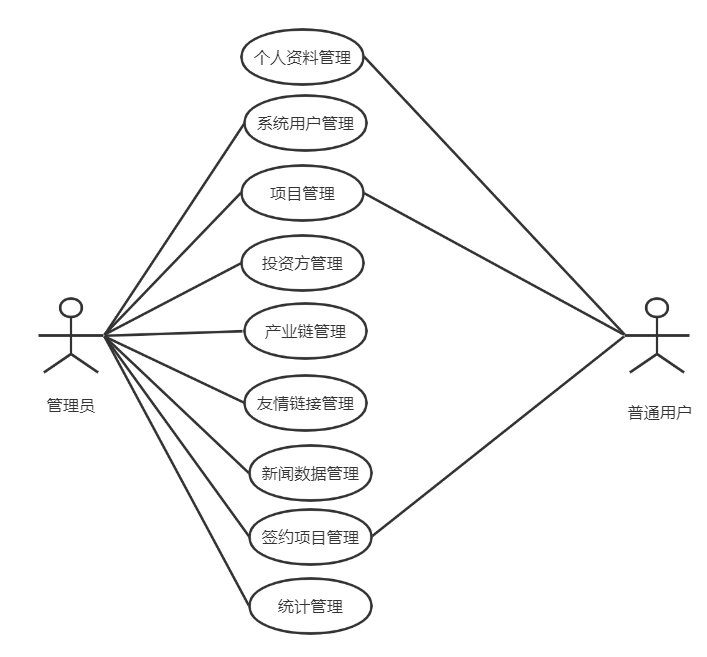


图2-8 系统总体用例图

（1）系统用户管理

系统用户信息管理包括新增用户、删除用户、查看用户、修改用户。其中修改用户可以对用户的密码、姓名、性别、地址、电话等信息进行修改。用户信息管理用例图如图2-9所示。



图2-9 系统用户管理用例图

在系统用户管理页面，管理员可以对用户进行相应操作，新增用户用例如表2-2所示。

表2-2 新增用户用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0001 | 用例名称 | 新增用户 |
| 用例描述 | 该用例是增加用户用例。管理员可以进行浏览用户信息，并且对用户进行增加操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击新增用户 | | |
| 后置条件 | 如果这个用例成功，管理员即可新增用户 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击用户管理；  2.点击新增用户；  3添加用户信息；  4点击添加按钮；  5.用户添加成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在系统用户管理页面，管理员可以对用户进行相应操作，删除用户用例如表2-3所示。

表2-3 删除用户用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0002 | 用例名称 | 删除用户 |

续表2-3

|  |  |
| --- | --- |
| 用例描述 | 该用例是删除用户用例。管理员可以进行浏览用户信息，并且对用户进行删除操作. |
| 执行者 | 管理员 |
| 前置条件 | 管理员点击删除用户 |
| 后置条件 | 如果这个用例成功，管理员即可删除用户 |
| 涉众利益 | 无 |
| 用例场景 | 后台管理界面 |
| 基本事件流 | 基本流程（Base Flow）  1.点击用户管理；  2.点击删除用户；  3.用户删除成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在系统用户管理页面，管理员可以对用户进行相应操作，查找用户用例如表2-4所示。

表2-4 查找用户用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0003 | 用例名称 | 查找用户 |
| 用例描述 | 该用例是查找用户用例。管理员可以进行浏览用户信息，并且对用户进行查找操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击查找 | | |
| 后置条件 | 如果这个用例成功，管理员即可查找用户 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击用户管理；  2.输入所要查找用户的信息；  3.点击查找按钮；  4.显示查找后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在系统用户管理页面，管理员可以对用户进行相应操作，修改用户用例如表2-5所示。

表2-5 修改用户用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0004 | 用例名称 | 修改用户 |
| 用例描述 | 该用例是修改用户用例。管理员可以进行浏览用户信息，并且对用户进行修改操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击编辑 | | |
| 后置条件 | 如果这个用例成功，管理员即可修改用户 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击用户管理；  2.选择所要修改的用户的信息，点击编辑；  3.输入所要修改的信息，点击提交；  4.修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

（2）项目管理

项目管理功能包括普通用户对个人项目进行新增、修改、删除、查看，以及管理员对所有项目可以进行修改、删除、查找操作。项目管理用例图如图2-10所示。

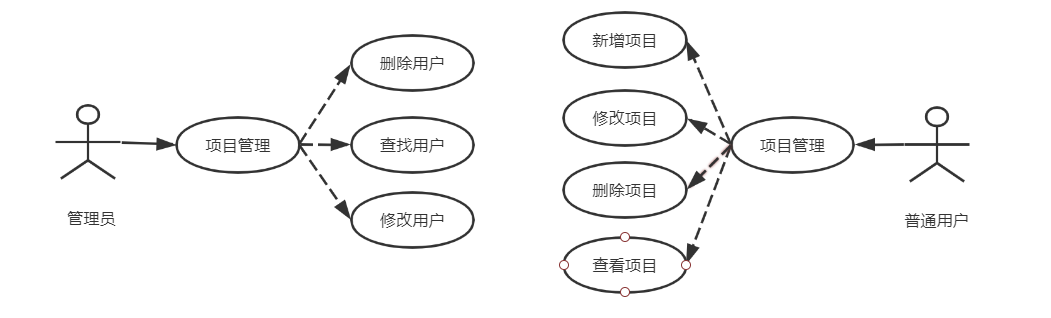


图2-10 项目管理用例图

在项目管理页面，管理员可以对项目进行删除操作，删除项目用例如表2-6所示。

表2-6 删除项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0005 | 用例名称 | 删除项目 |
| 用例描述 | 该用例是删除项目用例。管理员可以进行浏览项目信息，并且对项目进行删除操作. | | |

续表2-6

|  |  |
| --- | --- |
| 执行者 | 管理员 |
| 前置条件 | 管理员点击删除项目 |
| 后置条件 | 如果这个用例成功，管理员即可删除项目 |
| 涉众利益 | 无 |
| 用例场景 | 后台管理界面 |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.点击删除项目；  3.项目删除成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在项目管理页面，管理员可以对项目进行修改操作，修改项目用例如表2-7所示。

表2-7 修改项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0006 | 用例名称 | 修改项目 |
| 用例描述 | 该用例是修改项目用例。管理员可以进行浏览项目信息，并且对项目进行修改操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击修改项目 | | |
| 后置条件 | 如果这个用例成功，管理员即可修改项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.点击修改项目；  3.输入修改后的项目信息；  4.点击提交按钮；  5.项目修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
|  |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在项目管理页面，管理员可以对项目进行查找操作，查找项目用例如表2-8所示。

表2-8 查找项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0007 | 用例名称 | 查找项目 |
| 用例描述 | 该用例是查找项目用例。管理员可以进行浏览项目信息，并且对项目进行查找操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击查找项目 | | |
| 后置条件 | 如果这个用例成功，管理员即可查找项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.输入所要查找的项目信息，点击查询；  3.显示查找后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在项目管理页面，普通用户可以对项目进行新增操作，新增项目用例如表2-9所示。

表2-9 发布项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0008 | 用例名称 | 新增项目 |
| 用例描述 | 该用例是发布项目用例。普通用户对个人项目进行新增操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击项目添加 | | |
| 后置条件 | 如果这个用例成功，普通用户即可发布项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.点击添加项目；  3.填写项目信息；  4.点击提交按钮；  5.项目发布成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |

续表2-9

|  |  |
| --- | --- |
| 编写人 |  |

在项目管理页面，普通用户可以对项目进行删除操作，删除项目用例如表2-10所示。表2-10 删除项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0009 | 用例名称 | 删除项目 |
| 用例描述 | 该用例是删除项目用例。普通用户对个人项目进行删除操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击删除项目 | | |
| 后置条件 | 如果这个用例成功，普通用户即可删除项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.点击删除项目；  3.项目删除成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在项目管理页面，普通用户可以对项目进行修改操作，修改项目用例如表2-11所示。

表2-11 修改项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0010 | 用例名称 | 修改项目 |
| 用例描述 | 该用例是修改项目用例。普通用户对个人项目进行修改操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击修改项目 | | |
| 后置条件 | 如果这个用例成功，普通用户即可修改项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.点击编辑项目；  3.输入修改后的项目信息；  4.项目修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role） | | |

续表2-11

|  |  |
| --- | --- |
|  | 无 |
| 特殊需求 | 无 |
| 编写人 |  |

在项目管理页面，普通用户可以对项目进行查看操作，查看项目用例如表2-12所示。

表2-12 查看项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0011 | 用例名称 | 查看项目 |
| 用例描述 | 该用例是查看项目用例。普通用户对个人项目进行查看操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击项目 | | |
| 后置条件 | 如果这个用例成功，普通用户即可查看项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.选择所要查看的项目信息，点击该项目；  3.显示点击后后的项目信息；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

（3）投资方管理

投资方管理功能包括普通用户对投资方信息进行查看，以及管理员对所有投资方可以进行新增、删除、修改、查找操作。投资方管理用例图如图2-11所示。

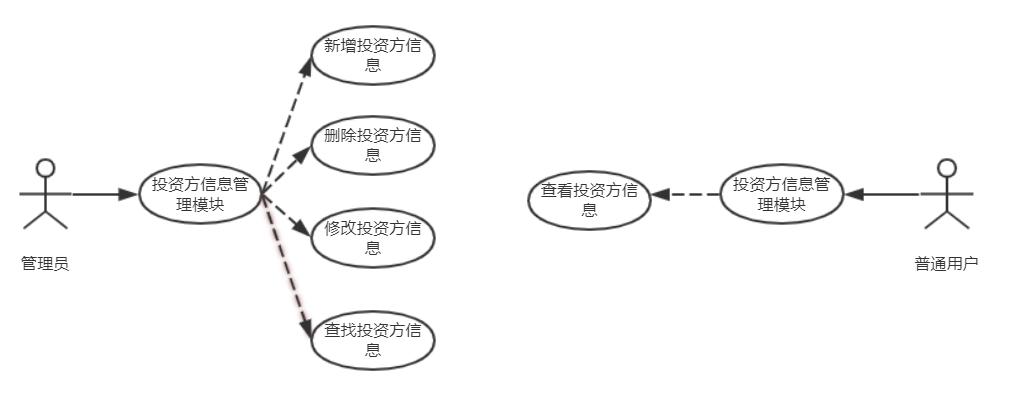


图2-11 投资方管理用例图

在投资方管理页面，管理员可以对投资方进行新增操作，新增投资方用例如表2-13所示。

表2-13 新增投资方用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0012 | 用例名称 | 新增投资方 |
| 用例描述 | 该用例是增加投资方用例。管理员可以进行浏览投资方信息，并且对投资方进行增加操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击新增投资方 | | |
| 后置条件 | 如果这个用例成功，管理员即可新增投资方 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击投资方管理；  2.点击新增投资方；  3.添加投资方信息；  4.点击提交按钮；  5.投资方添加成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在投资方管理页面，管理员可以对投资方进行删除操作，删除投资方用例如表2-14所示。

表2-14 删除投资方用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0013 | 用例名称 | 删除投资方 |
| 用例描述 | 该用例是删除投资方用例。管理员可以进行浏览投资方信息，并且对投资方进行删除操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击删除投资方 | | |
| 后置条件 | 如果这个用例成功，管理员即可删除投资方 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击投资方管理；  2.点击删除投资方；  3.删除投资方成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无 | | |

续表2-14

|  |  |
| --- | --- |
|  | 业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在投资方管理页面，管理员可以对投资方进行修改操作，修改投资方用例如表2-15所示。

表2-15 修改投资方用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0014 | 用例名称 | 修改投资方 |
| 用例描述 | 该用例是修改投资方用例。管理员可以进行浏览投资方信息，并且对投资方进行修改操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击修改投资方 | | |
| 后置条件 | 如果这个用例成功，管理员即可修改投资方 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击投资方管理；  2.点击修改投资方；  3.输入修改后的投资方信息；  4.点击提交按钮；  5.投资方修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
|  |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在投资方管理页面，管理员可以对投资方进行查找操作，查找投资方用例如表2-16所示。

表2-16 查找投资方用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0015 | 用例名称 | 查找投资方 |
| 用例描述 | 该用例是查找投资方用例。管理员可以进行浏览投资方信息，并且对投资方进行查找操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击查找投资方 | | |
| 后置条件 | 如果这个用例成功，管理员即可查找投资方 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击投资方管理； | | |

续表2-16

|  |  |
| --- | --- |
|  | 2.输入需要查找的投资方信息，点击查询按钮；  3.显示查找后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在投资方前台页面，普通用户可以对投资方进行查看操作，查看投资方用例如表2-17所示。

表2-17 查看投资方信息用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0016 | 用例名称 | 查看投资方信息 |
| 用例描述 | 该用例是查看投资方信息用例。普通用户可以进行查看投资方信息。 | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户查看投资方 | | |
| 后置条件 | 如果这个用例成功，管理员即可查看投资方信息 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 前台界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击投资方；  2.点击所要查看的投资方；  3.显示点击后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
|  |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

（4）产业链管理

产业链管理功能包括普通用户对产业链的查看查找，以及管理员对产业链的新增、删除、修改、查找操作。产业链管理用例图如图2-12所示。

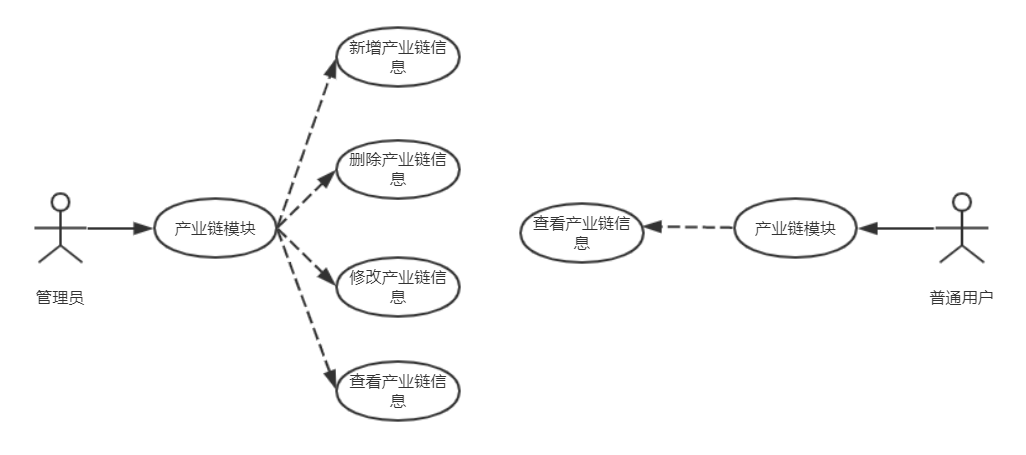


图2-12 产业链管理用例图

在产业链管理页面，管理员可以对产业链进行新增操作，新增产业链操作用例如表2-18所示。

表2-18 新增产业链用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0017 | 用例名称 | 新增产业链 |
| 用例描述 | 该用例是增加产业链用例。管理员可以进行浏览产业链信息，并且对产业链进行增加操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击新增产业链 | | |
| 后置条件 | 如果这个用例成功，管理员即可新增产业链 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击产业链管理；  2.点击添加产业链；  3.添加产业链信息；  4.点击提交按钮；  5.产业链添加成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
|  |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在产业链管理页面，管理员可以对产业链进行删除操作，删除产业链操作用例如表2-19所示。

表2-19 删除产业链用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0018 | 用例名称 | 删除产业链 |

续表2-19

|  |  |
| --- | --- |
| 用例描述 | 该用例是删除产业链用例。管理员可以进行浏览产业链信息，并且对产业链进行删除操作. |
| 执行者 | 管理员 |
| 前置条件 | 管理员点击删除产业链 |
| 后置条件 | 如果这个用例成功，管理员即可删除产业链 |
| 涉众利益 | 无 |
| 用例场景 | 后台管理界面 |
| 基本事件流 | 基本流程（Base Flow）  1.点击产业链管理；  2.点击删除产业链；  3.删除产业链成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在产业链管理页面，管理员可以对产业链进行修改操作，修改产业链操作用例如表2-20所示。

表2-20 修改产业链用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0019 | 用例名称 | 修改产业链 |
| 用例描述 | 该用例是修改产业链用例。管理员可以进行浏览产业链信息，并且对产业链进行修改操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击修改产业链 | | |
| 后置条件 | 如果这个用例成功，管理员即可修改产业链 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击产业链管理；  2.点击编辑产业链；  3.输入修改后的产业链信息；  4.点击提交按钮；  5.产业链修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |

续表2-20

|  |  |
| --- | --- |
| 特殊需求 | 无 |
| 编写人 |  |

在产业链管理页面，管理员可以对产业链进行查找操作，查找产业链操作用例如表2-21所示。

表2-21 查找产业链用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0020 | 用例名称 | 查找产业链 |
| 用例描述 | 该用例是查找产业链用例。管理员可以进行浏览产业链信息，并且对产业链进行查找操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击查找产业链 | | |
| 后置条件 | 如果这个用例成功，管理员即可查找产业链 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击产业链管理；  2.输入需要查找的产业链信息，点击查询；  3.显示查找后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在产业链前台页面，普通用户可以对产业链进行查看操作，查看产业链操作用例如表2-22所示。

表2-22 查看产业链信息用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0021 | 用例名称 | 查看产业链信息 |
| 用例描述 | 该用例是查看产业链用例。普通用户可以进行查看产业链信息。 | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户查看产业链 | | |
| 后置条件 | 如果这个用例成功，管理员即可查看产业链信息 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 前台界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击产业链；  2.点击所要查看的产业链；  3.显示点击后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List） | | |

续表2-22

|  |  |
| --- | --- |
|  | 无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

（5）新闻信息管理

新闻信息管理功能包括普通用户对新闻的查看操作，以及管理员对新闻信息的新增、删除、修改、查找操作。新闻信息管理用例图如图2-13所示。

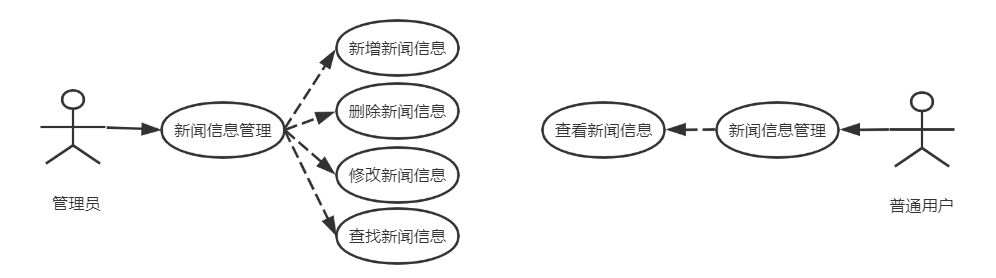


图2-13 新闻信息管理用例图

在新闻信息管理页面，管理员可对新闻进行新增操作，新增新闻用例如表2-23所示。

表2-23 新增新闻数据用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0022 | 用例名称 | 新增新闻 |
| 用例描述 | 该用例是增加新闻用例。管理员可以进行浏览新闻数据，并且对新闻进行增加操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击添加新闻 | | |
| 后置条件 | 如果这个用例成功，管理员即可新增新闻 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击新闻信息管理；  2.点击新增信息数据；  3.添加新闻信息信息；  4.点击添加按钮；  5.新闻数据添加成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在新闻信息管理页面，管理员可对新闻进行删除操作，删除新闻用例如表2-24所示。

表2-24 删除新闻数据用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0023 | 用例名称 | 删除新闻 |
| 用例描述 | 该用例是删除新闻用例。管理员可以进行浏览新闻信息，并且对新闻进行删除操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击删除新闻 | | |
| 后置条件 | 如果这个用例成功，管理员即可删除新闻 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击新闻信息管理；  2.点击删除新闻信息；  3.删除新闻信息成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
|  |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在新闻信息管理页面，管理员可对新闻进行新增操作，新增新闻用例如表2-25所示。

表2-25 修改新闻数据用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0024 | 用例名称 | 修改新闻 |
| 用例描述 | 该用例是修改新闻用例。管理员可以进行浏览新闻信息，并且对新闻进行修改操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击修改新闻 | | |
| 后置条件 | 如果这个用例成功，管理员即可修改新闻 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击新闻信息管理；  2.点击修改新闻信息；  3.输入修改后的新闻信息；  4.点击提交按钮；  5.新闻信息修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role） | | |

续表2-25

|  |  |
| --- | --- |
|  | 无 |
| 特殊需求 | 无 |
| 编写人 |  |

在新闻信息管理页面，管理员可对新闻进行查找操作，查找新闻用例如表2-26所示。

表2-26 查找新闻数据用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0025 | 用例名称 | 查找新闻 |
| 用例描述 | 该用例是查找新闻用例。管理员可以进行浏览新闻信息，并且对新闻进行查找操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击查找新闻 | | |
| 后置条件 | 如果这个用例成功，管理员即可查找新闻 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击新闻信息管理；  2.输入需要查找的新闻信息；  3.显示查找后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
|  |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在新闻信息前台页面，普通用户可对新闻进行查看操作，查看新闻用例如表2-27所示。

表2-27 查看新闻数据信息用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0026 | 用例名称 | 查看新闻 |
| 用例描述 | 该用例是查看新闻信息用例。普通用户可以进行查看新闻信息。 | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户查看新闻 | | |
| 后置条件 | 如果这个用例成功，管理员即可查看新闻信息 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 前台界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击新闻信息；  2.点击所要查看的新闻信息；  3.显示点击后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无 | | |

续表2-27

|  |  |
| --- | --- |
|  | 业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

（6）友情链接管理

友情链接管理功能包括普通用户对友情链接的查看操作，以及管理员对友情链接的新增、删除、修改、查找操作。友情链接管理用例图如图2-14所示。

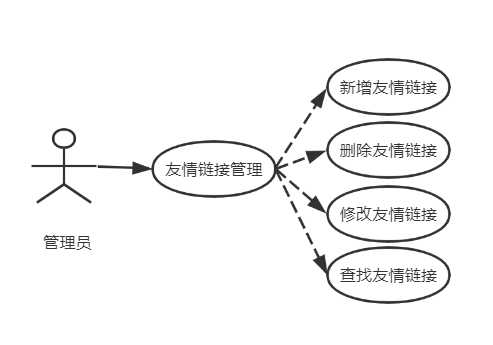


图2-14 友情链接管理用例图

在友情链接管理页面，管理员可以对友情链接进行新增操作，新增友情链接用例如表2-28所示。

表2-28 新增友情链接用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0027 | 用例名称 | 新增友情链接 |
| 用例描述 | 该用例是增加友情链接用例。管理员可以进行浏览友情链接信息，并且对友情链接进行增加操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击新增友情链接 | | |
| 后置条件 | 如果这个用例成功，管理员即可新增友情链接 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击友情链接管理；  2.点击新增友情链接；  3.添加友情链接信息；  4.点击提交按钮；  5.友情链接添加成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |

续表2-28

|  |  |
| --- | --- |
| 编写人 |  |

在友情链接管理页面，管理员可以对友情链接进行删除操作，删除友情链接用例如表2-29所示。

表2-29删除友情链接用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0028 | 用例名称 | 删除友情链接 |
| 用例描述 | 该用例是删除友情链接用例。管理员可以进行浏览友情链接信息，并且对友情链接进行删除操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击删除友情链接 | | |
| 后置条件 | 如果这个用例成功，管理员即可删除友情链接 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击友情链接管理；  2.点击删除友情链接；  3.删除友情链接成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在友情链接管理页面，管理员可以对友情链接进行修改操作，修改友情链接用例如表2-30所示。

表2-30 修改友情链接用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0029 | 用例名称 | 修改友情链接 |
| 用例描述 | 该用例是修改友情链接用例。管理员可以进行浏览友情链接信息，并且对友情链接进行修改操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击修改友情链接 | | |
| 后置条件 | 如果这个用例成功，管理员即可修改友情链接 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击友情链接管理；  2.点击修改友情链接；  3.输入修改后的友情链接信息；  4.点击提交按钮；  5.友情链接修改成功；  扩展流程（Extend Flow） | | |

续表2-30

|  |  |
| --- | --- |
|  | 无  字段列表（Filed List）  无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在友情链接管理页面，管理员可以对友情链接进行查找操作，查找友情链接用例如表2-31所示。

表2-31 查找友情链接用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0030 | 用例名称 | 查找友情链接 |
| 用例描述 | 该用例是查找友情链接用例。管理员可以进行浏览友情链接信息，并且对友情链接进行查找操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击查找友情链接 | | |
| 后置条件 | 如果这个用例成功，管理员即可查找友情链接 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击友情链接管理；  2 输入需要查找的友情链接信息，点击查询；  3.显示查找后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

