# 目 录

1 绪 论 1

* 1. [研究的背景及意义 1](#_bookmark0)
     1. [选题的背景 1](#_bookmark1)
     2. 国内外研究现状 1
     3. 研究的意义 2

1.2 系统目标 2

2 需求分析 4

2.1 业务需求 4

2.2 功能需求 6

2.2.1 角色分析 6

2.2.2 业务需求 7

2.3 非功能需求 14

2.3.1 环境需求 14

2.3.2 性能需求 15

2.3.3 安全需求 15

3 总体设计 17

* 1. 系统设计的原则 17
  2. 系统体系结构设计 17
  3. 系统功能结构设计 20

4 数据库设计 22

* 1. 概念结构设计 22

4.1.1 设计思路 22

4.1.2 E-R 图 22

* 1. 逻辑结构设计 27

4.2.1 设计思路 27

4.2.2 逻辑模型 27

* 1. 物理结构设计 28

4.3.1 存取方式 28

4.3.2 存储结构 28

5 界面设计 33

* 1. 界面关系图或工作流图 33
  2. 界面设计成果 36

5.2.1 主界面 36

5.2.2 子界面 36

6 详细设计 43

* 1. 系统主要功能模块介绍 43
  2. 活动管理模块设计 44
     1. 功能结构设计 44

6.2.2 类图设计 44

6.2.3 顺序图设计 58

6.2.4 核心处理流程设计 60

* 1. 成果库管理模块设计 61
     1. 功能结构设计 61

6.3.2 类图设计 62

6.3.3 顺序图设计 65

6.3.4 核心处理流程设计 65

* 1. 标签维度管理模块设计 66
     1. 功能结构设计 66

6.4.2 类图设计 66

6.4.3 顺序图设计 69

6.4.4 核心处理流程设计 69

* 1. 专家资料管理模块设计 70
     1. 功能结构设计 70

6.5.2 类图设计 70

6.5.3 顺序图设计 74

6.5.4 核心处理流程设计 75

* 1. 专家库管理模块设计 76
     1. 功能结构设计 76

6.6.2 类图设计 76

6.6.3 顺序图设计 79

6.6.4 核心处理流程设计 80

* 1. 工作联络管理模块设计 80
     1. 功能结构设计 80

6.7.2 类图设计 81

6.7.3 顺序图设计 84

6.7.4 核心处理流程设计 85

7 编码 86

* 1. 代码实现与核心算法 86
     1. 日志管理模块 86
     2. 活动管理模块 86
     3. 专家资料管理模块 87
     4. 工作联络管理模块 88
  2. 代码优化分析 89

8 测试 92

* 1. 测试方案设计 92

8.1.1 测试策略 92

8.1.2 测试进度安排 92

8.1.3 测试资源 92

[8.1.4 关键测试点 92](#_bookmark2)

* 1. 测试用例构建 93
     1. 测试用例编写约定 93
     2. 测试用例设计 93
     3. 关键测试用例 93
     4. 测试用例维护 97

9 总结与展望 98

* 1. 设计工作总结 98
  2. 未来工作展望 98

谢 辞 100

参考文献 101

附录 A 外文翻译—原文部分 103

附录 B 外文翻译—译文部分 107

附录 C 软件使用说明书 110

附录 D 主要源代码 112

# 1 绪 论

### 研究的背景及意义

* + 1. 选题的背景

进入新时代，国内外环境形势发生了深刻变化，中国社会主义现代化建设的任务艰巨而艰巨。这凸显了建立具有中国特色的新型智库的更高的新要求。省教育与经济和社会发展智库在 2015 年于南昌正式启动。为了更好地管理智库工作的会议记录，智库工作的专家成果，智库专家的信息，并允许工作人员与专家和领导联系以咨询专家和会议，确实有必要开发一个智库管理信息系统：省教育智库和经济社会发展智库系统，以此来满足工作的需要。根据省委的要求，智库要明确定位，立足本土，聚焦现实，提出更具针对性和可操作性的研究成果。还需要创新思想和方法，加强基础学科问题的学科合作和技术支持，以探索有用的学习和研究技能，提高产学研向融合。需要组建一流的团队，创造优