（7）签约项目管理

签约项目管理功能包括普通用户对个人签约项目进行新增、修改、删除、查看，以及管理员对所有签约项目可以进行修改、删除、查找操作。项目管理用例图如图2-15所示。



图2-15 签约项目管理用例图

在签约项目管理页面，管理员可以对项目进行删除操作，删除项目用例如表2-32所示。

表2-32 删除签约项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0031 | 用例名称 | 删除签约项目 |
| 用例描述 | 该用例是删除签约项目用例。管理员可以进行浏览签约项目信息，并且对签约项目进行删除操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击删除签约项目 | | |
| 后置条件 | 如果这个用例成功，管理员即可删除签约项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击签约项目管理；  2.点击删除签约项目；  3.签约项目删除成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在签约项目管理页面，管理员可以对签约项目进行修改操作，修改签约项目用例如表2-33所示。

表2-33 修改项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0032 | 用例名称 | 修改签约项目 |
| 用例描述 | 该用例是修改签约项目用例。管理员可以进行浏览签约项目信息，并且对签约项目进行修改操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击修改签约项目 | | |
| 后置条件 | 如果这个用例成功，管理员即可修改签约项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击签约项目管理；  2.点击修改签约项目；  3.输入修改后的签约项目信息；  4.点击提交按钮；  5.签约项目修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List） | | |

续表2-33

|  |  |
| --- | --- |
|  | 无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在签约项目管理页面，管理员可以对签约项目进行查找操作，查找签约项目用例如表2-34所示。

表2-34 查找项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0033 | 用例名称 | 查找签约项目 |
| 用例描述 | 该用例是查找签约项目用例。管理员可以进行浏览签约项目信息，并且对签约项目进行查找操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击查找签约项目 | | |
| 后置条件 | 如果这个用例成功，管理员即可查找签约项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击签约项目管理；  2.输入所要查找的签约项目信息，点击查询；  3.显示查找后的页面；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在签约项目管理页面，普通用户可以对签约项目进行新增操作，新增签约项目用例如表2-35所示。

表2-35 新增签约项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0034 | 用例名称 | 新增签约项目 |
| 用例描述 | 该用例是新增签约项目用例。用户对签约项目进行新增操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击项目签约 | | |
| 后置条件 | 如果这个用例成功，普通用户即可签约项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击项目管理；  2.点击项目签约； | | |

续表2-35

|  |  |
| --- | --- |
|  | 5.项目签约成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在签约项目管理页面，普通用户可以对签约项目进行删除操作，删除签约项目用例如表2-36所示。

表2-36 删除签约项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0035 | 用例名称 | 删除签约项目 |
| 用例描述 | 该用例是删除签约项目用例。用户对签约项目进行删除操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击删除签约项目 | | |
| 后置条件 | 如果这个用例成功，普通用户即可删除签约项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击签约项目管理；  2.点击删除签约项目；  3.签约项目删除成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

在签约项目管理页面，普通用户可以对签约项目进行修改操作，修改签约项目用例如表2-37所示。

表2-37 修改签约项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0036 | 用例名称 | 修改签约项目 |
| 用例描述 | 该用例是修改签约项目用例。用户对签约项目进行修改操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击修改签约项目 | | |
| 后置条件 | 如果这个用例成功，普通用户即可修改签约项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |

续表2-37

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）  1.点击签约项目管理；  2.点击编辑签约项目；  3.输入修改后的签约项目信息；  4.签约项目修改成功；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 |
| 特殊需求 | 无 |
| 编写人 |  |

在签约项目管理页面，普通用户可以对签约项目进行查看操作，查看签约项目用例如表2-38所示。

表2-38 查看项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0037 | 用例名称 | 查看签约项目 |
| 用例描述 | 该用例是查看签约项目用例。普通用户对个人签约项目进行查看操作. | | |
| 执行者 | 普通用户 | | |
| 前置条件 | 普通用户点击签约项目 | | |
| 后置条件 | 如果这个用例成功，普通用户即可查看签约项目 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击签约项目管理；  2.选择所要查看的签约项目信息，点击该签约项目；  3.显示点击后后的签约项目信息；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
|  |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

（8）数据统计管理

数据统计管理功能包括管理员对统计结果进行查看操作。数据统计管理用例图如图2-16所示。

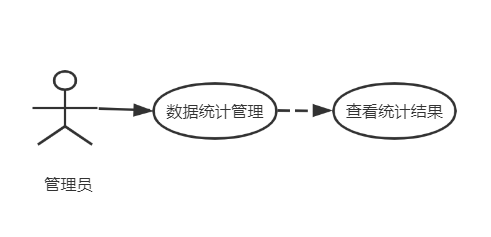


图2-16 数据统计管理用例图

在数据统计管理页面，管理员对统计结果进行查看，查看数据用例如表2-39所示。

表2-39 查看项目用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0038 | 用例名称 | 查看统计 |
| 用例描述 | 该用例是查看统计用例。管理员对统计结果进行查看操作. | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员点击统计信息 | | |
| 后置条件 | 如果这个用例成功，管理员即可查看统计结果 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 后台管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）  1.点击数据统计管理；  2.输入所要查看的数据统计信息，点击；  3.显示点击后的统计结果；  扩展流程（Extend Flow）  无  字段列表（Filed List）  无  业务规则（Business role）  无 | | |
| 特殊需求 | 无 | | |
| 编写人 |  | | |

## 2.3 非功能需求

### 2.3.1环境需求

开发环境需求如表2-40所示。

表2-40 开发环境需求表

|  |  |
| --- | --- |
| 要求 | 内容 |
| 操作系统 | Windows 10 1903及以上版本 |
| JDK | 7.0.800.15及其以上版本 |
| 开发工具 | Eclipse1.3.100及以上版本 |
| 浏览器 | IE、Microsoft Edge、Chrome |
| CPU | Intel Core i5-4200H处理器及以上配置 |
| 内存 | 4GB(2G\*2) DDR3L-1333及以上配置 |

服务器环境需求如表2-41所示。

表2-41 服务器环境需求表

|  |  |
| --- | --- |
| 要求 | 内容 |
| 操作系统 | Red Hat Enterprise Linux 7.8 |
| 硬盘 | SCSI 硬盘144Gbytes及以上配置 |
| CPU | Intel Xeon 5500及以上配置 |

客户端运行环境需求如表2-42所示。

表2-42 客户端运行环境需求表

|  |  |
| --- | --- |
| 要求 | 内容 |
| CPU | Intel Core i5-4200H处理器及以上配置 |
| 内存 | 4GB(2G\*2) DDR3L-1333及以上配置 |
| 硬盘 | 256G SSD及以上配置 |
| 显卡 | NVIDIA GT940M 2GB 及以上配置 |

### 2.3.2 性能需求

（1）技术可行性

技术的可行性是指在当今所拥有的技术条件之下，计算机软件和硬件的发展要能够满足本平台设计要求。由于本系统开发基于jsp语言，在一般的计算机上就能够满足开发本系统所需要的软硬件条件，所以在硬件是可行的；本系统在技术上都是用了目前市场上常用的技术，所有在软件方面也是可行的。

（2）经济可行性

对系统从开发者的经济状况和用户的使用费用进行分析。本系统不需要消耗大量的人力以及物力，开发前期也不需要耗费过多的财力，开发此软件不需要大量经费。由于系统开发的工具和软件都是可以免费使用的，因此系统开发的费用很少，甚至只需要少许的费用就可以开发出本系统，也可以忽略不计。因此具有经济可行性。

1. 吞吐量

在数据库方面，本系统中数据库在5秒内执行任意SQL语句不低于10条。

（4）可扩展性

功能扩展：系统从系统结构、功能设计、管理对象等各方面的功能扩展来考虑，以满足用户今后系统扩充和扩大使用范围的要求。

### 2.3.3 安全需求

（1）系统保密性

本系统中，对于每一种不同的的功能，都需要特定的用户能够进行相关的操作，从防止防止信息的非法泄露。

（2）可用性和抗毁性

本系统中，如果出现操作失败，则此时备份机制和容错机制应当介入，从而保证系统不出错。

（3）系统完整性

本系统需要保证所有的信息使用对象为其原形被授权的用户，也只有这些用户能进行修改等特殊操作。

3 总体设计

## 3.1系统设计的原则

在充分考虑到本系统的总体规划目标，以及系统的应用环境前提下，本系统的设计将严格遵循以下的设计原则：

（1）实用性原则。它的好坏直接关系到该平台是否能正常使用。所以在系统设计的时候，我们要把适用性原则放在比较重要的位置。本系统能够满足用户基本功能需求，以求解决掉以往的招商引资的繁琐步骤。

（2）规范性原则。系统中的所有设计都严格遵循国家标准、行业标准，符合软件工程中的设计原则与规范。所有文档的书写也都符合相关要求。

（3）经济性原则。经济性原则是指在完成系统要求的前提下，尽可能减少系统的费用支出。一方面，在系统硬件投资上不能盲目追求技术上的先进，而应以满足系统需要为前提。另一方面，系统设计中应避免不必要的复杂化，各模块应尽可能简洁。

## 3.2 系统体系结构设计

本系统采用的系统体系结构是B/S结构(浏览器-服务器)。B/S结构如图3-1所示。

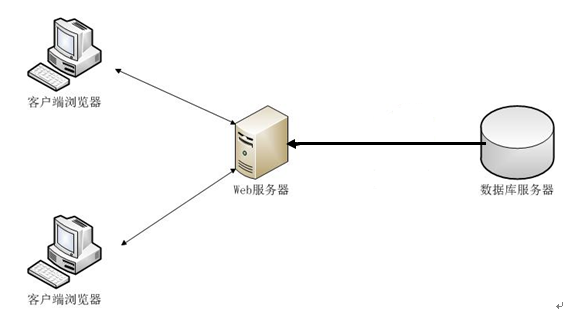


图3-1 B/S体系结构图

在B/S体系结构中，除了数据库服务器外，应用程序网页的形式存放于Web服务器上，用户运行某个应用程序时只需在客户端上的浏览器中输入相应的网址。调用Web服务器上的应用程序并对数据库进行操作完成相应的数据处理工作，最后将结果通过浏览器显示给用户。

本系统采用的框架是SSM框架，即Spring+SpringMVC+MyBatis，在这个框架中，他们各自有不同的作用：

Spring就像一个大工厂，也是项目中的粘合剂，它可以指定参数去调用实体类的构造方法来实例化对象。

SpringMVC的职责就像一个前台，将用户的请求传送到Controller，然后Controller就会根据不同的请求执行对应的操作。

MyBatis是对jdbc的封装，让数据库的操作变得透明，对各种数据进行相应的操作。

SSM框架结构图如图3-2所示。



图3-2 SSM框架结构图

## 3.3 系统功能结构设计

本系统的系统总包图如图3-3所示。

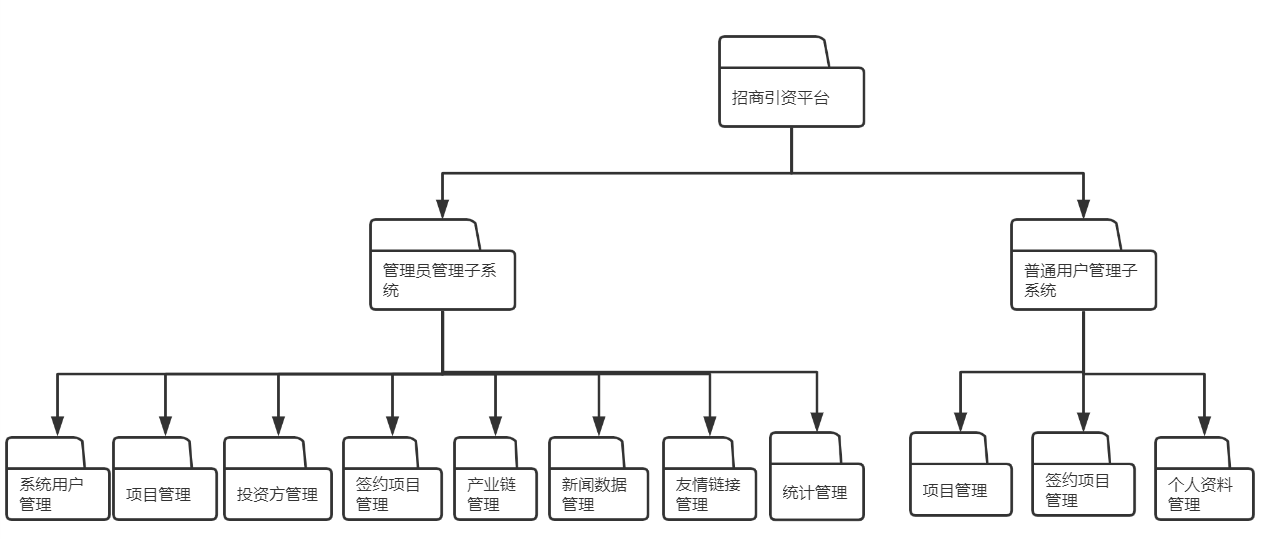


图3-3 系统总包图

如上图所示，系统一共分为两个子系统，管理员管理子系统和普通用户管理子系统。其中，管理员管理系统又分为系统用户管理、项目管理、投资方管理、签约项目管理、产业链管理、新闻信息管理、友情链接管理、统计管理共8个模块。系统用户管理包括新增用户、删除用户、修改用户、查找用户。项目管理包括删除项目、修改项目、查找项目。投资方管理包括新增投资方、删除投资方、修改投资方、查找投资方。签约项目管理包括删除签约项目、修改签约项目、查找签约项目。产业链管理包括新增产业链、删除产业链、修改产业链、查找产业链。新闻信息管理包括新增新闻信息、删除新闻信息、修改新闻信息、查找新闻信息。友情链接管理包括新增友情链接、删除友情链接、修改友情链接、查看友情链接。数据统计管理包括查看统计结果。普通用户管理子系统又分为项目管理、签约项目管理、个人资料管理共3个模块。项目管理包括新增项目、删除项目、修改项目、查找项目。签约项目管理包括新增签约项目、删除签约项目、修改签约项目、查找签约项目。个人资料管理包括修改个人资料。

# **4 数据库设计**

## 4.1 概念结构设计

一个完整的数据分析过程与这个数据库的逻辑结构和物理结构有十分密切的联系，需要完成对应用数据库的信息进行确认、编纂、组织、挑选。我们可以采用特定的方法来获取有关用户的所有的有效信息，从而使这些有用信息可以直接成为数据库的重要数据。

### 4.1.1设计思路

系统里最为关键的部分是数据库方面，需要十分清晰的思路，从开始的设计时就需要做到确立模块之间的联系，从而可以很明确的建立表间的联系和表中所需的内容。设计数据库还有一点是减少表的繁杂的创建，所以联系很关键，可以大大的减少数据表中重复的事项。由此可以保证数据的完整和统一，不会造成数据的错误和重复，并且可以使数据得到数据库安全的保护。

我们在数据库分析中采用E-R图来建立相对应的关系，它有实体、关系、属性三种概念。实体可以说是一个用来维持和不断利用数据的大集合；属性可以说是这个物体的所具有的特点；关系很明显指的就是每个数据实体之间的种种联系，关系给各个实体之间建立起了相对应的模型。根据系统的需求分析和数据库的概念，建立E-R图。

### 4.1.2 E-R图

招商引资平台系统的总体E-R图如下图4-1所示。

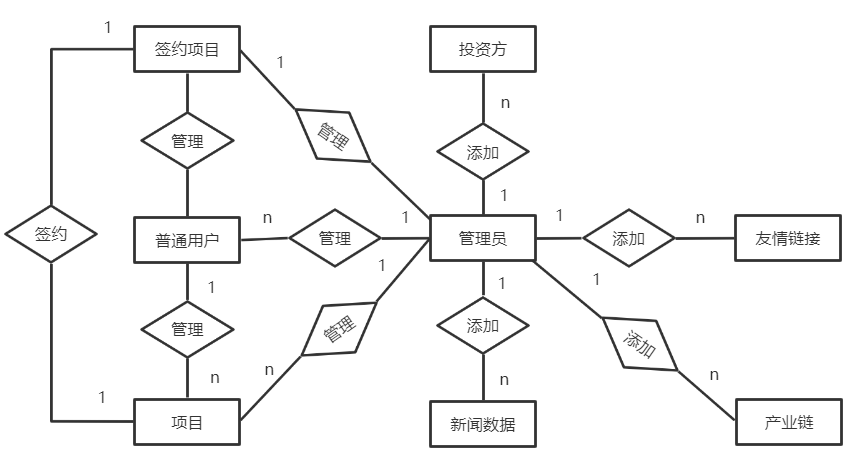


图4-1 总体E-R图

### 4.1.3主要实体属性图

（1）用户实体类属性包括用户编号、用户账号、用户密码、用户名字、用户性别、用户手机号、用户身份证号、用户地址、用户职业，用户的实体类属性如图4-2所示。

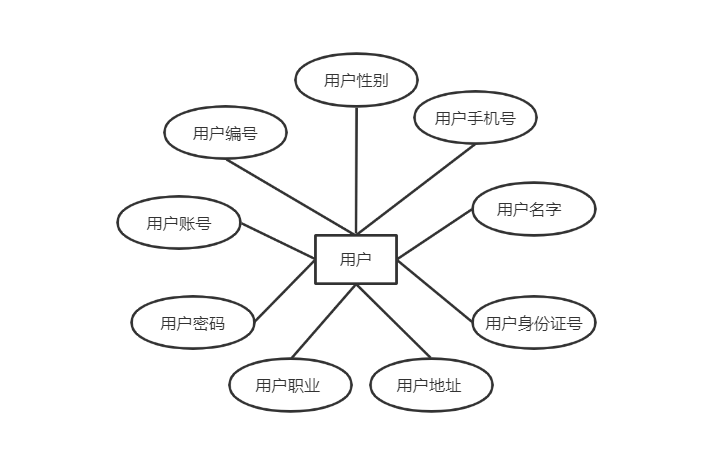


图4-2 用户实体属性图

（2）项目实体类属性包括项目id、项目编号、项目名称、项目类别、项目内容、联系方式、项目状态、发布人、发布时间，项目的实体属性图如图4-3所示。

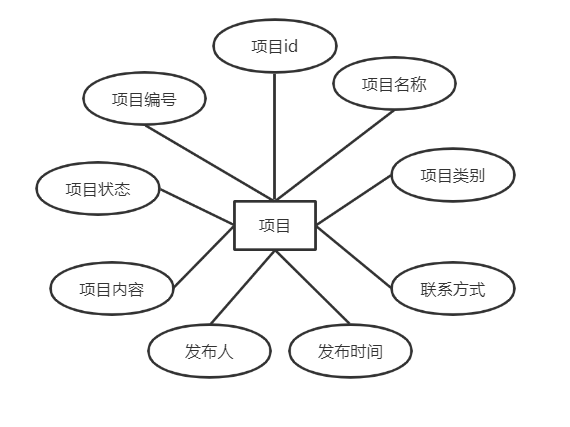


图4-3 项目实体属性图

（3）投资方实体属性包括投资方id、投资方编号、投资方名称、投资方分类、投资意向、联系方式、投资方地址，投资方的实体属性图如图4-4所示。

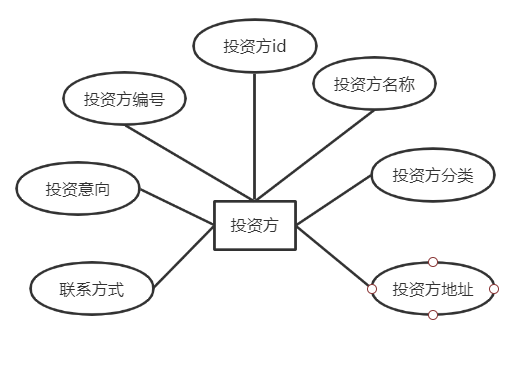


图4-4 投资方实体属性图

（4）产业链实体属性包括产业链id、产业链编号、项目类别、上游产品、中游产品、下游产品、时间、备注，产业链的实体属性图如图4-5所示。

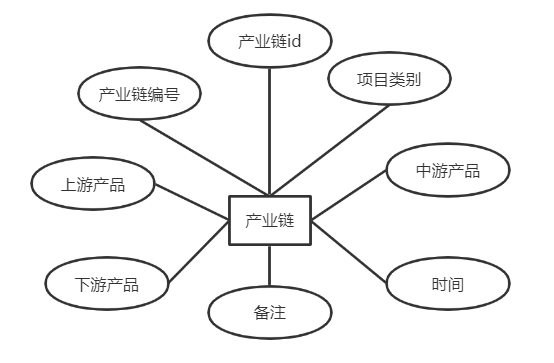


图4-5 产业链实体属性图

（5）新闻数据实体属性包括新闻id、新闻标题、新闻内容、图片、点击率、新闻类别、摘要、发布时间，新闻实体属性图如图4-6所示。

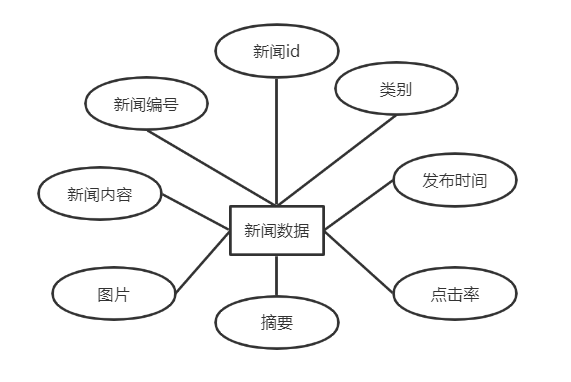


图4-6 新闻数据实体属性图

（6）签约项目实体属性包括项目id、项目编号、项目类别、项目内容、项目状态、联系方式、发布时间、发布人，签约项目实体属性图如图4-7所示。

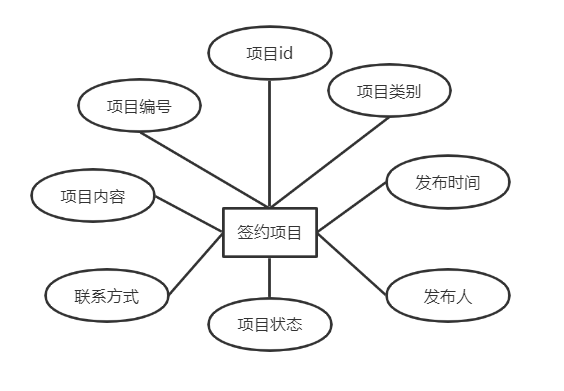


图4-7 签约项目实体属性图

（7）友情链接实体属性包括ID、网站名称、网址、发布时间、logo，友情链接实体属性图如图4-8所示。

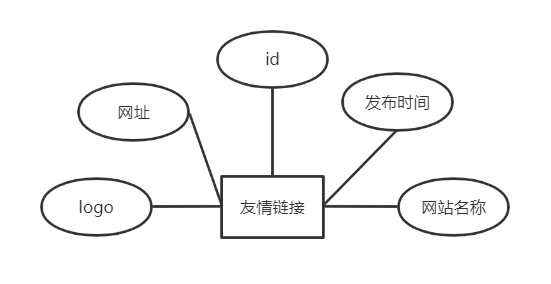


图4-8 友情链接实体属性图

## 4.2 逻辑结构设计

### 4.2.1 设计思路

数据库的逻辑结构设计就是将概念结构转换为某个DBMS所支持的数据模型。本系统根据各种概念结构之间的关系构建出相应的数据模型。

### 4.2.2 逻辑模型

由实体属性图得到的关系模型如下：

1. 用户（用户id、用户账号、用户密码、用户名字、用户性别、用户职业、联系方式、用户地址、用户身份证号）。
2. 项目（项目id、项目编号、项目名称、项目内容、项目类别、发布人、发布时间、项目状态、联系方式）。
3. 投资方（投资方id、投资方编号、投资方名称、投资方分类、投资意向、联系方式、投资方地址）。
4. 产业链（产业链id、产业链编号、项目类别、上游产品、中游产品、下游产品、时间、备注）。
5. 签约项目（项目id、项目编号、项目类别、项目内容、项目状态、联系方式、发布时间、发布人）。
6. 新闻数据（新闻id、新闻标题、新闻内容、图片、点击率、新闻类别、摘要、发布时间）。
7. 友情链接（链接ID、网站名称、网址、发布时间、logo）。

根据以上主要关系模型设计了如下表来处理本系统模块功能，其中7张数据表为关键数据表，数据总表如表4-1所示。

表4-1 系统数据表

|  |  |
| --- | --- |
| 表名 | 功能说明 |
| allusers | 管理员表 |
| yonghuxinxi | 用户表 |
| xiangmuxinxi | 项目信息表 |
| chanyelianxinxi | 产业链信息表 |
| touzifang | 投资方表 |
| xiangmuqianyue | 签约项目表 |
| xinwentongzhi | 新闻数据表 |
| wangongxiangmu | 完工项目表 |
| youqinglianjie | 友情链接表 |

1. 管理员表

管理员表字段包括管理员id、管理员名字、管理员密码、注册时间。具体表结构如表4-2所示。

表4-2 管理员表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段类型 | 大小 | 允许为空 | 最大长度 | 备注 |
| ID | int | 4 | 否 | 4 | 主键ID |
| username | varchar | 20 | 否 | 50 | 用户名 |
| pwd | varchar | 20 | 否 | 50 | 密码 |
| cx | varchar | 20 | 否 | 50 | 权限 |
| addtime | datetime | 8 | 否 | 8 | 添加时间 |

1. 用户表

用户表字段包括用户id、用户账号、用户密码、用户名字、用户性别、用户职业、联系方式、用户地址、用户身份证号。具体表结构如表4-3所示。

表4-3 用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **大小** | **允许为空** | **最大长度** | **备注** |
| id | int | 4 | 否 | 4 | 主键ID |
| yonghuming | varchar | 20 | 否 | 50 | 用户名 |
| mima | varchar | 8 | 是 | 20 | 密码 |
| xingming | varchar | 20 | 是 | 50 | 姓名 |
| xingbie | varchar | 20 | 是 | 50 | 性别 |
| dianhua | varchar | 20 | 是 | 50 | 电话 |
| shenfenzheng | varchar | 20 | 是 | 50 | 身份证 |
| dizhi | varchar | 20 | 是 | 50 | 地址 |
| addtime | datetime | 8 | 否 | 8 | 添加时间 |

1. 项目信息表

项目信息字段包括项目id、项目编号、项目名称、项目内容、项目类别、发布人、发布时间、项目状态、联系方式。具体表结构如表4-4所示。

表4-4 项目信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **大小** | **允许为空** | **最大长度** | **备注** |
| id | int | 4 | 否 | 8 | 主键ID |
| xiangmubianhao | varchar | 50 | 是 | 50 | 项目编号 |
| xiangmumingcheng | varchar | 50 | 否 | 50 | 项目名称 |
| xiangmuleibie | varchar | 50 | 是 | 50 | 项目类别 |
| xiangmuneirong | text | 8 | 否 | 8 | 项目内容 |
| lianxifangshi | varchar | 50 | 否 | 50 | 联系方式 |
| xiangmuzhuangtai | varchar | 50 | 是 | 50 | 项目状态 |

续表4-4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| fabushijian | datetime | 8 | 否 | 8 | 发布时间 |
| faburen | varchar | 50 | 是 | 50 | 发布人 |
| addtime | datetime | 8 | 否 | 8 | 添加时间 |

1. 投资方信息表

投资方信息字段包括投资方id、投资方编号、投资方名称、投资方分类、投资意向、联系方式、投资方地址。具体表结构如表4-5所示。

表4-5 投资方信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **大小** | **允许为空** | **最大长度** | **备注** |
| id | int | 4 | 否 | 8 | 主键ID |
| touzifangbianhao | varchar | 50 | 是 | 50 | 投资方编号 |
| touzifangmingcheng | varchar | 50 | 否 | 50 | 投资方名称 |
| touzifangfenlei | varchar | 50 | 是 | 50 | 投资方分类 |
| touzixiangmuyixiang | text | 8 | 否 | 8 | 投资项目意向 |
| lianxifangshi | varchar | 50 | 否 | 50 | 联系方式 |
| dizhi | varchar | 50 | 否 | 50 | 地址 |
| addtime | datetime | 8 | 否 | 8 | 添加时间 |

1. 产业链信息表

产业链信息字段包括产业链id、产业链编号、项目类别、上游产品、中游产品、下游产品、时间、备注。具体表结构如表4-6所示。

表4-6 产业链信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **大小** | **允许为空** | **最大长度** | **备注** |
| id | int | 4 | 否 | 8 | 主键ID |
| bianhao | varchar | 50 | 是 | 50 | 编号 |
| xiangmuleibie | varchar | 50 | 否 | 50 | 项目类别 |
| shangyou | varchar | 50 | 否 | 50 | 上游 |
| zhongyou | varchar | 50 | 否 | 50 | 中游 |
| xiayou | varchar | 50 | 否 | 50 | 下游 |
| beizhu | text | 8 | 是 | 8 | 备注 |
| addtime | datetime | 8 | 否 | 8 | 添加时间 |

1. 新闻信息表

新闻信息字段包括新闻id、新闻标题、新闻内容、图片、点击率、新闻类别、摘要、发布时间。具体表结构如表4-7所示。

表4-7 新闻信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **大小** | **允许为空** | **最大长度** | **备注** |
| id | int | 4 | 否 | 4 | 主键ID |
| biaoti | varchar | 50 | 否 | 300 | 标题 |
| leibie | varchar | 20 | 否 | 50 | 类别 |
| neirong | text | 8 | 否 | 8 | 内容 |
| tianjiaren | varchar | 20 | 否 | 50 | 添加人 |
| shouyetupian | varchar | 20 | 否 | 50 | 图片 |
| dianjilv | int | 4 | 否 | 4 | 点击率 |
| addtime | datetime | 8 | 否 | 8 | 添加时间 |

1. 签约项目信息表

签约项目字段包括项目id、项目编号、项目类别、项目内容、项目状态、联系方式、发布时间、发布人。具体表结构如表4-8所示。

表4-8 签约项目表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **大小** | **允许为空** | **最大长度** | **备注** |
| id | int | 4 | 否 | 8 | 主键ID |
| xiangmubianhao | varchar | 50 | 是 | 50 | 项目编号 |
| xiangmumingcheng | varchar | 50 | 是 | 50 | 项目名称 |
| xiangmuleibie | varchar | 50 | 是 | 50 | 项目类别 |
| xiangmuneirong | varchar | 50 | 是 | 50 | 项目内容 |
| lianxifangshi | varchar | 50 | 是 | 50 | 联系方式 |
| xiangmuzhuangtai | varchar | 50 | 是 | 50 | 项目状态 |
| fabushijian | varchar | 50 | 是 | 50 | 发布时间 |
| faburen | varchar | 50 | 是 | 50 | 发布人 |
| addtime | datetime | 8 | 否 | 8 | 添加时间 |

1. 友情链接信息表

友情链接字段包括ID、网站名称、网址、发布时间、logo。具体表结构如表4-9所示。

表4-9 友情链接表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **大小** | **允许为空** | **最大长度** | **备注** |
| id | int | 4 | 否 | 8 | 主键ID |
| wangzhanmingcheng | varchar | 50 | 是 | 50 | 网站名称 |
| wangzhi | varchar | 50 | 是 | 50 | 网址 |
| addtime | datetime | 50 | 是 | 50 | 发布时间 |

续表4-9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LOGO | image | 50 | 是 | 50 | logo图片 |

## 4.3 物理结构设计

### 4.3.1 存取方式

存取方式是快速存取数据库中的数据的技术，数据库管理系统一般都提供多种存取方法，具体采取哪种存取方法由系统根据数据的存储方式来决定，用户一般不能干预。

### 4.3.2 存储结构

确定数据库物理结构主要指确定数据的存放位置和存储结构，包括确定关系、索引、聚簇、日志、备份等的存储安排和存储结构，确定系统配置等，本系统采用MySQL数据库进行存储。

确定数据的存放位置和存储结构要综合考虑存取时间、存储空间利用率和维护代价三方面的因素。这三个方面常常是相互矛盾的，因此需要进行权衡选择一个折中方案。

# **5 界面设计**

## 5.1 界面关系图或工作流图

招商引资平台系统从浏览器输入指定URL即可进入网站首页，首页一开始可根据按钮到项目列表页，投资方列表页，新闻信息页，产业链信息页，通过选择可分类展示，可以在首页点击登录进入登录页，如果是第一次登录则需要注册，登录成功可进入后台主页，后台主页普通用户能访问页面有分别为个人资料管理页、项目管理页、签约项目管理页，管理页能访问的页面有系统用户管理页、项目管理页、投资方管理页、产业链管理页、新闻信息管理页、友情链接管理页、签约项目管理页、数据统计页。界面关系图如图5-1所示。

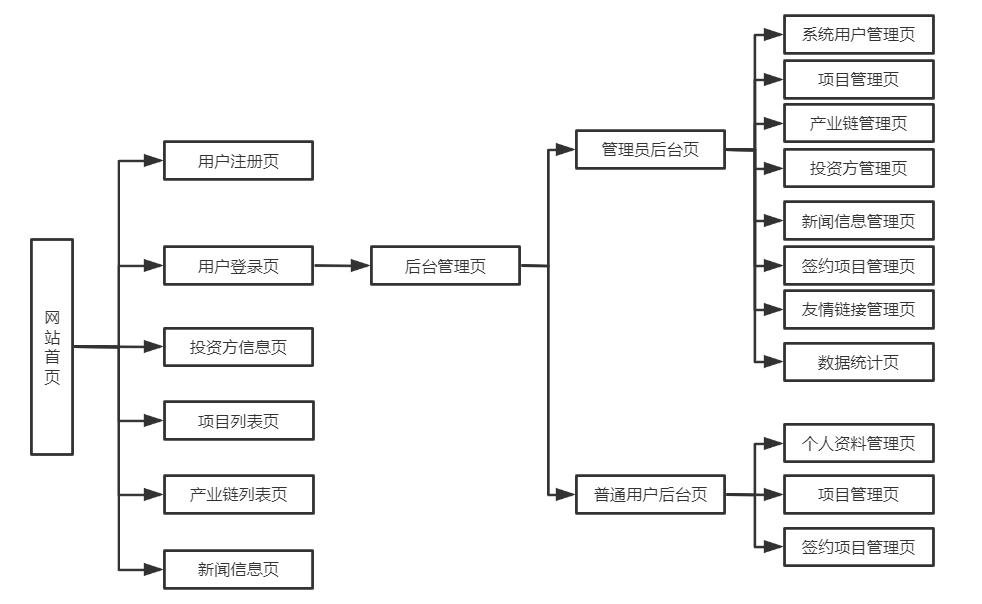


图5-1 界面关系图

## 5.2 界面设计成果

### 5.2.1 主界面

1. 用户主界面

本系统运行即进入网站主界面，如图5-2所示。



图5-2 主界面图

1. 管理页后台管理主页面

管理员登录后进入到管理员后台页面，如图5-3所示。



图5-3 管理员后台管理界面图

1. 普通用户后台管理界面

普通用户登录后进入到后台页面，如图5-4所示。

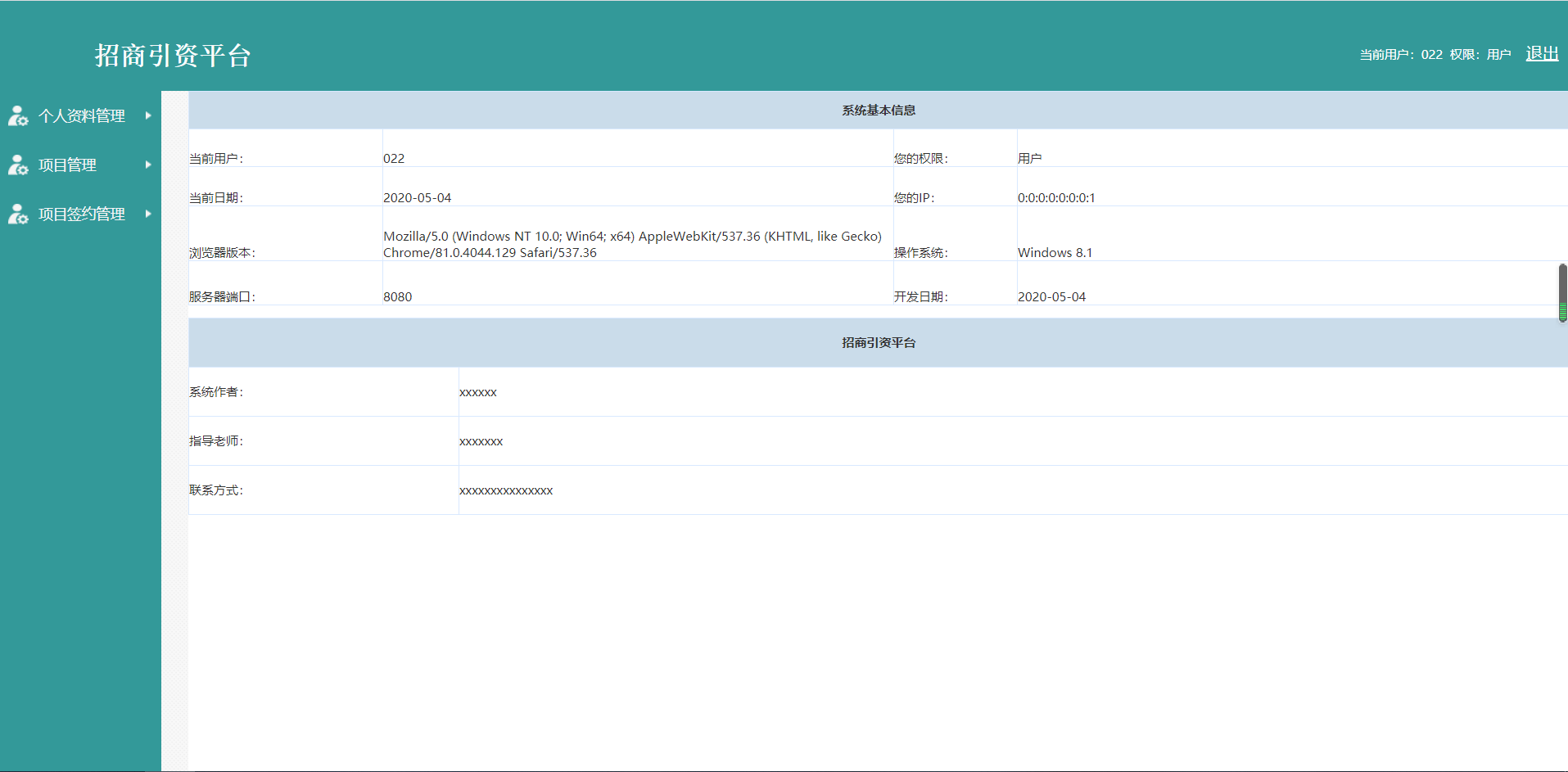


图5-4 普通用户后台管理界面图

### 5.2.2 子界面

（1）项目信息界面

进入系统后点击项目信息进入到项目信息页面，如图5-5所示。



图5-5 项目信息界面图

1. 投资方信息界面

进入系统后点击投资方进入到投资方信息页面，如图5-6所示。

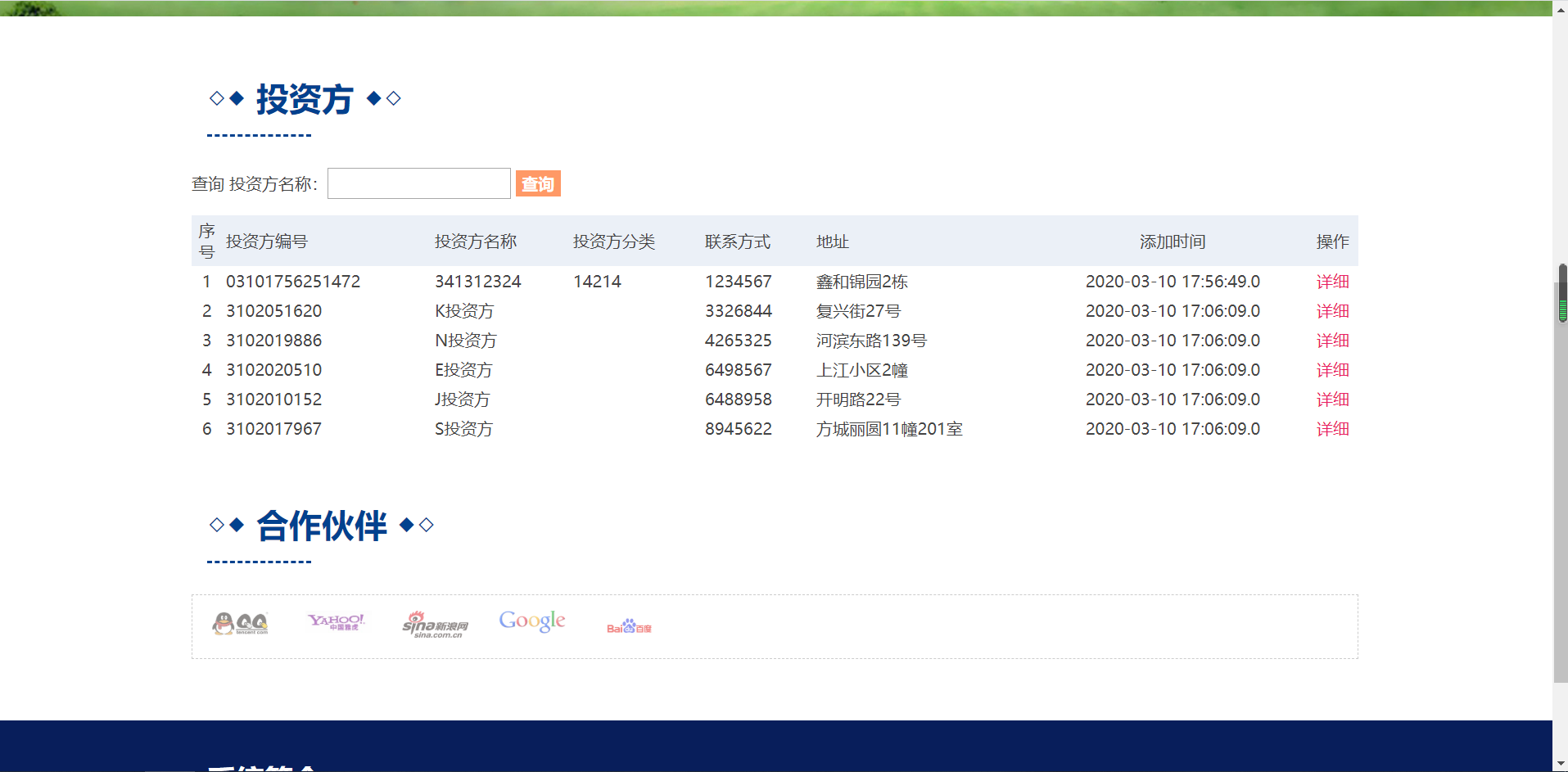


图5-6 投资方信息界面图

1. 产业链信息界面

进入系统后点击产业链进入到产业链信息页面，如图5-7所示。



图5-7 产业链信息界面图

6 详细设计

## 6.1 系统主要功能模块介绍

招商引资平台系统包括两个子系统，管理员管理子系统和普通用户管理子系统，其中管理员子系统中包括用户管理、项目管理、投资方管理、产业链管理、签约项目管理、友情链接管理、新闻数据管理、统计管理。普通用户管理子系统包括个人资料管理、项目管理、签约项目管理。系统的结构模块图如图6-1所示。

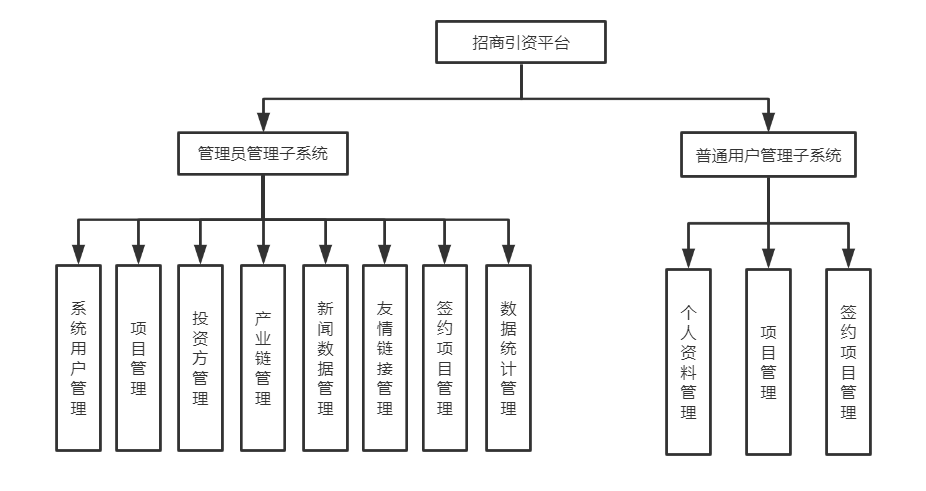


图6-1 系统模块结构图

## 6.2功能模块设计

### 6.2.1系统用户管理

（1）功能结构设计

系统用户管理包图如图6-2所示。

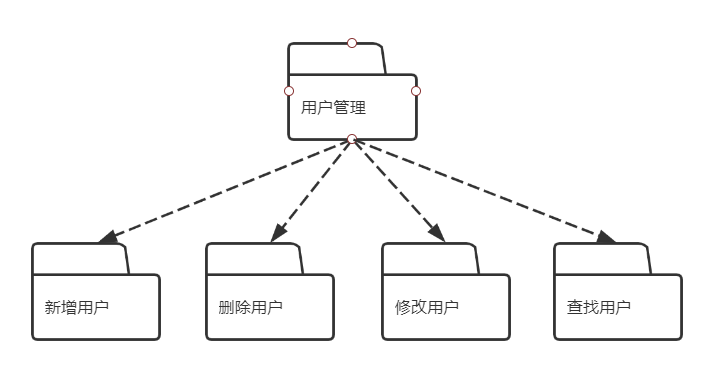


图6-2 系统用户管理包图

（2）类图设计

系统用户管理类图如图6-3所示。

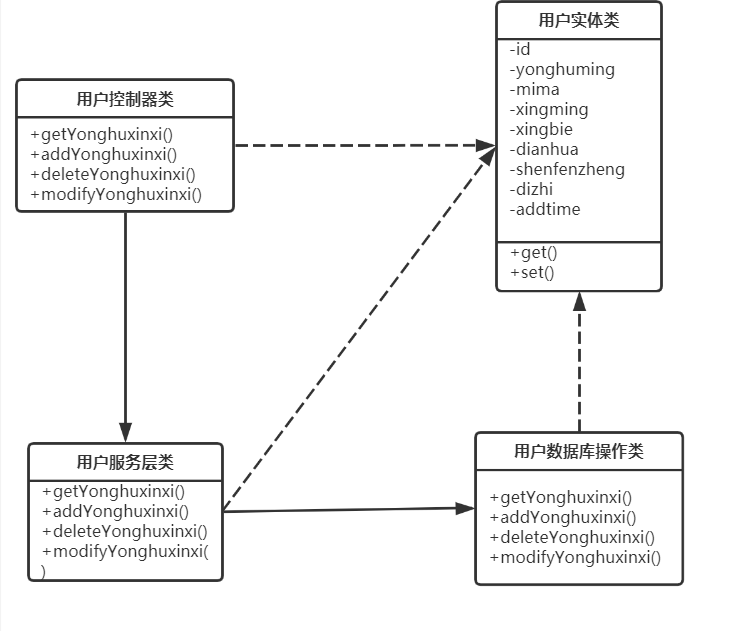


图6-3 系统用户管理类图

在系统用户管理类图中，主要涉及到四个类的关联，用户控制器类关联用户服务层类，同时关联用户实体类。用户服务层类关联用户数据库操作类，同时关联用户实体类。系统用户管理涉及到的各个类的描述如下所示，其中用户控制器类描述如表6-1所示。

表6-1 系统用户控制器类描述

|  |  |  |
| --- | --- | --- |
| 用户控制器类 | | |
| 类的描述：用户控制器类 | | |
| 操作描述 | addyonghuxinxi () | 新增用户 |
| ModifyMember() | 修改用户 |
| DeleteMember() | 删除用户 |
| getMember() | 获取用户 |

系统用户服务层类描述如表6-2所示。

表6-2 系统用户服务层类描述

|  |  |  |
| --- | --- | --- |
| 用户服务层类 | | |
| 类的描述：用户服务层类 | | |
| 操作描述 | addyonghuxinxi () | 新增用户 |
| ModifyMember() | 修改用户 |
| DeleteMember() | 删除用户 |
| getMember() | 获取用户 |

系统用户数据库操作类描述如表6-3所示。

表6-3 系统用户数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 用户数据库操作类 | | |
| 类的描述：用户数据库操作类 | | |
| 操作描述 | addyonghuxinxi () | 新增用户 |
| ModifyMember() | 修改用户 |
| DeleteMember() | 删除用户 |
| getMember() | 获取用户 |

（3）顺序图设计

系统用户管理顺序图如图6-4所示。

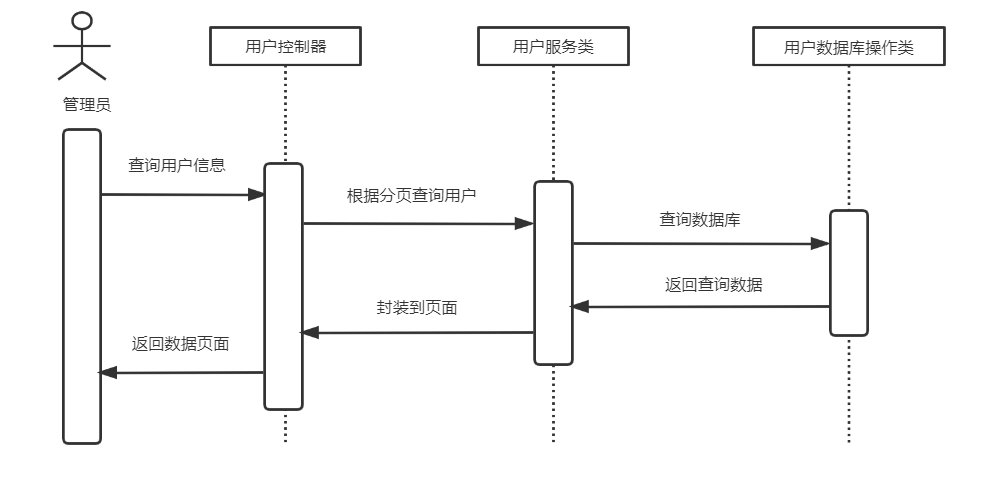


图6-4 系统用户管理顺序图

（4）核心处理流程设计

系统用户管理活动图如图6-5所示。

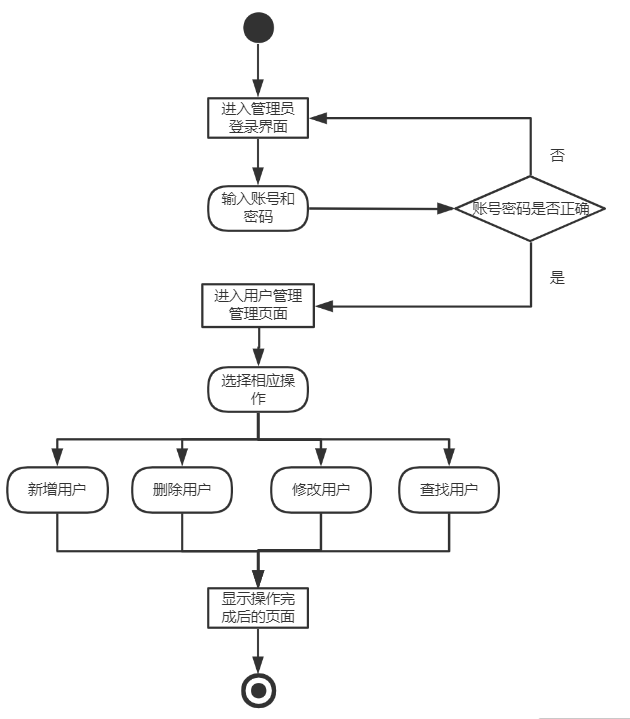


图6-5 用户管理活动图

### 6.2.2项目管理

（1）功能结构设计

项目管理包图如图6-6所示。

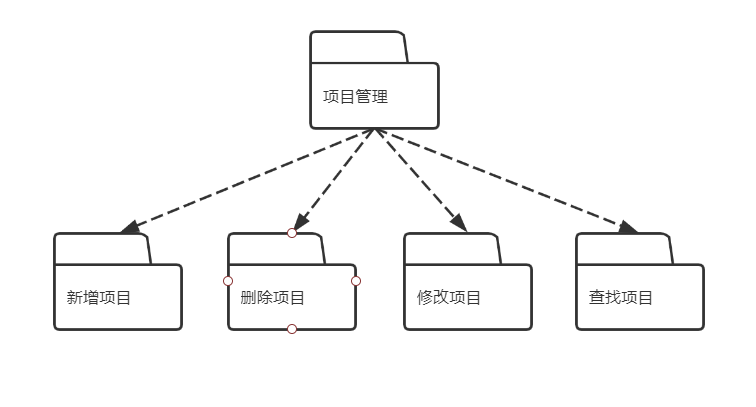


图6-6 项目管理包图

（2）类图设计

项目管理类图如图6-7所示。

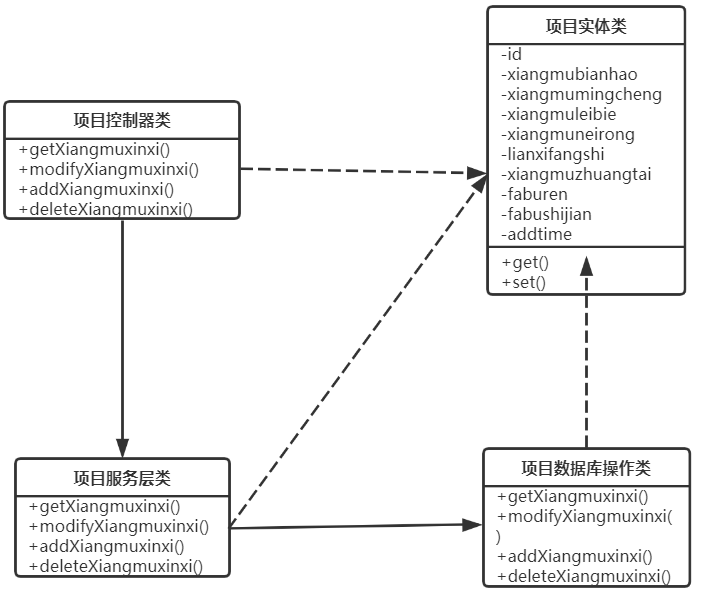


图6-7 项目管理类图

在项目管理类图中，主要涉及到四个类的关联，项目控制器类关联项目服务层类，同时关联项目实体类。项目服务层类关联项目数据库操作类，同时关联项目实体类。系统项目管理涉及到的各个类的描述如下所示，其中项目控制器类描述如表6-4所示。

表6-4 项目控制器类描述

|  |  |  |
| --- | --- | --- |
| 项目控制器类 | | |
| 类的描述：项目控制器类 | | |
| 操作描述 | addXiangmuxinxi () | 新增项目 |
| modifyXiangmuxinxir() | 修改项目 |
| deleteXiangmuxinxi() | 删除项目 |
| getXiangmuxinxi() | 获取项目 |

项目服务层类描述如表6-5所示。

表6-5 项目服务层类描述

|  |  |  |
| --- | --- | --- |
| 项目服务层类 | | |
| 类的描述：项目服务层类 | | |
| 操作描 | addXiangmuxinxi () | 新增项目 |
| modifyXiangmuxinxir() | 修改项目 |

续表6-5

|  |  |  |
| --- | --- | --- |
| 述 | deleteXiangmuxinxi() | 删除项目 |
| getXiangmuxinxi() | 获取项目 |

项目数据库操作类描述如表6-6所示。

表6-6 项目数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 项目数据库操作类 | | |
| 类的描述：项目数据库操作类 | | |
| 操作描述 | addXiangmuxinxi () | 新增项目 |
| modifyXiangmuxinxir() | 修改项目 |
| deleteXiangmuxinxi() | 删除项目 |
| getXiangmuxinxi() | 获取项目 |

（3）顺序图设计

项目管理顺序图如图6-8所示。

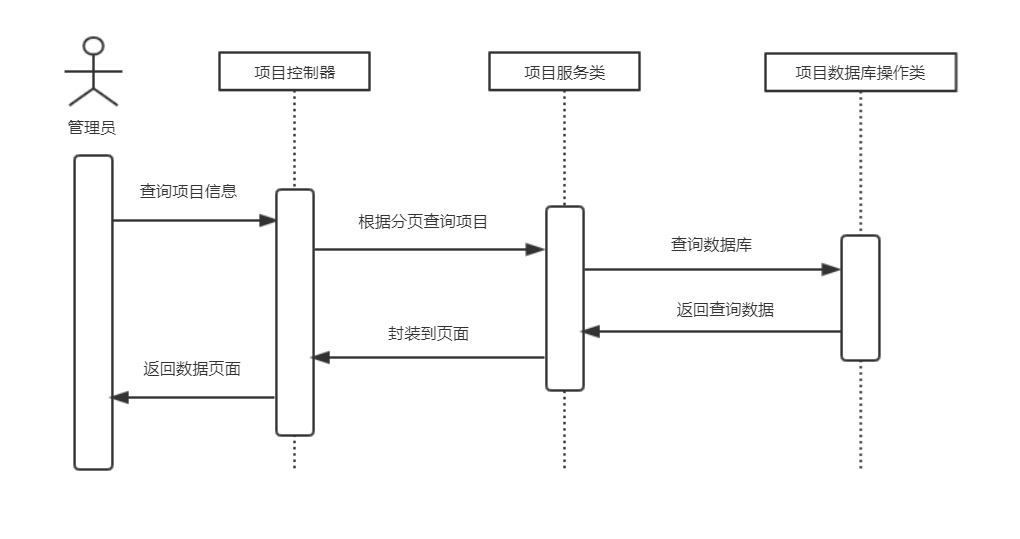


图6-8 项目管理顺序图

（4）核心处理流程设计

项目管理活动图如图6-9所示。

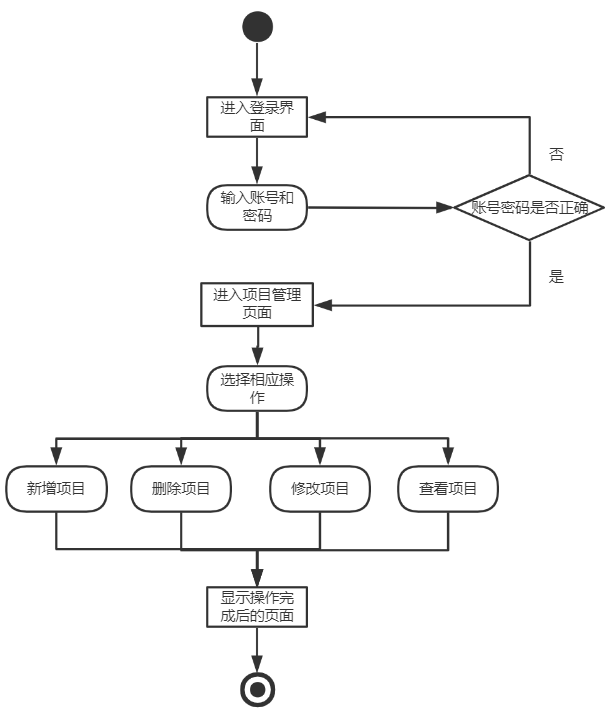


图6-9 项目管理活动图

### 6.2.3投资方管理

（1）功能结构设计

投资方管理包图如图6-10所示。

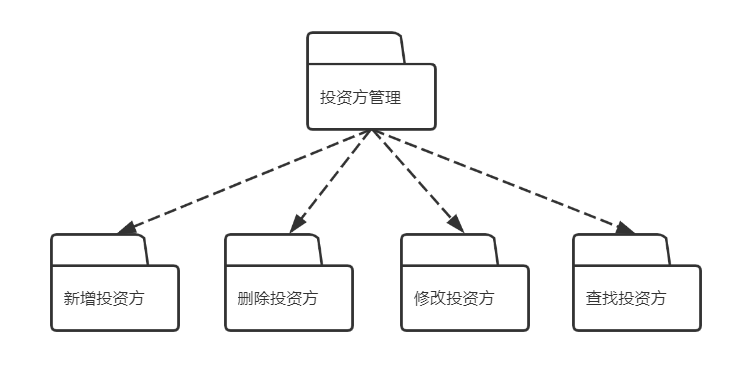


图6-10 投资方管理包图

（2）类图设计

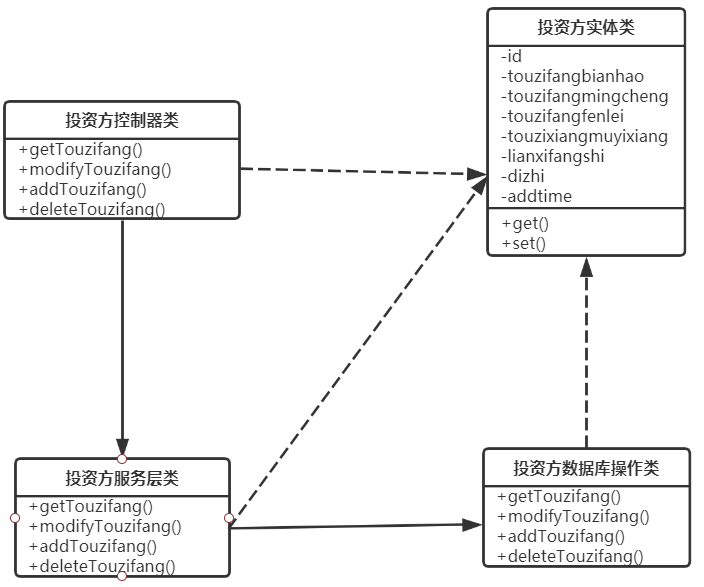


图6-11 投资方管理类图

在投资方管理类图中，主要涉及到四个类的关联，投资方控制器类关联投资方服务层类，同时关联投资方实体类。投资方服务层类关联投资方数据库操作类，同时关联投资方实体类。系统投资方管理涉及到的各个类的描述如下所示，其中投资方控制器类描述如表6-7所示。

表6-7 投资方控制器类描述

|  |  |  |
| --- | --- | --- |
| 投资方控制器类 | | |
| 类的描述：投资方控制器类 | | |
| 操作描述 | addTouzifang () | 新增投资方 |
| modifyTouzifang() | 修改投资方 |
| deleteTouzifang() | 删除投资方 |
| getTouzifang() | 获取投资方 |

投资方控制器类描述如表6-8所示。

表6-8 投资方服务层类描述

|  |  |  |
| --- | --- | --- |
| 投资方服务层类 | | |
| 类的描述：投资方服务层类 | | |
| 操作描 | addTouzifang () | 新增投资方 |
| modifyTouzifang() | 修改投资方 |

续表6-8

|  |  |  |
| --- | --- | --- |
| 述 | deleteTouzifang() | 删除投资方 |
| getTouzifang() | 获取投资方 |

投资方数据库操作类如表6-9所示。

表6-9 投资方数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 投资方数据库操作类 | | |
| 类的描述：投资方数据库操作类 | | |
| 操作描述 | addTouzifang () | 新增投资方 |
| modifyTouzifang() | 修改投资方 |
| deleteTouzifang() | 删除投资方 |
| getTouzifang() | 获取投资方 |

（3）顺序图设计

投资方管理顺序图如图6-12所示。

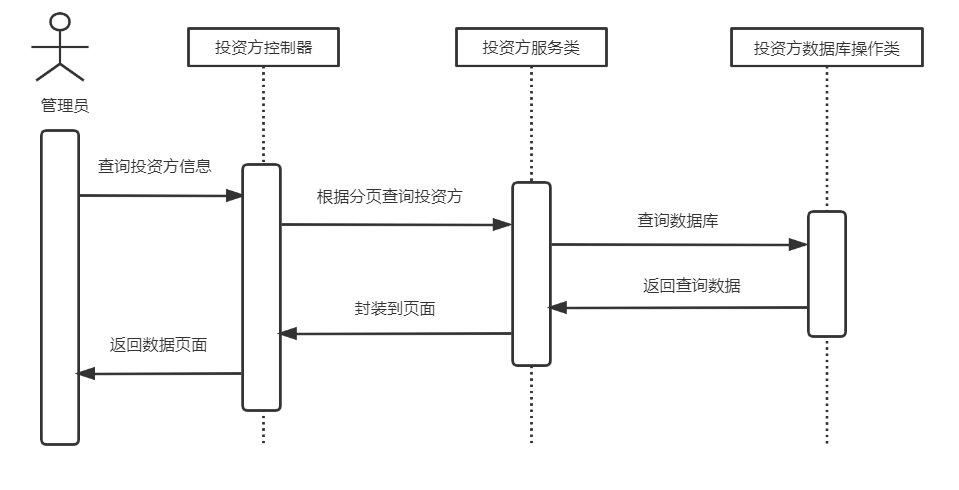


图6-12 投资方管理顺序图

（4）核心处理流程设计

投资方管理活动图如图6-13所示。

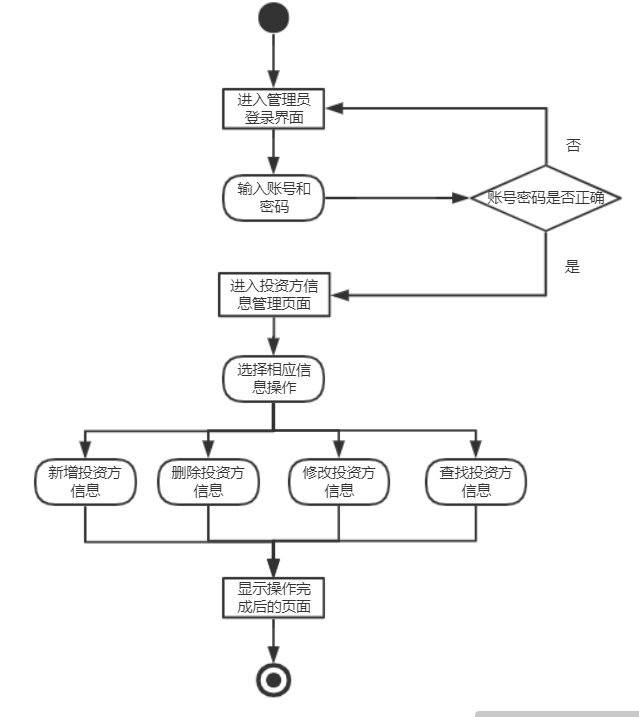


图6-13 投资方管理活动图

### 6.2.4签约项目管理

（1）功能结构设计

签约项目管理包图如图6-14所示。

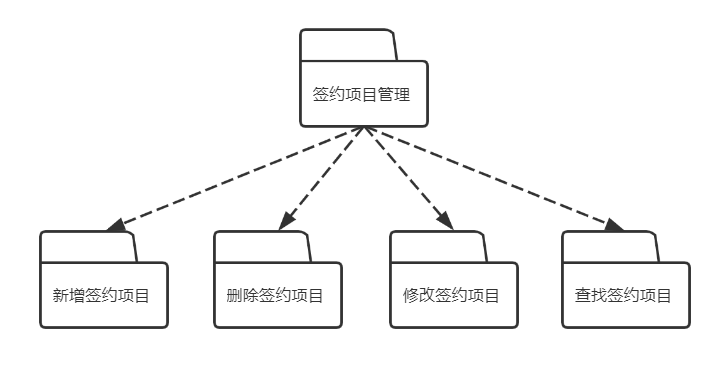


图6-14 签约项目管理包图

（2）类图设计

签约项目管理类图如图6-15所示。

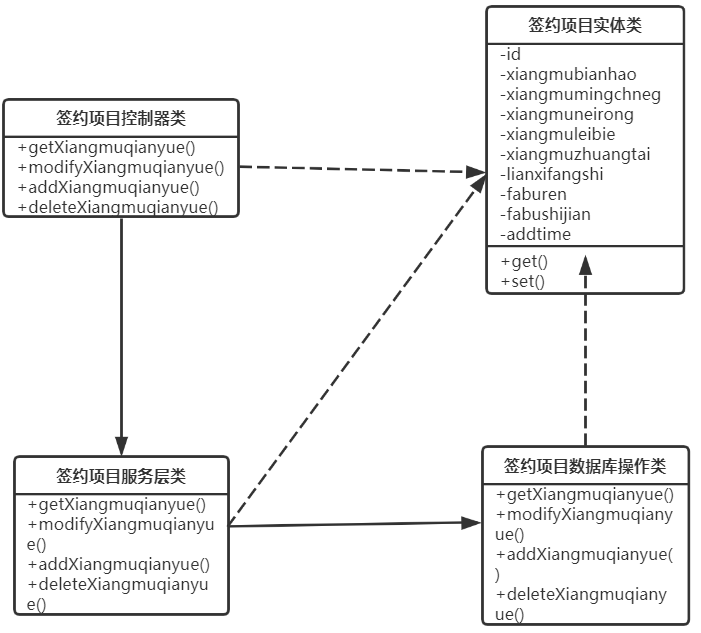


图6-15 签约项目管理类图

在签约项目管理类图中，主要涉及到四个类的关联，签约项目控制器类关联签约项目服务层类，同时关联签约项目实体类。签约项目服务层类关联签约项目数据库操作类，同时关联签约项目实体类。系统签约项目管理涉及到的各个类的描述如下所示，其中签约项目控制器类描述如表6-10所示。

表6-10 签约项目控制器类描述

|  |  |  |
| --- | --- | --- |
| 签约项目控制器类 | | |
| 类的描述：签约项目控制器类 | | |
| 操作描述 | addXiangmuqianyue () | 新增签约项目 |
| modifyXiangmuqianyue () | 修改签约项目 |
| deleteXiangmuqianyue () | 删除签约项目 |
| getXiangmuqianyue() | 获取签约项目 |

签约项目服务层类描述如表6-11所示。

表6-11 签约项目服务层类描述

|  |  |  |
| --- | --- | --- |
| 签约项目服务层类 | | |
| 类的描述：签约项目服务层类 | | |
| 操 | addXiangmuqianyue () | 新增签约项目 |

续表6-11

|  |  |  |
| --- | --- | --- |
| 作描述 | modifyXiangmuqianyue () | 修改签约项目 |
| deleteXiangmuqianyue () | 删除签约项目 |
| getXiangmuqianyue() | 获取签约项目 |

签约项目数据库操作类描述如表6-12所示。

表6-12 签约项目数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 签约项目服务层类 | | |
| 类的描述：签约项目服务层类 | | |
| 操作描述 | addXiangmuqianyue () | 新增签约项目 |
| modifyXiangmuqianyue () | 修改签约项目 |
| deleteXiangmuqianyue () | 删除签约项目 |
| getXiangmuqianyue() | 获取签约项目 |

（3）顺序图设计

签约项目管理顺序图如图6-16所示。

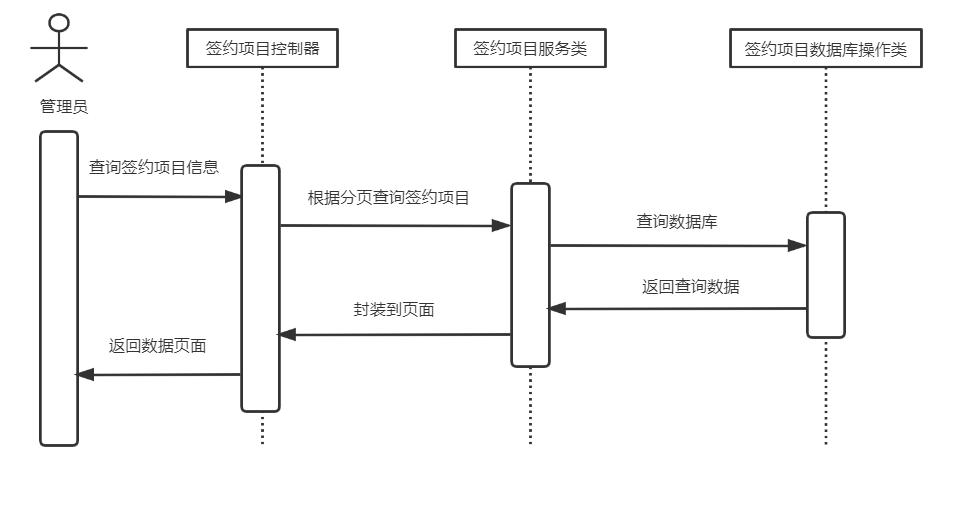


图6-16 签约项目管理顺序图

（4）核心处理流程设计

签约项目管理活动图如图6-17所示。

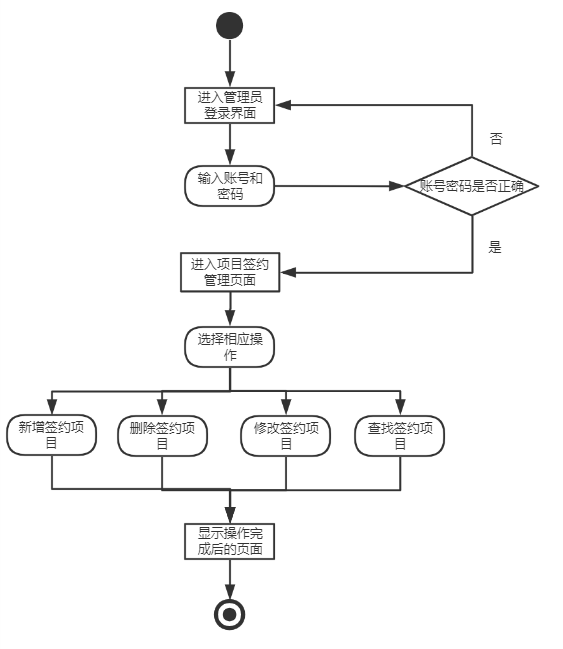


图6-17 签约项目管理活动图

### 6.2.5产业链管理

（1）功能结构设计

产业链管理包图如图6-18所示。

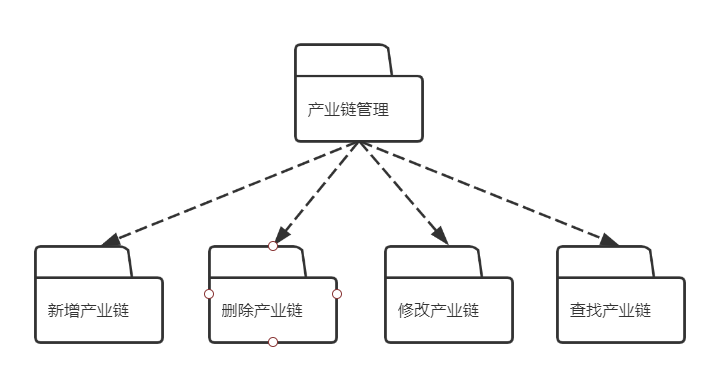


图6-18 产业链管理包图

（2）类图设计

产业链管理类图如图6-19所示。

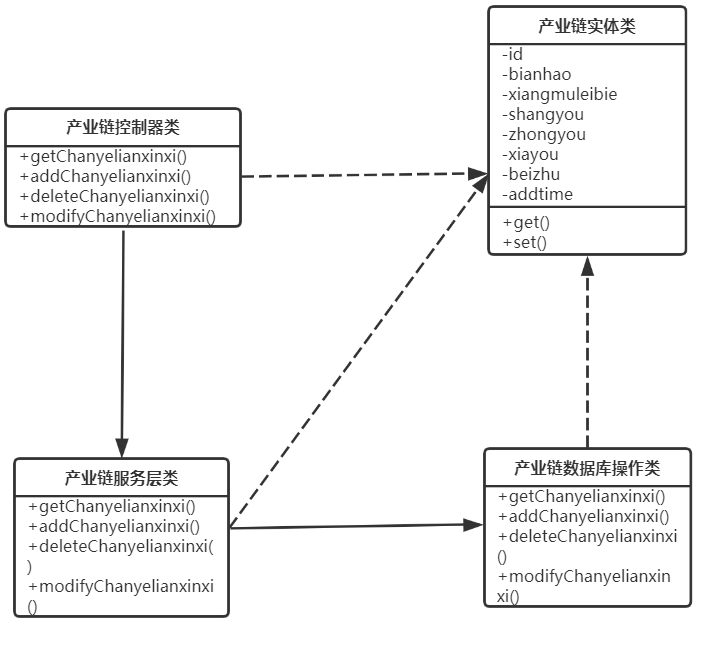


图6-19 产业链管理类图

在产业链管理类图中，主要涉及到四个类的关联，产业链控制器类关联产业链服务层类，同时关联产业链实体类。产业链服务层类关联产业链数据库操作类，同时关联产业链实体类。系统产业链管理涉及到的各个类的描述如下所示，其中产业链控制器类描述如表6-10所示。

表6-13 产业链控制器类描述

|  |  |  |
| --- | --- | --- |
| 产业链控制器类 | | |
| 类的描述：产业链控制器类 | | |
| 操作描述 | addChanyelianxinxi() | 新增产业链 |
| modifyChanyelianxinxi () | 修改产业链 |
| deleteChanyelianxinxi () | 删除产业链 |
| getChanyelianxinxi() | 获取产业链 |

产业链服务层类描述如表6-14所示。

表6-14 产业链服务层类描述

|  |  |  |
| --- | --- | --- |
| 产业链服务层类 | | |
| 类的描述：产业链服务层类 | | |
| 操作描 | addChanyelianxinxi() | 新增产业链 |
| modifyChanyelianxinxi () | 修改产业链 |

续表6-14

|  |  |  |
| --- | --- | --- |
| 述 | deleteChanyelianxinxi () | 删除产业链 |
| getChanyelianxinxi() | 获取产业链 |

产业链数据库操作类描述如表6-15所示。

表6-15 产业链数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 产业链数据库操作类 | | |
| 类的描述：产业链数据库操作类 | | |
| 操作描述 | addChanyelianxinxi() | 新增产业链 |
| modifyChanyelianxinxi () | 修改产业链 |
| deleteChanyelianxinxi () | 删除产业链 |
| getChanyelianxinxi() | 获取产业链 |

（3）顺序图设计

产业链管理顺序图如图6-20所示。

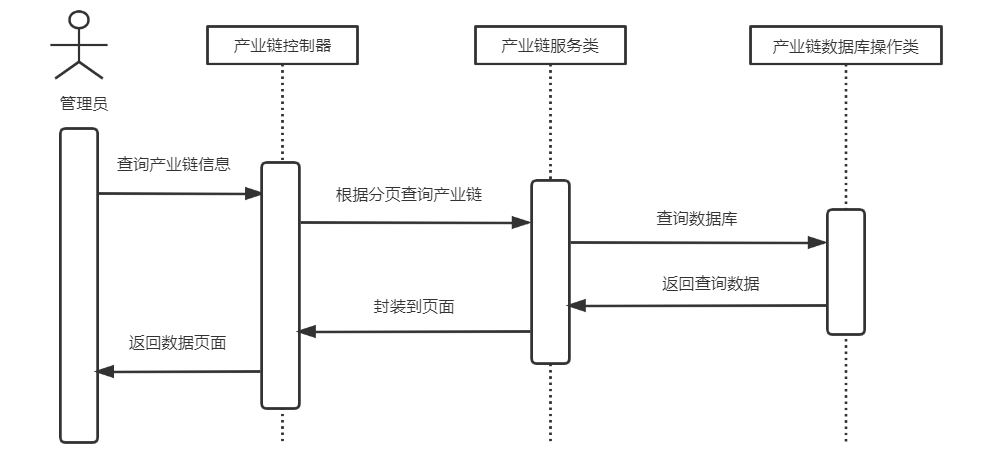


图6-20 产业链管理顺序图

（4）核心处理流程设计

产业链管理活动图如图6-21所示。

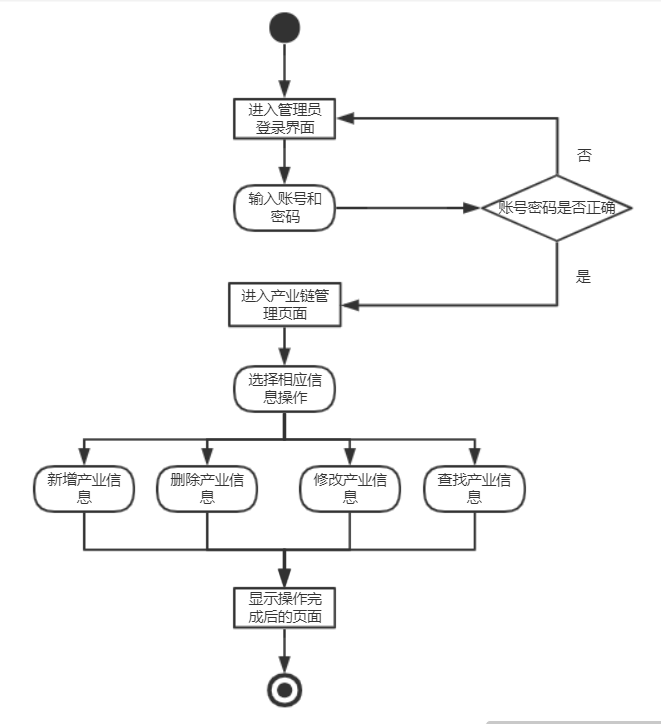


图6-21 产业链管理活动图

### 6.2.6新闻信息管理

（1）功能结构设计

新闻信息管理包图如图6-22所示。

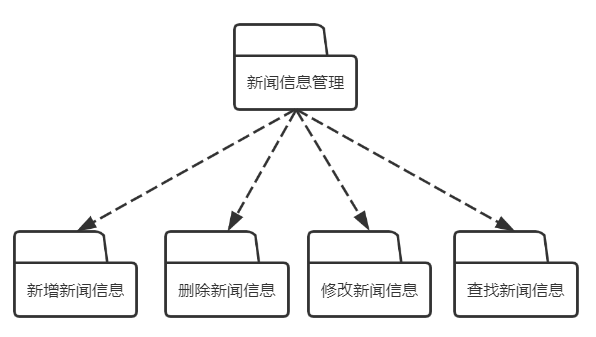


图6-22 新闻信息包图

（2）类图设计

新闻信息管理类图如图6-23所示。

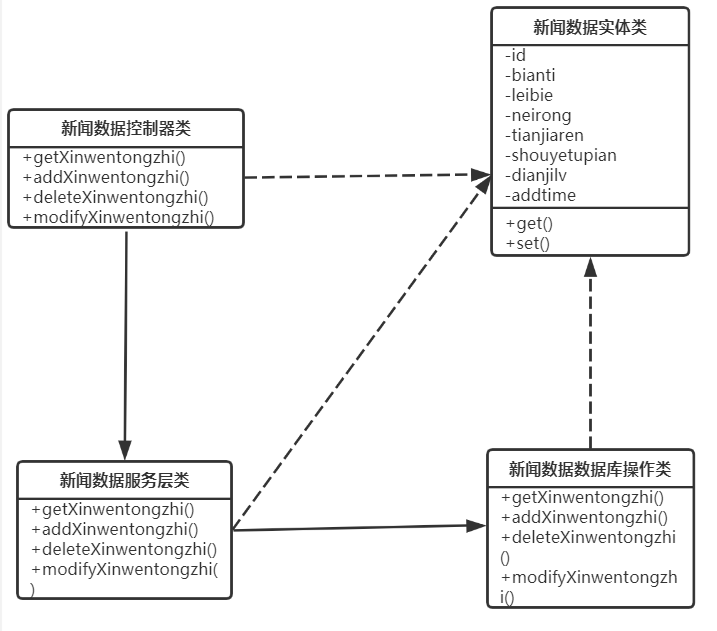


图6-23 新闻信息管理类图

在新闻信息管理类图中，主要涉及到四个类的关联，新闻信息控制器类关联新闻信息服务层类，同时关联新闻信息实体类。新闻信息服务层类关联新闻信息数据库操作类，同时关联新闻信息实体类。系统新闻信息管理涉及到的各个类的描述如下所示，其中新闻信息控制器类描述如表6-16所示。

表6-16 新闻信息控制器类描述

|  |  |  |
| --- | --- | --- |
| 新闻信息控制器类 | | |
| 类的描述：新闻信息控制器类 | | |
| 操作描述 | addXinwentongzhi() | 新增新闻数据 |
| modifyXinwentongzhi() | 修改新闻数据 |
| deleteXinwentongzhi() | 删除新闻数据 |
| getXinwentongzhi() | 获取新闻数据 |

新闻信息服务层类描述如表6-17所示。

表6-17 新闻数据服务层类描述

|  |  |  |
| --- | --- | --- |
| 新闻信息服务层类 | | |
| 类的描述：新闻信息服务层类 | | |
| 操 | addXinwentongzhi() | 新增新闻数据 |

续表6-17

|  |  |  |
| --- | --- | --- |
| 作描述 | modifyXinwentongzhi() | 修改新闻数据 |
| deleteXinwentongzhi() | 删除新闻数据 |
| getXinwentongzhi() | 获取新闻数据 |

新闻信息数据库操作类描述如表6-18所示。

表6-18 新闻数据数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 新闻信息数据库操作类 | | |
| 类的描述：新闻信息数据库操作类 | | |
| 操作描述 | addXinwentongzhi() | 新增新闻数据 |
| modifyXinwentongzhi() | 修改新闻数据 |
| deleteXinwentongzhi() | 删除新闻数据 |
| getXinwentongzhi() | 获取新闻数据 |

（3）顺序图设计

新闻信息管理顺序图如图6-24所示。

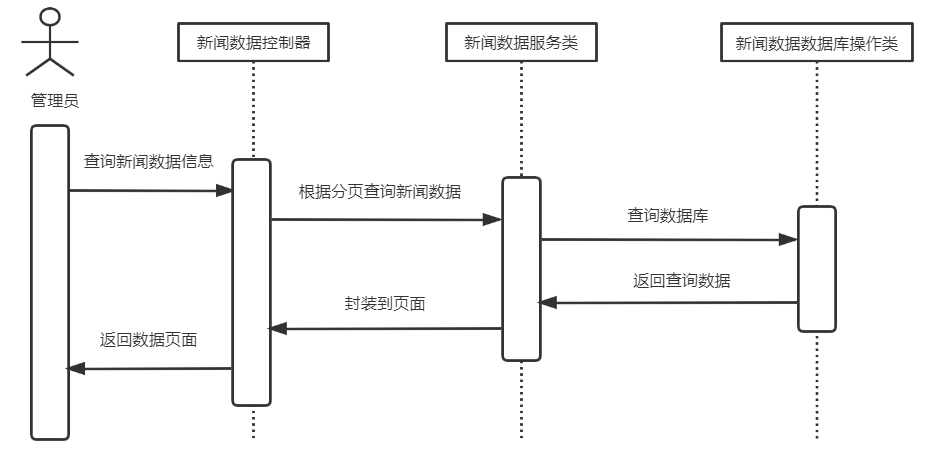


图6-24 新闻信息顺序图

（4）核心处理流程设计

新闻信息管理活动图如图6-25所示。

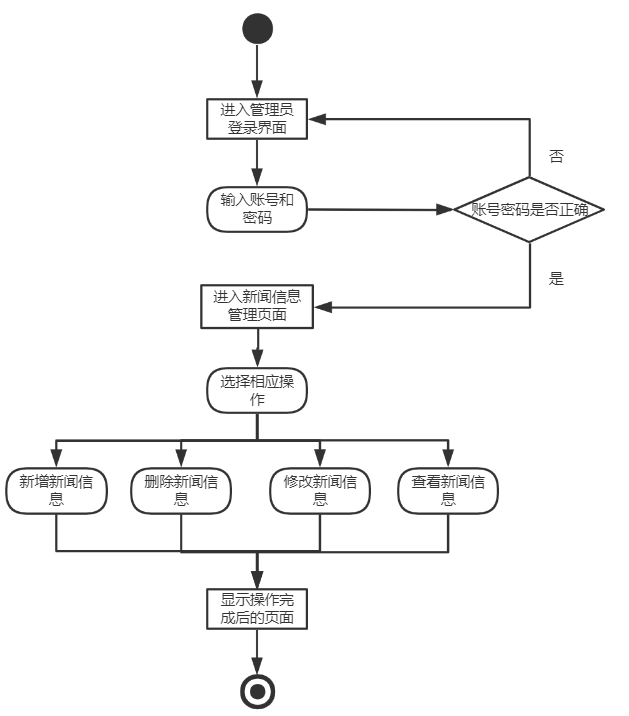


图6-25 新闻信息活动图

### 6.2.7友情链接管理

（1）功能结构设计

友情链接管理包图如图6-26所示。

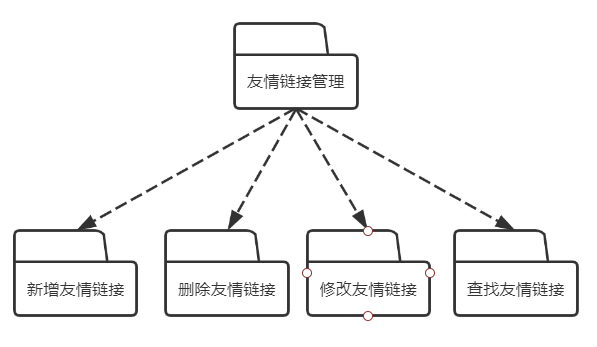


图6-26 友情链接管理包图

（2）类图设计

友情链接类图如图6-27所示。

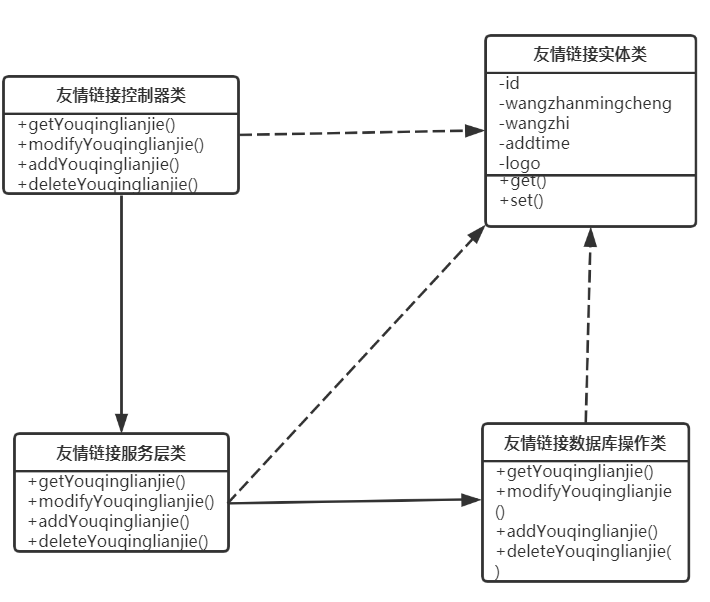


图6-27 友情链接管理类图

在友情链接管理类图中，主要涉及到四个类的关联，友情链接控制器类关联友情链接服务层类，同时关联友情链接实体类。友情链接服务层类关联友情链接数据库操作类，同时关联友情链接实体类。系统友情链接管理涉及到的各个类的描述如下所示，其中友情链接控制器类描述如表6-19所示。

表6-19 友情链接控制器类描述

|  |  |  |
| --- | --- | --- |
| 友情链接控制器类 | | |
| 类的描述：友情链接控制器类 | | |
| 操作描述 | addYouqinglianjie () | 新增友情链接 |
| modifyYouqinglianjie () | 修改友情链接 |
| deleteYouqinglianjie () | 删除友情链接 |
| getYouqinglianjie() | 获取友情链接 |

友情链接服务层类描述如表6-20所示。

表6-20 友情链接服务层类描述

|  |  |  |
| --- | --- | --- |
| 友情链接服务层类 | | |
| 类的描述：友情链接服务层类 | | |
| 操作描 | addYouqinglianjie () | 新增友情链接 |
| modifyYouqinglianjie () | 修改友情链接 |

续表6-20

|  |  |  |
| --- | --- | --- |
| 述 | deleteYouqinglianjie () | 删除友情链接 |
| getYouqinglianjie() | 获取友情链接 |

友情链接数据库操作类描述如表6-21所示。

表6-21 友情链接数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 友情链接数据库操作类 | | |
| 类的描述：友情链接数据库操作类 | | |
| 操作描述 | addYouqinglianjie () | 新增友情链接 |
| modifyYouqinglianjie () | 修改友情链接 |
| deleteYouqinglianjie () | 删除友情链接 |
| getYouqinglianjie() | 获取友情链接 |

（3）顺序图设计

友情链接顺序图如图6-28所示。

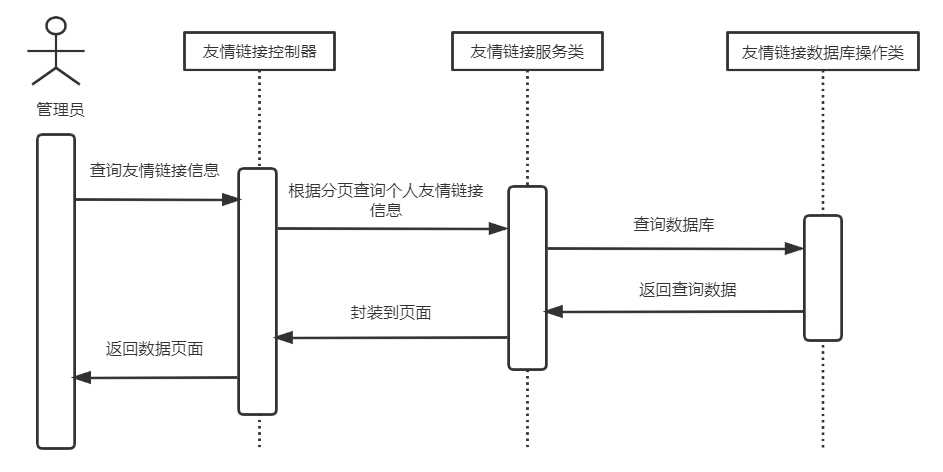


图6-28 友情链接管理顺序图

（4）核心处理流程设计

友情链接管理活动图如图6-29 所示。

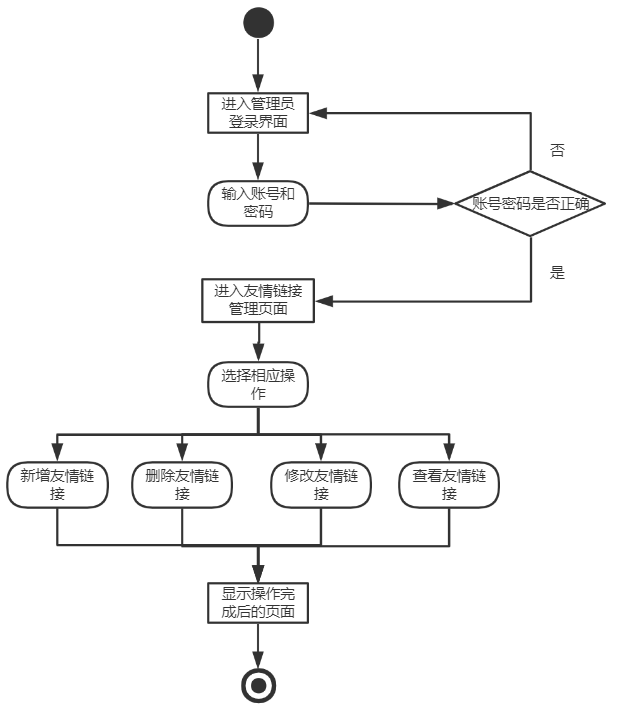


图6-29 友情链接管理活动图

### 6.2.8个人资料管理

（1）功能结构设计

个人资料管理包图如图6-30所示。



图6-30 个人资料管理包图

（2）类图设计

个人资料管理类图如图6-31所示。

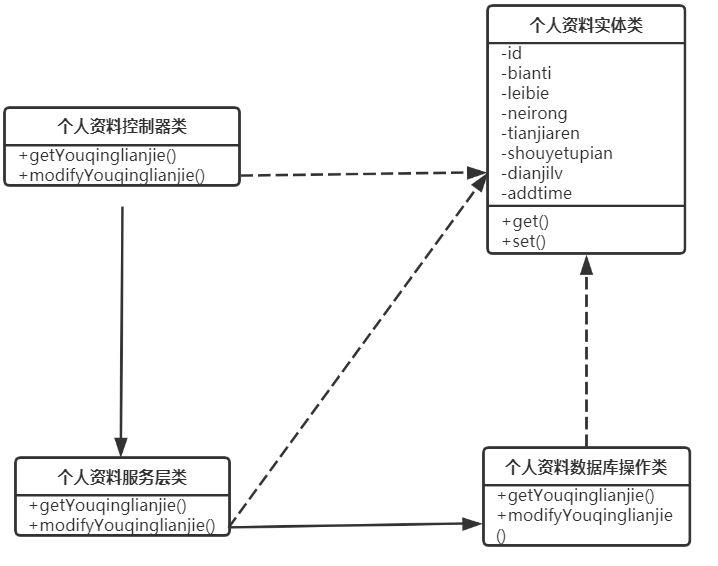


图6-31 个人资料管理类图

在个人资料管理类图中，主要涉及到四个类的关联，个人资料控制器类关联个人资料服务层类，同时关联个人资料实体类。个人资料服务层类关联个人资料数据库操作类，同时关联个人资料实体类。系统个人资料管理涉及到的各个类的描述如下所示，其中个人资料控制器类描述如表6-22所示。

表6-22 个人资料控制器类描述

|  |  |  |
| --- | --- | --- |
| 个人资料控制器类 | | |
| 类的描述：个人资料控制器类 | | |
| 操作描述 | modifyYonghuxinxi() | 修改个人信息 |
| getYonghuxinxi() | 获取个人信息 |

个人资料服务层类描述如表6-23所示。

表6-23 个人资料服务层类描述

|  |  |  |
| --- | --- | --- |
| 个人资料服务层类 | | |
| 类的描述：个人资料服务层类 | | |
| 操作描述 | modifyYonghuxinxi() | 修改个人信息 |
| getYonghuxinxi() | 获取个人信息 |

个人资料数据库操作类描述如表6-24所示。

表6-24 个人资料数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 个人资料数据库操作类 | | |
| 类的描述：个人资料数据库操作类 | | |
| 操作描述 | modifyYonghuxinxi() | 修改个人信息 |
| getYonghuxinxi() | 获取个人信息 |

（3）顺序图设计

个人资料管理顺序图如图6-32所示。

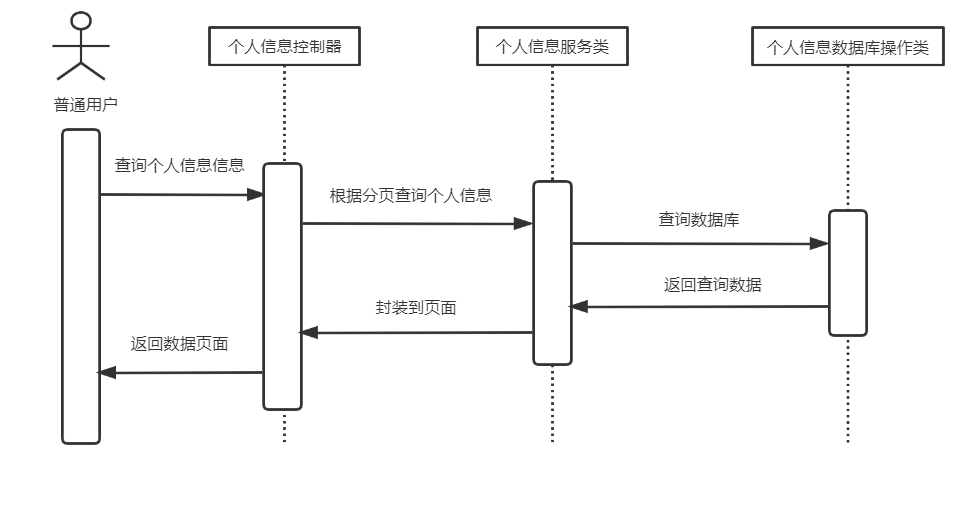


图6-32 个人资料管理顺序图

（4）核心处理流程设计

个人资料管理活动图如图6-33所示。

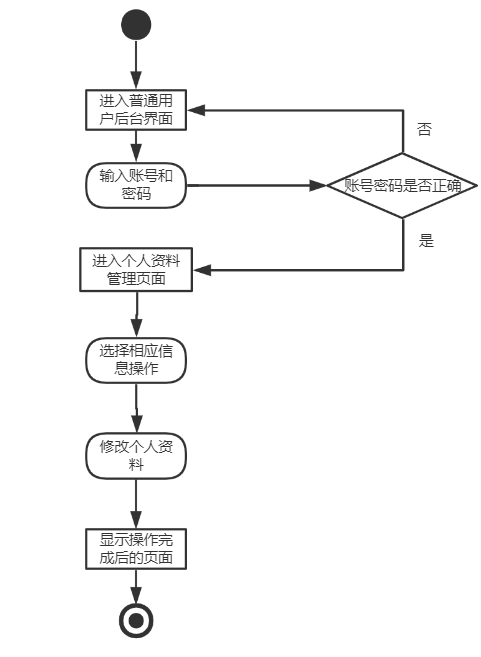


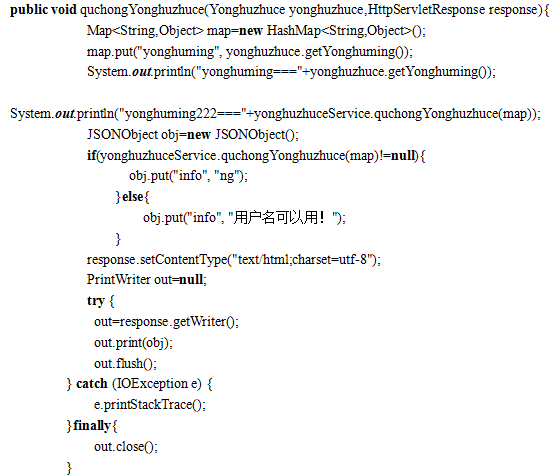
图6-33 个人资料管理活动图

7 编码

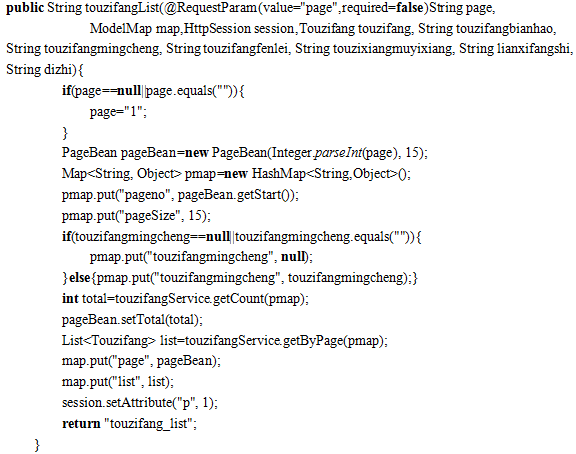
## 7.1 代码实现与核心算法

（1）用户注册登录主要代码，本段代码主要是实现用户注册功能。

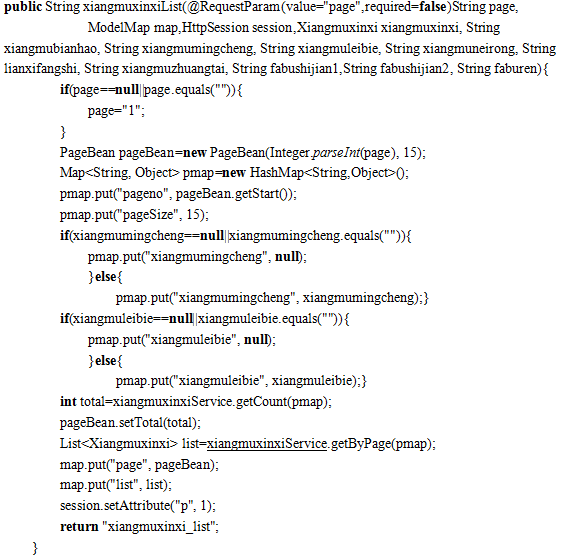




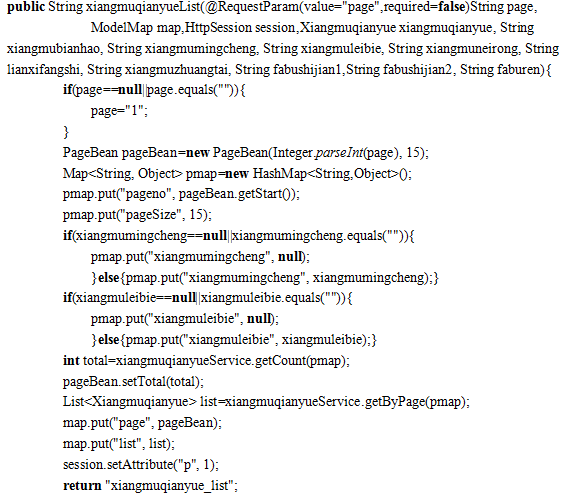
（2）查询投资方核心代码，本段代码实现投资方查询。



（3）查询项目核心代码，本段代码实现项目查询。



（4）查询签约项目核心代码，本段代码实现签约项目查询。



## 7.2 代码优化分析

1. 使用MyBatis框架。MyBatis框架可以用来解除SQL语句和代码之间的耦合，也可以全局管理数据库连接对象，如果不使用MyBatis可能造成数据库连接异常，MyBatis配置如下。



8 测试

## 8.1 测试方案设计

### 8.1.1 测试策略

本系统的测试主要按照单元测试、黑盒测试、白盒测试和系统集成测试四种测试原则进行，其中单元测试为对系统中每个Java类的功能是否正确而设置的测试；黑盒测试主要测试系统对外界的输入是否达到了需求分析中所要求的功能，返回了正确的、人性化的输出给用户；白盒测试则结合每个功能点和相关代码，力求保证代码的健壮性；系统集成测试为项目发布前的最后一轮测试，保证项目的总体功能和性能达到既定标准。

### 8.1.2 测试进度安排

招商引资平台系统的测试进度安排如表8-1所示。

表8-1 测试进度安排表

|  |  |  |
| --- | --- | --- |
| 测试阶段 | 测试任务 | 测试日期及用时 |
| 单元测试 | 对模块接口、局部数据结构、重要执行通路、出错处理通路、边界条件进行白盒测试。 | 第8学期4~7周 |
| 黑盒测试  （前台部分） | 将黑盒测试应用于系统前台所有功能，收集测试结果和问题 | 第8学期第7周 |
| 黑盒测试  （后台部分） | 将黑盒测试应用于系统后台所有功能，收集测试结果和问题 | 第8学期第6周 |
| 白盒测试 | 对系统重要功能点进行白盒测试，确保程序主要执行通路能否按照预定要求正常工作。 | 第8学期第7周 |
| 系统集成测试 | 对系统进行测试和组装，检查是否有与接口相关的问题。 | 第8学期第7周 |

### 8.1.3 测试资源

招商引资平台系统作为一套B/S架构的项目，如果真的可以上线，则可能会有不同用户通过不同的浏览器进行访问，故测试资源中必须包含不同浏览器以测试在不同浏览器环境下是否能够正常运行，测试资源表(硬件环境)如表8-2所示。

表8-2 测试资源表

|  |  |  |
| --- | --- | --- |
| 硬件环境 | | |
| 资源类型 | 资源描述 | 数量 |
| 内存 | 4G DDR3L | 1 |
| 硬盘 | 256G SATA3接口 | 1 |
| 处理器 | Intel I5-4210M | 1 |
| 操作系统 | Windows 10 1903企业版 | 1 |
| JDK版本 | JDK 7.0.800.15 | 1 |

续表8-2

|  |  |  |
| --- | --- | --- |
| 浏览器1 | Chrome 浏览器 | 1 |
| 浏览器2 | FireFox 浏览器 | 1 |
| 浏览器3 | 360安全浏览器 | 1 |
| 浏览器4 | Microsoft Edge 浏览器 | 1 |
| 浏览器5 | IE浏览器 | 1 |

### 8.1.4 关键测试点

要想测试系统的功能是否完备，需要对关键点进行测试，设置了一下测试点进行测试。

1. 管理员是否能够修改、删除项目信息。
2. 普通用户是否能发布项目。
3. 管理员是否能对新闻信息进行查询。
4. 管理员是否可以对产业链进行修改。
5. 管理员是否可以对投资方进行新增。
6. 管理员是否可以对友情链接进行删除。
7. 管理员是否可以对签约项目进行查询。

## 8.2 测试用例构建

### 8.2.1 测试用例编写约定

进行关键点测试的时候，需要对测试用例进行编写约定，根据测试设置了以下约定。

（1）用例编号：测试用例的编号，具有唯一性。

（2）用例名称：测试用例的命名。

（3）前提条件：该测试用例执行前需要满足的条件。

（4）操作步骤：测试用例如何执行。

（5）预期结果：理论上测试用例应该得到的结果。

（6）实际结果：实际上测试用例应该得到的结果。

（7）问题描述：测试用例执行后的错误表现。

（8）BUG编号：出错的编号。

（9）测试结果与结论：通过或者不通过。

（10）测试执行者：测试用例执行的人员。

### 8.2.2 测试用例设计

为保证系统完整性，为关键点设置了严格的测试，对于该系统的主要测试用例结果如以下所示。

（1）管理员修改产业链测试用例如表8-3所示。

表8-3 产业链修改测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL01 |
| 用例名称 | 修改产业链 |

续表8-3

|  |  |
| --- | --- |
| 前提条件 | 管理员成功登录系统 |
| 操作步骤 | 进入产业链管理，选择相应产业链，点击编辑，输入修改内容，点击提交 |
| 预期结果 | 系统显示修改后的页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

（2）管理员新增投资方测试用例如表8-4所示。

表8-4 投资方新增测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL02 |
| 用例名称 | 新增投资方 |
| 前提条件 | 管理员成功登录系统 |
| 操作步骤 | 进入投资方管理，点击添加投资方，输入投资方内容，点击提交 |
| 预期结果 | 系统显示提交后的页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

（3）管理员删除友情链接测试用例如表8-5所示。

表8-5 友情链接删除测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL03 |
| 用例名称 | 删除友情链接 |
| 前提条件 | 管理员成功登录系统 |
| 操作步骤 | 进入友情链接管理，选择相应链接，点击删除 |
| 预期结果 | 系统显示删除后的页面 |

续表8-5

|  |  |
| --- | --- |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

（4）管理员查询签约项目测试用例如表8-6所示。

表8-6 签约项目查询测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL04 |
| 用例名称 | 查询签约项目 |
| 前提条件 | 管理员成功登录系统 |
| 操作步骤 | 进入签约项目管理，输入相应查询条件，点击查询 |
| 预期结果 | 系统显示查询后的页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

（5）管理员查询新闻信息测试用例如表8-7所示。

表8-7 新闻信息查询测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL05 |
| 用例名 | 查询新闻信息 |
| 前提条件 | 管理员成功登录系统 |
| 操作步骤 | 进入新闻信息管理，输入查询内容，点击查询 |
| 预期结果 | 系统显示查询结果页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG编号 | 暂无 |
| 测试结果与结论 | 测试通过 |

续表8-7

|  |  |
| --- | --- |
| 测试执行者 |  |

### 8.2.3 关键测试用例

为保证系统中关键点功能成功实现，设置了以下关键点测试用例。

（1）管理员删除项目测试用例如表8-8所示。

表8-8 删除项目测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL06 |
| 用例名称 | 删除项目 |
| 前提条件 | 管理员成功登录系统 |
| 操作步骤 | 选择相应项目，点击删除按钮 |
| 预期结果 | 系统显示删除后页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

（2）普通用户发布项目测试用例如表8-9所示。

表8-9 发布项目测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL07 |
| 用例名称 | 发布项目 |
| 前提条件 | 普通用户成功登录系统 |
| 操作步骤 | 选择项目管理，点击添加项目，填写项目信息，点击提交按钮 |
| 预期结果 | 系统显示提交后页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 8.2.4 测试用例维护

1. 及时添加新的测试用例。在系统开发周期过程中，一些需求可能会被添加进来，而已有的测试用例可能覆盖不到新的需求，此时需要增加测试用例来覆盖到新的需求中。
2. 及时删除多余的测试用例。在测试用例设计时，由于考虑不周可能导致多组测试用例经过相同的执行通路，此时需要剔除多余的测试用例保持一组即可。
3. 改进测试用例。在系统开发周期过程中，可能在初期一组测试用例覆盖多条测试路径，但由于后期系统改进导致此组测试用例覆盖度不如之前，但也覆盖到多数，这是需要改进测试用例。
4. 及时删除无效的测试用例。在系统开发周期过程中，可能由于功能的删除而导致此组测试用例无效，这时就需要删除这组无效的测试用例。

9 总结与展望

## 9.1 设计工作总结

通过这次毕业设计，我学到了很多，有的东西只有自己亲自动手才能更深刻的理解和记忆，也只有实践才能把课堂上学到的知识真正转化为自己的东西。

这次毕业设计，从需求分析开始到代码编写，再到论文编写，我花了很多时间去查资料，也是通过这次毕业设计，我知道了程序编写的严谨，知道了在每一个上线程序的背后都需要相关工作人员付出多少的努力。在需求分析中，不仅需要调研了解当代用户的需求，而且需要从该程序的安全性、可行性、可扩展性等多个方面进行深入的分析。只有这样才能让程序的后续开发更加的完整。所以一个好的程序员，不仅只了解编程技术，还应该懂得分析需求，分析程序，在大方向更全面地看待这个程序，才能做出最终让客户满意度程序。

在这次完成毕业设计的过程中，由于经验不足，从没有一个人完成一个这么完整的项目，所以遇到了很多困难，我就只能在完成毕业设计的同时去请教老师以及上网查找相关资料和解决办法。虽然程序编写的进度比较慢，但是可以让自己学的更多，理解也更深刻，比单纯的理论知识也更加有意思。

由于是第一次独立完成项目，能力有限，项目完成的不够完善，对于需求方面考虑的也不够全面，所以我今后会更加努力的学习，争取能够自己做成更加完善的程序。

## 9.2 未来工作展望

由于目前技术能力不足，需求考虑得不够全面，导致系统还有很多不够完善的地方。页面设计的不够美观，导致用户视觉效果不好，没有设计评论功能，不能让用户实现互动沟通，有的功能模块不够完整，数据统计应该可以设计的更加细节一点，能让更加具体的显示所需要的数据信息。希望以后通过不断地学习，能够不断地提升自己，对这些缺陷进行改进，既让系统更加完善，也可以让自己的技术得到提高。

谢 辞

参考文献

[1] 张海籓，牟永敏编 .软件工程导论（第6版）[M]. 北京：清华大学出版社，2018  
[2] 王珊, 萨师煊编. 数据库系统概论[M]. 第5版. 北京: 高等教育出版社, 2017.

[3]包洁琼. 招商引资项目档案管理与信息资源共享分析[J]. 海峡科技与产业, 2019(1).

[4]贺伟,李凤. 基于项目驱动式教学的《Java面向对象程序设计》课程实践[J]. 计算机产品与流通,2019.

[5]王越. JAVA编程语言在计算机软件开发中的应用[J]. 电子技术与软件工程,2019.

[6]明日科技.Java从入门到精通.第五版.清华大学出版社.2019

[7]倪海顺. 计算机软件开发的Java编程语言应用探讨[J]. 信息与电脑(理论版),2019

[8]熊亚蒙。 HTML5在移动互联网中的应用研究[J]. 无线互联科技，2016

[9]赵春刚, 才智, 高志军, et al. UML使用基础教程[M]. 北京大学出版社, 2013.

[10]薛均晓 李占波. UML系统分析与设计[M]. 机械工业出版社, 2014.

[11]白伟华。基于协同的软件工程专业实践教学改革研究[J].科教文汇（中旬刊），2017,（08）：47-48.  
[12].电子技术与软件工程[J].电子技术与软件工程，2017,（20）：266.  
[13]卜晓波。软件工程金融信息化[J].电子技术与软件工程，2017,（20）：45.  
[14]王依楠。数据库基于计算机软件工程的编程技术[J].电子技术与软件工程，2017,（20）：165.  
[15]沈海波，周如旗，朱雄泳。工程教育认证背景下的软件工程专业建设和实践[J].软件，2017,38（09）：87-90.  
[16]张博，郝杰。项目驱动与微课混合教学模式用于软件工程实践的教学研究[J].卫生职业教育，2017,35（21）：73-74.  
[17]路璐，凌捷。软件工程专业五位一体培养体系的探索与实践[J].电脑知识与技术，2017,13（28）：180-181+183.  
[18]侯欢欢，张麟华。软件工程类课程实践教学研究[J].软件导刊（教育技术），2017,16（10）：45-46.  
[19]潘正高，陈黎黎，李鸿，房爱东，于子甲，徐旭。地方应用型高校软件工程专业课程体系的研究[J].西昌学院学报（自然科学版），2017,31（03）：94-97.  
[20]邓奉先，卓书尧。基于CIPP模式的高职《软件工程》项目化教学评价的研究[J].电脑知识与技术，2017,13（27）：158-159.

[21]Hollinger J P , Peirce J L , Poe G A . SSM/I instrument evaluation[J]. IEEE Transactions on Geoscience & Remote Sensing, 1990, 28(5):781-790.

[22]Ryu J , Kamata S I . SSM-HPC: Front View Gait Recognition Using Spherical Space Model with Human Point Clouds[C]// 18th IEEE International Conference on Image Processing, ICIP 2011, Brussels, Belgium, September 11-14, 2011. IEEE, 2011.

[23]Anatoly Akkerman, Alexander Totok, Vijay Karamcheti. Infrastructure for Automatic Dynamic Deployment of J2EE Applications in Distributed Environments[C]// Component Deployment, Third International Working Conference, Cd, Grenoble, France, November. Springer-Verlag, 2005.

[24] J T Nugroho,Zylshal,G A Chulafak,D Kushardono. Performance of LAPAN-A2 satellite data to classify land cover/land use in Semarang, Central Java[J]. IOP Conference Series: Earth and Environmental Science,2017,54.

[25] E Yulihastin,Trismidianto,H Satyawardhana,G A Nugroho. MJO modulation on diurnal rainfall over West Java during pre-monsoon and strong El Ni?o periods[J]. IOP Conference Series: Earth and Environmental Science,2017,54.

[26]Avtonomov Dmitry M,Raskind Alexander,Nesvizhskii Alexey I. BatMass: a Java Software Platform for LC-MS Data Visualization in Proteomics and Metabolomics.[J]. Journal of proteome research,2016,15.  
[27]Fitzastri Alrassi,Emil Salim,Anastasia Nina,Luthfi Alwi,Projo Danoedoro,Muhammad Kamal. GEOBIA For Land Use Mapping Using Worldview2 Image In Bengkak Village Coastal, Banyuwangi Regency, East Java[J]. IOP Conference Series: Earth and Environmental Science,2016,47.  
[28]S. W. Nurdian,N. Adu,I. R. Palupi,W. Raharjo. Comparison tomography relocation hypocenter grid search and guided grid search method in Java island[J]. Journal of Physics: Conference Series,2016,776.  
[29]Achmad Darul,Dasapta Erwin Irawan,Nurjanna Joko Trilaksono,Aditya Pratama,Ulfi Rizki Fitria. Conceptual model of groundwater and river water interactions in Cikapundung riverbank, Bandung, West Java[J]. IOP Conference Series: Earth and Environmental Science,2016,29.  
[30]Suharsono,S. W Nurdian,I. R. Palupi. Earthquake effect on volcano and the geological structure in central java using tomography travel time method and relocation hypocenter by grid search method[J]. Journal of Physics: Conference Series,2016,776.

附录A 外文翻译—原文部分

**Original source:** Baidu Scholar

**Infrastructure for Automatic Dynamic Deployment**

**Of J2EE Application in Distributed Environments**

**Abstract:** in order to achieve such dynamic adaptation, we need an infrastructure for automating J2EE application deployment in such an environment. This need is quite evident to anyone who has ever tried deploying a J2EE application even on a single application server, which is a task that involves a great deal of configuration of both the system services and application components.

**Key words:** J2EE; component; Distributed; Dynamic Deployment;

**1 Introduction**

In recent years, we have seen a significant growth in component-based enterprise application development. These applications are typically deployed on company Intranets or on the Internet and are characterized by high transaction volume, large numbers of users and wide area access. Traditionally they are deployed in a central location, using server clustering with load balancing (horizontal partitioning) to sustain user load. However, horizontal partitioning has been shown very efficient only in reducing application-related overheads of user-perceived response times, without having much effect on network-induced latencies. Vertical partitioning (e.g., running web tier and business tier in separate VMs) has been used for fault isolation and load balancing but it is sometimes impractical due to significant run-time overheads (even if one would keep the tiers on a fast local-area network) related to heavy use of remote invocations. Recent work [14] in the context of J2EE component based applications has shown viability of vertical partitioning in wide-area networks without incurring the aforementioned overheads. The key conclusions from that study can be summarized as follows:

• Using properly designed applications, vertical distribution across wide-area networks improves user-perceived latencies.

• Wide-area vertical layering requires replication of application components and maintaining consistency between replicas.

• Additional replicas may be deployed dynamically to handle new requests.

• Different replicas may, in fact, be different implementations of the same component based on usage (read-only, read-write).

• New request paths may reuse components from previously deployed paths.

Applying intelligent monitoring [6] and AI planning [2, 12] techniques in conjunction with the conclusions of that study, we see a potential for dynamic adaptation in industry-standard J2EE component-based applications in wide area networks

Through deployment of additional application components dynamically based on active monitoring. However, in order to achieve such dynamic adaptation, we need an infrastructure for automating J2EE application deployment in such an environment. This need is quite evident to anyone who has ever tried deploying a J2EE application even on a single application server, which is a task that involves a great deal of configuration of both the system services and application components. For example one has to set up JDBC data sources, messaging destinations and other resource adapters before application components can be configured and deployed. In a wide area deployment that spans multiple server nodes, this proves even more complex, since more system services that facilitate inter-node communications need to be configured and started and a variety of configuration data, like IP addresses, port numbers, JNDI names and others have to be consistently maintained in various configuration files on multiple nodes.

This distributed deployment infrastructure must be able to:

• address inter-component connectivity specification and define its effects on component configuration and deployment,

• address application component dependencies on application server services, their configuration and deployment,

• provide simple but expressive abstractions to control adaptation through dynamic deployment and undeployment of components,

• enable reuse of services and components to maintain efficient use of network nodes’ resources,

• provide these facilities without incurring significant additional design effort on behalf of application programmers.

In this paper we propose the infrastructure for automatic dynamic deployment of J2EE applications, which addresses all of the aforementioned issues. The infrastructure defines architecture description languages (ADL) for component and link description and assembly. The Component Description Language is used to describe application components and links. It provides clear separation of application components from system components. A flexible type system is used to define compatibility of component ports and links. A declaration and expression language for configurable component properties allows for specification of inter-component dependencies and propagation of properties between components. The Component (Replica) Assembly Language allows for assembly of replicas of previously defined components into application paths by

Connecting appropriate ports via link replicas and specifying the mapping of these component replicas onto target application server nodes. The Component Configuration Process evaluates an application path’s correctness, identifies the dependencies

of application components on system components, and configures component replicas for deployment. An attempt is made to match and reuse any previously deployed replicas in the new path based on their configurations. We implement the infrastructure as a part of the JBoss open source Java application server [11] and test it on several

Sample J2EE applications – Java Pets tore [23], Rubies [20] and TPC-W-NYU [32]. The infrastructure implementation utilizes the JBoss’s extendable *micro-kernel* architecture, based on the JMX [27] specification. Componentized architecture of JBoss allows incremental service deployments depending on the needs of deployed applications. We believe that dynamic reconfiguration of application servers through dynamic deployment and undeployment of system services is essential to building a resource-efficient framework for dynamic distributed deployment of J2EE applications. The rest of the paper is organized as follows. Section 2 provides necessary background for understanding the specifics of the J2EE component technology which are relevant to this study. Section 3 gives a general description of the infrastructure architecture, while section 4 goes deeper in describing particularly important and interesting internal mechanisms of the infrastructure. Section 5 describes the implementation of the framework, and related work is discussed in section 6.

**2 J2EE Background**

**2.1 Introduction**

**Component frameworks.** A component framework is a middleware system that supports applications consisting of components conforming to certain standards. Application components are “plugged” into the component framework, which establishes their environmental conditions and regulates the interactions between them. This is usually done through containers, component holders, which also provide commonly required support for naming, security, transactions, and persistence. Component frameworks provide an integrated environment for component execution, as a result significantly reduce the effort .it takes to design, implement, deploy, and maintain applications. Current day industry component framework standards are represented by Object Management Group’s CORBA Component Model [18], Sun Microsystems’ Java 2 Platform Enterprise Edition (J2EE) [25] and Microsoft’s .NET [17], with J2EE being currently the most popular and widely used component framework in the enterprise arena.

**J2EE.** Java 2 Platform Enterprise Edition (J2EE) [25] is a comprehensive standard for developing multi-tier enterprise Java applications. The J2EE specification among other things defines the following:

• Component programming model,

• Component contracts with the hosting server,

• Services that the platform provides to these components,

• Various human roles,

• Compatibility test suites and compliance testing procedures.

Among the list of services that a compliant application server must provide are messaging, transactions, naming and others that can be used by the application components. Application developed using J2EE adhere to the classical 3-Tier architectures – Presentation Tier, Business Tier, and Enterprise Information System (EIS) Tier (see Fig. 1). J2EE components belonging to each tier are developed adhering to the

Specific J2EE standards.

1. **Presentation or Web tier.**

This tier is actually subdivided into client and server sides. The client side hosts a web browser, applets and Java applications that communicate with the server side of presentation tier or the business tier. The server side hosts Java Servlet components [30], Java Server Pages (JSPs) [29] and static web content. These components are responsible for presenting business data to the end users. The data itself is typically acquired from the business tier and sometimes directly from the Enterprise Information System tier. The server side of the presentation tier is typically accessed through HTTP(S) protocol.

2. **Business or EJB tier.**

This tier consists of Enterprise Java Beans (EJBs) [24] that model the business logic of the enterprise application. These components provide persistence mechanisms and transactional support. The components in the EJB tier are invoked through remote invocations (RMI), in-JVM invocations or asynchronous message delivery, depending on the type of EJB component. The EJB specification defines several types of components. They differ in invocation style (synchronous vs. asynchronous, local vs. remote) and statefulness: completely stateless (e.g., Message-Driven Bean), stateful non-persistent

(e.g., Stateful Session Bean), stateful persistent (e.g., Entity Bean). Synchronously invocable EJB components expose themselves through a special factory proxy object (an EJB Home object, which is specific to a given EJB), which is typically bound in JNDI by the deployer of the EJB. The EJB Home object allows creation or location of an EJB

Object, which is a proxy to a particular instance of an EJB 1.

**3. Enterprise Information System (EIS) or Data tier.**

This tier refers to the enterprise information systems, like relational databases, ERP systems, messaging systems and the like. Business and presentation tier component communicate with this tier with the help of resource adapters as defined by the Java Connector Architecture [26].The J2EE programming model has been conceived as a distributed programming model where application components would run in J2EE servers and communicate with each other. After the initial introduction and first server implementations, the technology, most notably, the EJB technology has seen some a significant shift away from purely distributed computing model towards local interactions 2. There were very legitimate performance-related reasons behind this shift, however the

Distributed features are still available. The J2EE specification has seen several revisions, the latest stable being version 1.3, while version 1.4 is going through last review phases 3. We shall focus our attention on the former, while actually learning from the latter. Compliant commercial J2EE implementations are widely available from BEA Systems [4], IBM [9], Oracle [21] and other vendors. Several open source implementations, including JBoss [11] and JOnAS [19] claim compatibility as well. A

Recent addition to the list is a new Apache project Geronimo [1].

**2.2 J2EE Component Programming Model**

Before we describe basic J2EE components, let’s first address the issue of defining what a component is a software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties [31].According to this definition the following entities which make up a typical J2EE application would be considered application components (some exceptions given below):

• EJBs (session, entity, message-driven),

• Web components (servlets, JSPs),

• messaging destinations,

• Data sources,

EJB and Web components are deployed into their corresponding containers provided by the application server vendor. They have well-defined contracts with their containers that govern lifecycle, threading, persistence and other concerns. Both Web and EJB components use JNDI lookups to locate resources or other EJB components they want to communicate with. The JNDI context in which these lookups are performed is maintained separately for each component by its container. Bindings messaging destinations, such as topics and queues, are resources provided by a messaging service implementation. Data sources are resources provided by the application server for data access by business components into the enterprise information services (data) tier, and most commonly are exemplified by JDBC connection pools managed by the application

Server. A J2EE programmer explicitly programs only EJBs and Web components. These custom-written components interact with each other and system services both implicitly and explicitly. For example, an EJB developer may choose explicit transaction demarcation (i.e., Bean-Managed Transactions) which means that the developer assumes the burden of writing explicit programmatic interaction with the platform’s *Transaction Manager* Service through well-defined interfaces. Alternatively, the developer may choose Container-Managed transaction demarcation, where transactional behavior of a component is defined through its descriptors and handled completely by the EJB container, thus acting as an implicit dependency of the EJB on the underlying *Transaction Manager* service.

**2.3 Links Between Components**

**2.3.1 Remote Interactions**

J2EE defines only three basic inter-component connection types that can cross application server boundaries, in all three cases; communication is accomplished through special Java objects.

*•* Remote EJB invocation: synchronous EJB invocations through EJB Home and EJB Object interfaces.

• Java Connector outbound connection: synchronous message receipt, synchronous and asynchronous message sending,

Database query using Connection Factory and Connection interfaces.

• Java Connector inbound connection: asynchronous message delivery into Message-Driven Beans (MDBs) only, utilizing Activation Spec objects. In the first two cases, an application component developer writes the code that performs lookup of these objects in the component’s run-time JNDI context as well as code that issues method invocations or sends and receives messages to and from the remote component. The component’s run-time JNDI context is created for each deployment of the component.

Bindings in the context are initialized at component deployment time by the deployed (usually by means of component’s deployment descriptors). These bindings are assumed to be static, since the specification does not provide any contract between the container and the component to inform of any binding changes In the case of Java Connector inbound communication, Activation Spec object lookup and all subsequent interactions with it are done implicitly by the MDB container. The protocol for lookup has not been standardized, though it is reasonable to assume a JMX- or JNDI-based lookup assuming the underlying application server provides facilities to control each step of deployment process, establishment of a link between J2EE components would involve:

• Deployment of target component classes (optional for some components, like destinations),

• Creation of a special Java object to be used as a target component’s proxy,

• Binding of this object with component’s host naming service (JNDI or JMX),

• Start of the target component,

• Deployment of referencing component classes,

• Creation and population of referencing component’s run-time context in its host naming service,

• start of the referencing component.

However, none of modern application servers allow detailed control of the deployment process for all component types beyond what is possible by limited options in their deployment descriptors 4. Therefore our infrastructure will use a simplified approach that relies on features currently available on most application servers:

• Ability to deploy messaging destinations and data sources dynamically,

• Ability to create and bind into JNDI special objects to access messaging destinations and data sources,

• Ability to specify initial binding of EJB Home objects upon EJB component deployment,

• Ability to specify a JNDI reference 5 in the referencing component’s run-time context to point to the EJB Home binding of the referenced EJB component. In our infrastructure which is limited to homogeneous application servers, these options are sufficient to control intercomponent links through simple deployment descriptor manipulation. However, in context of heterogeneous application servers, simple JNDI references and thus simple descriptor manipulation are insufficient due to cross-application-server

Classloading issues.

**2.3.2 Local Interactions**

Some interactions between components can occur only between components co-located in the same application server JVM and sometimes only in the same container. In the Web tier, examples of such interactions are servlet-to-servlet request forwarding. In the EJB tier, such interactions are CMP Entity relations and invocations via EJB local interfaces. Such local deployment concerns need not be exposed at the level of a distributed deployment infrastructure other than to ensure collocation. Therefore, the infrastructure treats all components requiring collocation as a single component.

**2.4 Deployment of J2EE Applications and System Services**

**2.4.1 Deployment of Application Components**

Deployment and undeployment of standard J2EE components has not yet been standardized (see JSR 88 [10] for standardization effort 6). Therefore, each application server vendor provides proprietary facilities for component deployment and undeployment. And while the J2EE specification does define packaging of standard components which includes format and location of XML-based deployment descriptors within the package, this package is not required to be deployable by an application server without proprietary transformation. Examples of such transformation are

• Generation of additional proprietary descriptors that supplement or replace the standard ones,

• Code generation of application server-specific classes.

In order to proceed with building a dynamic distributed deployment infrastructure capable of deploying in heterogeneous networks, we propose a universal unit of deployment to be a single XML-based deployment descriptor or a set of such,

Bundled into an archive. The archive may optionally include Java classes that implement the component and any other resources that the component may need. Alternatively, the deployment descriptors may simply have URL references to codebases. We assume presence of a dynamic deployment/undeployment service on all compliant J2EE servers and a robust application server classloading architecture capable of repeated deployment cycles without undesired classloading-related issues. Most modern application servers (e.g., JBoss [11] and Geronimo [1]) do provide such facilities.

**2.4.2 Deployment of System Components (Services)**

While lacking only in the area of defining a clear specification of deployment and undeployment when it comes to application components, the J2EE standard falls much shorter with respect to system services. Not only a standardized deployment facility for system services is not specified, the specification, in fact, places no requirements even on life cycle properties of these services, nor does it address the issue of explicit specification of application component dependencies on the underlying system services. Instead it defines a role of human deploy who is responsible for ensuring that the required services are running based on his/her understanding of dependencies of application components on system services as implied by the nature of components and their deployment descriptors.

附录B 外文翻译—译文部分

原文出处：百度学术

基于J2EE在分布式环境下的底层结构的自动动态配置的应用

摘要：为了实现广域网中符合工业标准基于组件的应用程序中动态的可适应性，我们需要一种框架来在这样的环境里自动化地配置J2EE 应用程序。这种需要对于哪怕在单一的应用程序服务器上尝试部署J2EE应用的人来说也显而易见，这种任务设计到大量的系统服务和应用组件的配置。

关键词：J2EE；动态配置；分布式；组件；

1 前言

近几年，我们已经看到基于组件的企业应用开发的显著增加。这种应用程序通常被部署在公司的内部网或者是因特网上，以高事务容量，大量的用户和覆盖范围广的访问为特征，它通常会被部署在中央区域，采用服务器集群来均衡负载从而支持用户下载。但是这种平均负荷的方法被证明只对减少应用转发的用户可以察觉的反应时间有效，而对于减少网络方面的延迟作用不大，垂直分割（例如运行网络层和事务层在不同的虚拟机）被用于错误分离和均衡负荷，但是它是不符合实际的由于远程调运的大量使用显著地增加了运行时间。最近的著作已经表明在广域网中利用垂直负荷而不引起前面所述的超时问题的可行性。那非研究的主要结论可以概括如下：

(1) 应用合适的应用程序，在广域网中的垂直负荷可以察觉的延迟。

(2) 广域垂直层需要复制应用层组件而且需要维持和原组件间的一致性。

(3) 新加的复制组件可以被动态配置以满足新的需要。

(4) 事实上，不同的复制组件可能会根据应用不同的方式实现相组件。

(5) 新的请求路径可以复用先前的组件配置路径。

应用智能监视和人工智能规划方法再结合那个研究得出的结论，我们看到通过动态布置基于动态监视的额外的应用组件，在广域网中符合工业标准基于组件的应用程序中动态的可适应性是可以实现的。然而，为了实现这种动态可适性，我们需要一种框架来在这样的环境里自动化地配置J2EE 应用程序。这种需要对于哪怕在单一的应用程序服务器上尝试布置J2EE应用的人来说也显而易见，这种任务设计到大量的系统服务和应用组件的配置。例如你必须在配置和部署应用组件前先建立JDBC数据源，设立消息目的地和资源适配器。在需要跨越多个节点服务器的广域网配置中，这将更加复杂，因为更多的便利内部节点通信的系统服务需要配置和启动，而且多种配置数据比如IP地址，端口号，JNDI名字和其他的数据在多个节点的配置文件中必须维持一致性。这种分布式配置框架必须满足：

(1) 声明内部组件一致性规范和定义它对组件配置部署的影响。

(2) 声明应用程序组件对应用服务器，以及它们的配置和部署的依赖性。

(3) 提供简单但可表达的抽象方法去控制通过部署和拆卸组件获得的适用性。

(4) 能够复用服务和组件从而高效的利用网路节点资源。

(5) 提供上述便利而不会增加应用程序员的设计负担。

在本论文中，我们提出自动动态部署J2EE应用程序的框架涉及了上面的所有问题，这种框架为组件定义了结构描述语言，链接说明和集合。这种组件说明语言用来描述应用程序组件和链接，它使得应用组件与系统组件中清晰的分开。一种灵活的系统类型用来定义组件接口和端口的兼容性。一种为配置组件属性而开发的定义和表述语言允许内部组件间独立的规范和组件间属性的继承。组件集合语言允许先前定义的复制的组件通过连接合适的端口集合到应用路径，连接时通过链接复制对象和具体把这些复制组件映射到目标应用服务器节点。组件配置过程评估了应用程序路径的正确性，确认在系统组件上的应用组件的独立性和完成复制组件的部署。根据这些配置使先前部署的复制组件在新的路径中得以匹配和复用的努力正在做出。我们把这种架构作为JBoss开源java应用服务器的一部分加以实现，在几个J2EE样本程序比如Java PetStore,，RUB和 TPC\_W\_NYU中进行测试。这种架构实现利用了JBoss的可扩展的微内核结构，基于JMX规范。JBoss的组件结构允许根据部署应用程序的需要增加服务配置。我们相信通过动态部署和拆卸系统服务来重构应用服务器对构建高效资源框架的动态分布部署的J2EE应用程序来说是非常必要的。本文如下部分是这样组织的。第2部分提供了必要的背景以理解和研究有关的J2EE组件技术规范。第3部分对这种架构给出了一般性的描述。第4部分更深入的描述了有关这种架构特别重要的和有趣的内部机制。第五部分描述了如何实现这种架构，相关联的工作将在第六部分介绍。

**2 J2EE背景知识**

2.1 介绍

组件框架。组件框架是一种中间件系统，它支持遵守一定标准的有不同组件构成的应用程序。应用组件被塞入这种确立它们运行环境和规定它们交互的框架中。这通常是通过容器，组件持有者来实现的。这种容器也提供通常需要的功能以实现命名，安全性，事务，和持久性。组件框架为组件的执行提供了一个集成的环境，因此显著的减少了在设计、实现、部署和维护应用程序时工作。现在工业上的组件框架标准以对象管理组的CORBA组件模型，SUN 公司的JAVA 2 Platform J企业版[J2EE]和微软公司的.NET标准，其中在企业里应用最为广泛的组件框架是J2EE。J2EE是开发多层企业应用JAVA程序的综合性的标准。J2EE规范定义如下：

(1) 组件编程模型。

(2) 组件和主服务器的链接。

(3) 服务器提供给组件的服务。

(4) 各种各样的人物角色。

(5) 兼容性检验装置和编译测试程序。

在众多的服务列表中，消息通信，事务处理，命名机制和其它应用组件用到的服务是应用服务器必须提供的。用J2EE进行应用开发必须遵守经典的3层结构——表现层、业务层和企业信息系统层。属于各层的J2EE组件在开发时遵守具体的J2EE标准。

1、表现层或者网络层

这一层实际上又被分为客户端和服务器端。客户端包括浏览器，applets，Java应用程序等和负责和服务器端的表现层或者业务层进行交互。服务器端包括servlet、jsp和静态网页内容。这些组件负责把业务数据传递给终端用户。数据本身通常从业务层获得有时也从企业信息系统层直接获得。表现层的服务器端通常通过HTTP协议来进行访问。

2、业务层或者EJB层

这一层包含EJB，即企业应用的事务逻辑模型。这些组件提供了持久化机制和事务支持。EJB中的组件通过RMI被调用。在Java虚拟机调用或者异步的消息传递，取决与EJB组件的类型。EJB规范定义了很多种组件。它们在调用风格（同步和异步，本地和远程）与状态（完全状态，不可持久状态，可持久）方面不同。同步调用的EJB组件通过特定的工厂代理对象来表现自己。这种工厂代理对象通常被EJB部署者绑定在JNDI中。EJB对象允许或者本地EJB对象是特定EJB实例的代理。

3、企业信息系统或者数据层

这一层指的就是企业信息系统，比如关系数据库，ERP系统，消息系统等。业务层和持久层在资源适配器的帮助下与该层进行通信。资源适配器在Java连结结构中被定义。J2EE编程模型一直被认为是分布式的编程模型，在该模型中应用组件在J2EE服务器上运行并且彼此可以相互交互。经过初始化说明和第一个服务实现后，该技术，更显著的说EJB技术，已经明显地从纯粹的分布式计算模型转向了本地交互。转变的背后有合理的性能有关的原因，然而分布式的特征现在还存在。J2EE规范已经经过了好几次修订，现在最稳定的版本是1.3，1.4版本正处于重审阶段。我们应该把注意力放在1.3版本上，而实际上是在学习后者。适用与商业的J2EE实现可以大量的从BEA系统，IBM，Oracle等赞助商得到。包括JBoss和JOnAS在内的开源实现据称兼容性也不错。最近名单上有多出了新的Apache project Geronimo。

2.2 J2EE组件编程模型

在我们基本的J2EE组件前，先让我们强调一下什么是组件。软件组件是有一系列的具体的接口和明确的上下文环境构成。它可以被独立的部署而且易于被第三方重构。根据以上的定义，如下的组成J2EE应用程序的实体可以看作是软件组件：

(1) EJBS（会话，实体，消息驱动）。

(2) Web组件（Servlet、JSP）。.

(3) 消息目的。

(4) 数据源。

EJB和Web组件被部署在由应用服务赞助商提供的容器中.它们有定义良好的容器规则来管理生命周期，线程，持久化和其他问题。EJB和Web组件都利用JNDI目录机制去寻找资源和它们想要交互的其EJB组件。目录被执行的JNDI环境被独立的由容器的每个组件加以维护。该种环境下的绑定机制通常由组件部署的解释者加以配置。消息目的地，像对话和队列，是由消息服务执行所提供的资源。数据源是提供给应用服务器的为事务组件进入到企业信息服务层提供数据接口，通常由被应用服务器管理的JDBC连接池实例化。一个J2EE编程者明确编写的项目只有EJB和Web组件。这些用户编写的组件彼此交互而且系统服务可以是明显的也可以是隐含的。例如，EJB开发者可以选择明确的事务区分方式，这种方式意味着开发者假设通过定义良好接口的事务经理服务平台来书写明确的程序交互。或者，开发者也可选择容器管理事务区分的方式。这样由于组件的事务行为通过他的描述者来定义而且全部用EJB容器来处理，因此作为一个隐式独立的EJB提供潜在的事务管理服务。

2.3 组件间的链接

2.3.1 远程交互

J2EE仅定义了三种可以在不同应用服务器间传递的基本组件间连接类型。在这三种情况下，通信通过特定的Java对象来完成。

(1) 远程EJB调用：同步的EJB调用通过主EJB对象和EJB对象接口来实现。

(2) Java连结器的外部连接：同步消息接收，同步和异步消息发送，用连接工厂和连接接口进行数据库查询。

(3) Java连接器的内部连接：异步消息传递进入消息驱动Bean只能使用Activation Spec 对象。

在前两个实例中，应用组件的开发者不仅书写执行在组件的运行时JNDI环境中的对象目录代码，而且书写发布方法调用，与远程的组件相互发送和接受消息。组件的运行时JNDI环境为每一个组件部署所创建。环境中的绑定在组件部署时由部署者进行初始化。这些绑定被假设为是静态的，因为规格中没有提供任何的容器和组件间协议去提示绑定发生了变化。在 Java连接器的内部通信情景下，Activation Spec 对象查询以及所有的相应的M容器隐式的完成。虽然查询的协议还没有被标准化，但是假设一个基于JMX或者JNDI的查询是合理的。 假设潜在的应用服务器提供了所有的设备去控制部署过程的每一步，那么在两个J2EE组件间确立一个连接需要涉及：

(1) 部署目标组件类。

(2) 创建一个特定的Java对象用作目标组件代理。

(3) 用组件的命名服务去绑定目标。

(4) 启动目标组件。

(5) 部署指定的组件类。

(6) 在主机的命名服务中，创建和进行指定组件的运行环境。

(7) 启动指定的组件。

然而，没有一个现代的应用服务器允许详细的控制所有组件类型的部署过程除了在它们的部署解释器中的有限的选择。因此我们的架构将使用简化的途径，它所依赖的特征在现在的大多数的应用服务器上都可以得到。

(1) 动态部署消息目的和数据源的能力。

(2) 创建和绑定特定的JNDI目标去访问消息目的和数据源的能力。

(3) 把初始绑定的EJB对象到EJB部署组件的能力。

(4) 用在参考组件运行环境中的JNDI指引去指出绑定的参考EJB的能力。

在只有相同的应用服务器的架构中，上面的功能对通过简单的部署控制解释器方式来控件间的连接已经足够了。然而，在不同应用服务器的环境下，由于跨服务器的类下载问题，这种简单的控制解释器的方式是不够的。

2.3.2 本地交互

一些组件间的交互可以发生在同一地点的相同应用服务器虚拟机的组件间，有时候甚至可以发生在只有相同容器的组件间。在Web层，这种交互的例子是 servlet到 servlet的请求转发。在EJB层，这种交互的例子是CMP实体关系和通过EJB本地接口的调用。这种本地部署所关心的不是在分布式架构中去表现而是去增强一致性。因此，这种架构把所有的本地的组件请求当作一个单一的组件加以对待。

2.4 部署J2EE应用程序和系统服务

2.4.1 部署应用程序组件

部署和拆卸标准的J2EE组件还没有统一的标准，因此每个应用服务的提供商对组件的部署和拆卸提供了单独的功能于J2EE规范中没有定义标准组件的包，包的格式和包内的基于xml部署解释器的位置，因此这种包对于没有所属权变化的应用服务器不需要部署。具体变化的例子有：

(1) 支持或者取代标准所有者解释器的新的所有者解释器的产生。

(2) 具体服务应用程序类的代码的更替。

为了着手构建一个能够部署不可网络的动态的分布式的架构，我们提出了一种普遍的部署单元即一个简单的基于xml部署的解释器或者是一组类似的绑定到文档中的解释器。文档可能包含用于执行组件的Java类或者任何其它的所需组件。相应地，部署解释器也可以简单地用URL来索引代码。我们假设这种动态的部署和拆卸服务存在于所有的兼容的J2EE服务器上而且在不理解类重载相关问题时一个健壮的类重载结结构的应用服务器就能够重复的部署生命周期。大多数现代的应用服务器都提供这样的功能。

2.4.2 部署系统组件

对应用组件来说，J2EE规范只是少了在部署和拆卸时的明确定义，而对系统服务来说，在这方面做的更糟。对系统服务来说不仅没有具体的定义一个标准化的部署，实际上，这个规格甚至连没有强调在生命周期属性方面的要求，更不用手强调依赖也潜在的系统服务的应用组件的明确规范了取而代之的是它定义了部署者的角色，这个角色负责确保像组件的本性和系统的解释器所暗示的那样，所需的服务是基于应用组件对系统服务依赖性的基之上。

附录C 软件使用说明书

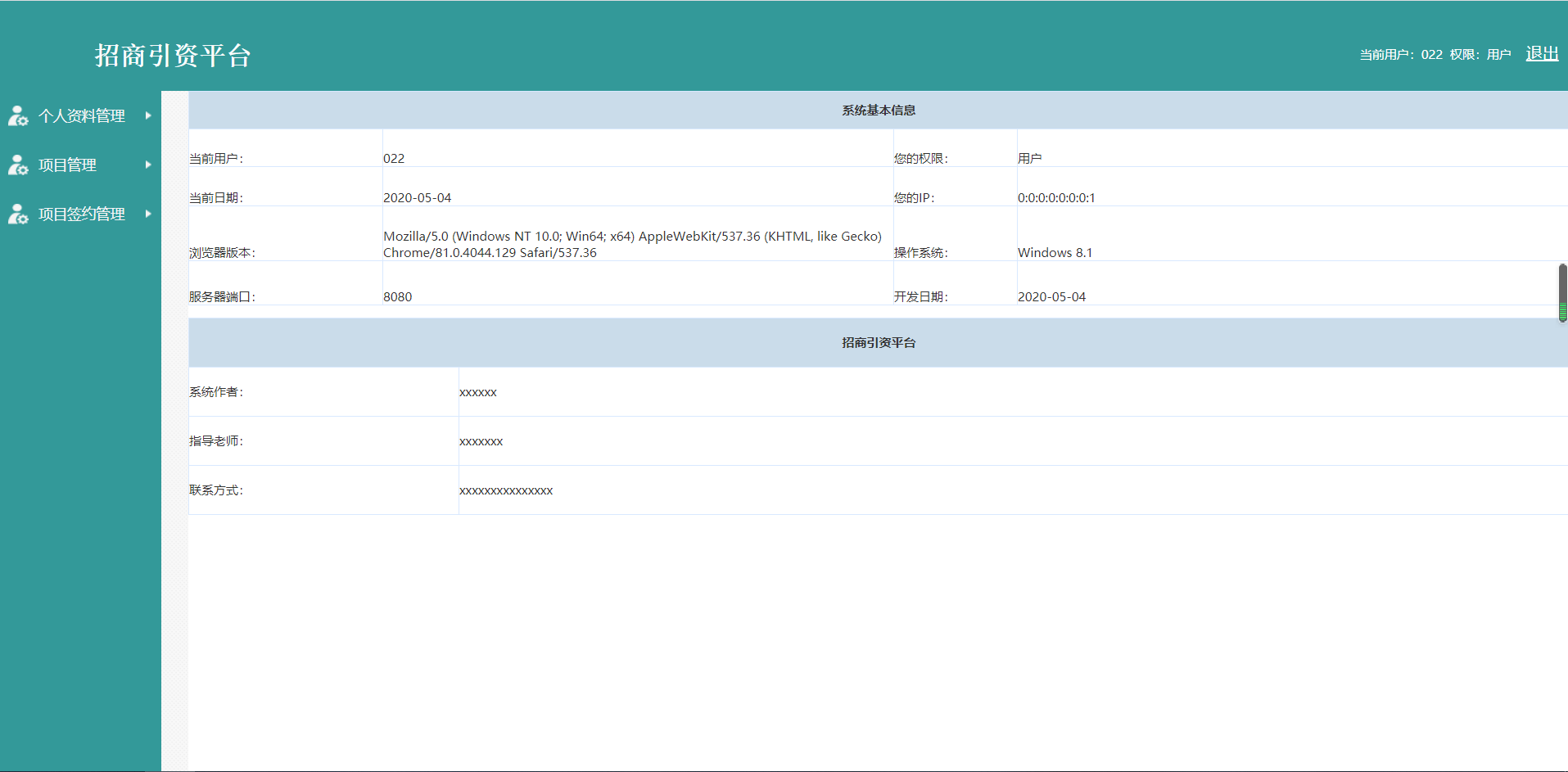
本系统运行即进入网站主界面。



管理员登录后进入到管理员后台页面。



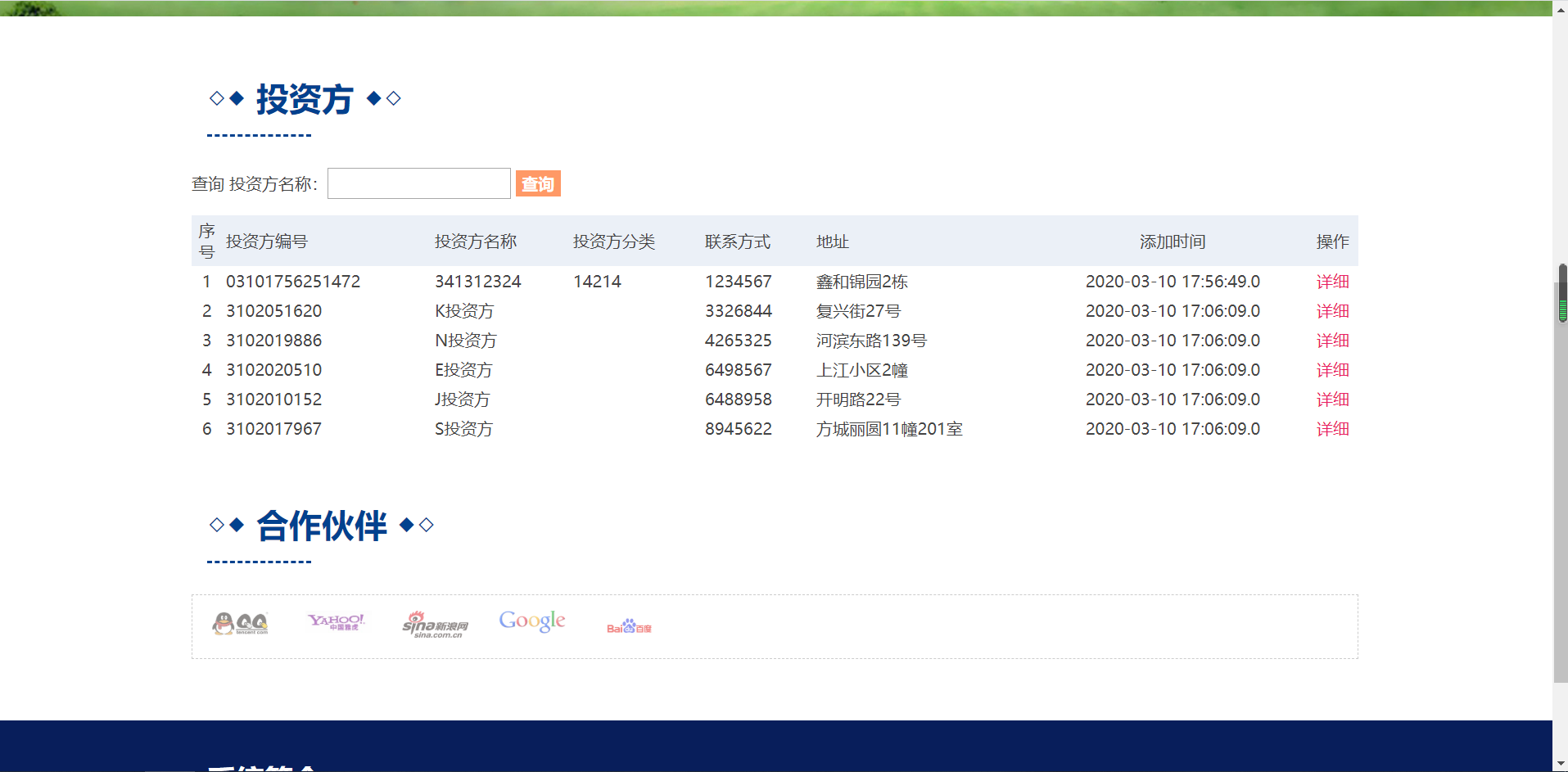
普通用户登录后进入到后台页面。



进入系统后点击项目信息进入到项目信息页面。



进入系统后点击投资方进入到投资方信息页面。



进入系统后点击产业链进入到产业链信息页面。



附录D 主要源代码

<!-- 连接数据库 -->

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver"></property>

<property name="url" value="jdbc:mysql://localhost:3306/ssmzsyzpthsg7591W1B4?useUnicode=true&amp;characterEncoding=utf-8"></property>

<property name="username" value="root"></property>

<property name="password" value=""></property>

</bean>

<!-- <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">

<property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"></property>

<property name="url" value="jdbc:sqlserver://localhost:1433;databaseName=ssmzsyzpthsg7591W1B4"></property>

<property name="username" value="sa"></property>

<property name="password" value="sa123456"></property>

</bean>-->

<!-- spring管理mybatis中的sqlSessionFactory -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"></property>

<property name="configLocation" value="classpath:mybatis-config.xml"></property>

</bean>

<!-- dao接口所在的位置 -->

<bean id="config" class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="com.dao"></property>

<property name="sqlSessionFactory" ref="sqlSessionFactory"></property>

</bean>

<!-- 配置事务类 -->

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource"></property>

</bean>

<!-- 配置事务的通用属性 -->

<tx:advice id="txAdvice" transaction-manager="transactionManager">

<tx:attributes>

<tx:method name="add\*" propagation="REQUIRED"/>

<tx:method name="update\*" propagation="REQUIRED"/>

<tx:method name="delete\*" propagation="REQUIRED"/>

<tx:method name="\*" propagation="REQUIRED" read-only="true"></tx:method>

</tx:attributes>

</tx:advice>

<!-- 配置切点 -->

<aop:config>

<aop:pointcut expression="execution(\* com.server.impl.\*.\*(..))" id="mycut"/>

<aop:advisor advice-ref="txAdvice" pointcut-ref="mycut"/>

</aop:config>

<!-- 支持上传文件 -->

<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver"/>

<!-- 自动扫描包中的注解，依赖注入 -->

<context:component-scan base-package="com"/>

<!-- 开启注解功能 -->

<mvc:annotation-driven/>

</beans>

@Controller

public class YonghuxinxiController {

@Resource

private YonghuxinxiServer yonghuxinxiService;

@RequestMapping("addYonghuxinxi.do")

public String addYonghuxinxi(HttpServletRequest request,Yonghuxinxi yonghuxinxi,HttpSession session) throws SQLException{

Timestamp time=new Timestamp(System.currentTimeMillis());

yonghuxinxi.setAddtime(time.toString().substring(0, 19));

yonghuxinxiService.add(yonghuxinxi);

session.setAttribute("backxx", "添加成功");

session.setAttribute("backurl", request.getHeader("Referer"));

//session.setAttribute("backurl", "yonghuxinxiList.do");

return "redirect:postback.jsp";

//return "redirect:yonghuxinxiList.do";

}

// 处理编辑

@RequestMapping("doUpdateYonghuxinxi.do")

public String doUpdateYonghuxinxi(int id,ModelMap map,Yonghuxinxi yonghuxinxi){

yonghuxinxi=yonghuxinxiService.getById(id);

map.put("yonghuxinxi", yonghuxinxi);

return "yonghuxinxi\_updt";

}

@RequestMapping("doUpdateYonghuxinxi2.do")

public String doUpdateYonghuxinxi2(ModelMap map,Yonghuxinxi yonghuxinxi,HttpServletRequest request){

yonghuxinxi=yonghuxinxiService.getById(Integer.parseInt((String)request.getSession().getAttribute("uid")));

map.put("yonghuxinxi", yonghuxinxi);

return "yonghuxinxi\_updt2";

}

@RequestMapping("updateYonghuxinxi2.do")

public String updateYonghuxinxi2(int id,ModelMap map,Yonghuxinxi yonghuxinxi){

yonghuxinxiService.update(yonghuxinxi);

return "redirect:doUpdateYonghuxinxi2.do";

}

// 后台详细

@RequestMapping("yonghuxinxiDetail.do")

public String yonghuxinxiDetail(int id,ModelMap map,Yonghuxinxi yonghuxinxi){

yonghuxinxi=yonghuxinxiService.getById(id);

map.put("yonghuxinxi", yonghuxinxi);

return "yonghuxinxi\_detail";

}

// 前台详细

@RequestMapping("yhxxDetail.do")

public String yhxxDetail(int id,ModelMap map,Yonghuxinxi yonghuxinxi){

yonghuxinxi=yonghuxinxiService.getById(id);

map.put("yonghuxinxi", yonghuxinxi);

return "yonghuxinxidetail";

}

//

@RequestMapping("updateYonghuxinxi.do")

public String updateYonghuxinxi(int id,ModelMap map,Yonghuxinxi yonghuxinxi,HttpServletRequest request,HttpSession session){

yonghuxinxiService.update(yonghuxinxi);

session.setAttribute("backxx", "修改成功");

session.setAttribute("backurl", request.getHeader("Referer"));

return "redirect:postback.jsp";

//String url = request.getHeader("Referer");

//return "redirect:"+url;

//return "redirect:yonghuxinxiList.do";

}

// 分页查询

@RequestMapping("yonghuxinxiList.do")

public String yonghuxinxiList(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Yonghuxinxi yonghuxinxi, String zhanghao, String mima, String xingming, String xingbie, String shoujihao, String shenfenzhenghao, String zhiye, String dizhi, String beizhu, String issh){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(zhanghao==null||zhanghao.equals("")){

pmap.put("zhanghao", null);

}else{pmap.put("zhanghao", zhanghao);}

if(xingming==null||xingming.equals("")){

pmap.put("xingming", null);

}else{pmap.put("xingming", xingming);}

int total=yonghuxinxiService.getCount(pmap);

pageBean.setTotal(total);

List<Yonghuxinxi> list=yonghuxinxiService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "yonghuxinxi\_list";

}

@RequestMapping("yhxxList.do")

public String yhxxList(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Yonghuxinxi yonghuxinxi, String zhanghao, String mima, String xingming, String xingbie, String shoujihao, String shenfenzhenghao, String zhiye, String dizhi, String beizhu, String issh){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(zhanghao==null||zhanghao.equals("")){

pmap.put("zhanghao", null);

}else{pmap.put("zhanghao", zhanghao);}

if(xingming==null||xingming.equals("")){

pmap.put("xingming", null);

}else{pmap.put("xingming", xingming);}

int total=yonghuxinxiService.getCount(pmap);

pageBean.setTotal(total);

List<Yonghuxinxi> list=yonghuxinxiService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "yonghuxinxilist";

}

@RequestMapping("yhxxListtp.do")

public String yhxxListtp(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Yonghuxinxi yonghuxinxi, String zhanghao, String mima, String xingming, String xingbie, String shoujihao, String shenfenzhenghao, String zhiye, String dizhi, String beizhu, String issh){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(zhanghao==null||zhanghao.equals("")){

pmap.put("zhanghao", null);

}else{pmap.put("zhanghao", zhanghao);}

if(xingming==null||xingming.equals("")){

pmap.put("xingming", null);

}else{pmap.put("xingming", xingming);}

int total=yonghuxinxiService.getCount(pmap);

pageBean.setTotal(total);

List<Yonghuxinxi> list=yonghuxinxiService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "yonghuxinxilisttp";

}

@RequestMapping("deleteYonghuxinxi.do")

public String deleteYonghuxinxi(int id,HttpServletRequest request){

yonghuxinxiService.delete(id);

String url = request.getHeader("Referer");

return "redirect:"+url;

//return "redirect:yonghuxinxiList.do";

}

@RequestMapping("quchongYonghuxinxi.do")

public void quchongYonghuxinxi(Yonghuxinxi yonghuxinxi,HttpServletResponse response){

Map<String,Object> map=new HashMap<String,Object>();

map.put("zhanghao", yonghuxinxi.getZhanghao());

System.out.println("zhanghao==="+yonghuxinxi.getZhanghao());

System.out.println("zhanghao222==="+yonghuxinxiService.quchongYonghuxinxi(map));

JSONObject obj=new JSONObject();

if(yonghuxinxiService.quchongYonghuxinxi(map)!=null){

obj.put("info", "ng");

}else{

obj.put("info", "账号可以用！");

}

response.setContentType("text/html;charset=utf-8");

PrintWriter out=null;

try {

out=response.getWriter();

out.print(obj);

out.flush();

} catch (IOException e) {

e.printStackTrace();

}finally{

out.close();

}

}

}

@Controller

public class TouzifangController {

@Resource

private TouzifangServer touzifangService;

@RequestMapping("addTouzifang.do")

public String addTouzifang(HttpServletRequest request,Touzifang touzifang,HttpSession session) throws SQLException{

Timestamp time=new Timestamp(System.currentTimeMillis());

touzifang.setAddtime(time.toString().substring(0, 19));

touzifangService.add(touzifang);

session.setAttribute("backxx", "添加成功");

session.setAttribute("backurl", request.getHeader("Referer"));

//session.setAttribute("backurl", "touzifangList.do");

return "redirect:postback.jsp";

//return "redirect:touzifangList.do";

}

// 处理编辑

@RequestMapping("doUpdateTouzifang.do")

public String doUpdateTouzifang(int id,ModelMap map,Touzifang touzifang){

touzifang=touzifangService.getById(id);

map.put("touzifang", touzifang);

return "touzifang\_updt";

}

// 后台详细

@RequestMapping("touzifangDetail.do")

public String touzifangDetail(int id,ModelMap map,Touzifang touzifang){

touzifang=touzifangService.getById(id);

map.put("touzifang", touzifang);

return "touzifang\_detail";

}

// 前台详细

@RequestMapping("tzfDetail.do")

public String tzfDetail(int id,ModelMap map,Touzifang touzifang){

touzifang=touzifangService.getById(id);

map.put("touzifang", touzifang);

return "touzifangdetail";

}

//

@RequestMapping("updateTouzifang.do")

public String updateTouzifang(int id,ModelMap map,Touzifang touzifang,HttpServletRequest request,HttpSession session){

touzifangService.update(touzifang);

session.setAttribute("backxx", "修改成功");

session.setAttribute("backurl", request.getHeader("Referer"));

return "redirect:postback.jsp";

//String url = request.getHeader("Referer");

//return "redirect:"+url;

//return "redirect:touzifangList.do";

}

// 分页查询

@RequestMapping("touzifangList.do")

public String touzifangList(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Touzifang touzifang, String touzifangbianhao, String touzifangmingcheng, String touzifangfenlei, String touzixiangmuyixiang, String lianxifangshi, String dizhi){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(touzifangmingcheng==null||touzifangmingcheng.equals("")){

pmap.put("touzifangmingcheng", null);

}else{pmap.put("touzifangmingcheng", touzifangmingcheng);}

int total=touzifangService.getCount(pmap);

pageBean.setTotal(total);

List<Touzifang> list=touzifangService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "touzifang\_list";

}

@RequestMapping("tzfList.do")

public String tzfList(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Touzifang touzifang, String touzifangbianhao, String touzifangmingcheng, String touzifangfenlei, String touzixiangmuyixiang, String lianxifangshi, String dizhi){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(touzifangmingcheng==null||touzifangmingcheng.equals("")){

pmap.put("touzifangmingcheng", null);

}else{pmap.put("touzifangmingcheng", touzifangmingcheng);}

int total=touzifangService.getCount(pmap);

pageBean.setTotal(total);

List<Touzifang> list=touzifangService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "touzifanglist";

}

@RequestMapping("tzfListtp.do")

public String tzfListtp(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Touzifang touzifang, String touzifangbianhao, String touzifangmingcheng, String touzifangfenlei, String touzixiangmuyixiang, String lianxifangshi, String dizhi){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(touzifangmingcheng==null||touzifangmingcheng.equals("")){

pmap.put("touzifangmingcheng", null);

}else{pmap.put("touzifangmingcheng", touzifangmingcheng);}

int total=touzifangService.getCount(pmap);

pageBean.setTotal(total);

List<Touzifang> list=touzifangService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "touzifanglisttp";

}

@RequestMapping("deleteTouzifang.do")

public String deleteTouzifang(int id,HttpServletRequest request){

touzifangService.delete(id);

String url = request.getHeader("Referer");

return "redirect:"+url;

//return "redirect:touzifangList.do";

}

}@Controller

public class ChanyelianxinxiController {

@Resource

private ChanyelianxinxiServer chanyelianxinxiService;

@RequestMapping("addChanyelianxinxi.do")

public String addChanyelianxinxi(HttpServletRequest request,Chanyelianxinxi chanyelianxinxi,HttpSession session) throws SQLException{

Timestamp time=new Timestamp(System.currentTimeMillis());

chanyelianxinxi.setAddtime(time.toString().substring(0, 19));

chanyelianxinxiService.add(chanyelianxinxi);

session.setAttribute("backxx", "添加成功");

session.setAttribute("backurl", request.getHeader("Referer"));

//session.setAttribute("backurl", "chanyelianxinxiList.do");

return "redirect:postback.jsp";

//return "redirect:chanyelianxinxiList.do";

}

// 处理编辑

@RequestMapping("doUpdateChanyelianxinxi.do")

public String doUpdateChanyelianxinxi(int id,ModelMap map,Chanyelianxinxi chanyelianxinxi){

chanyelianxinxi=chanyelianxinxiService.getById(id);

map.put("chanyelianxinxi", chanyelianxinxi);

return "chanyelianxinxi\_updt";

}

// 后台详细

@RequestMapping("chanyelianxinxiDetail.do")

public String chanyelianxinxiDetail(int id,ModelMap map,Chanyelianxinxi chanyelianxinxi){

chanyelianxinxi=chanyelianxinxiService.getById(id);

map.put("chanyelianxinxi", chanyelianxinxi);

return "chanyelianxinxi\_detail";

}

// 前台详细

@RequestMapping("cylxxDetail.do")

public String cylxxDetail(int id,ModelMap map,Chanyelianxinxi chanyelianxinxi){

chanyelianxinxi=chanyelianxinxiService.getById(id);

map.put("chanyelianxinxi", chanyelianxinxi);

return "chanyelianxinxidetail";

}

//

@RequestMapping("updateChanyelianxinxi.do")

public String updateChanyelianxinxi(int id,ModelMap map,Chanyelianxinxi chanyelianxinxi,HttpServletRequest request,HttpSession session){

chanyelianxinxiService.update(chanyelianxinxi);

session.setAttribute("backxx", "修改成功");

session.setAttribute("backurl", request.getHeader("Referer"));

return "redirect:postback.jsp";

//String url = request.getHeader("Referer");

//return "redirect:"+url;

//return "redirect:chanyelianxinxiList.do";

}

// 分页查询

@RequestMapping("chanyelianxinxiList.do")

public String chanyelianxinxiList(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Chanyelianxinxi chanyelianxinxi, String bianhao, String xiangmuleibie, String shangyou, String zhongyou, String xiayou, String beizhu){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(xiangmuleibie==null||xiangmuleibie.equals("")){

pmap.put("xiangmuleibie", null);

}else{pmap.put("xiangmuleibie", xiangmuleibie);}

int total=chanyelianxinxiService.getCount(pmap);

pageBean.setTotal(total);

List<Chanyelianxinxi> list=chanyelianxinxiService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "chanyelianxinxi\_list";

}

@RequestMapping("cylxxList.do")

public String cylxxList(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Chanyelianxinxi chanyelianxinxi, String bianhao, String xiangmuleibie, String shangyou, String zhongyou, String xiayou, String beizhu){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(xiangmuleibie==null||xiangmuleibie.equals("")){

pmap.put("xiangmuleibie", null);

}else{pmap.put("xiangmuleibie", xiangmuleibie);}

int total=chanyelianxinxiService.getCount(pmap);

pageBean.setTotal(total);

List<Chanyelianxinxi> list=chanyelianxinxiService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "chanyelianxinxilist";

}

@RequestMapping("cylxxListtp.do")

public String cylxxListtp(@RequestParam(value="page",required=false)String page,

ModelMap map,HttpSession session,Chanyelianxinxi chanyelianxinxi, String bianhao, String xiangmuleibie, String shangyou, String zhongyou, String xiayou, String beizhu){

if(page==null||page.equals("")){

page="1";

}

PageBean pageBean=new PageBean(Integer.parseInt(page), 15);

Map<String, Object> pmap=new HashMap<String,Object>();

pmap.put("pageno", pageBean.getStart());

pmap.put("pageSize", 15);

if(xiangmuleibie==null||xiangmuleibie.equals("")){

pmap.put("xiangmuleibie", null);

}else{pmap.put("xiangmuleibie", xiangmuleibie);}

int total=chanyelianxinxiService.getCount(pmap);

pageBean.setTotal(total);

List<Chanyelianxinxi> list=chanyelianxinxiService.getByPage(pmap);

map.put("page", pageBean);

map.put("list", list);

session.setAttribute("p", 1);

return "chanyelianxinxilisttp";

}

@RequestMapping("deleteChanyelianxinxi.do")

public String deleteChanyelianxinxi(int id,HttpServletRequest request){

chanyelianxinxiService.delete(id);

String url = request.getHeader("Referer");

return "redirect:"+url;

//return "redirect:chanyelianxinxiList.do";

}

}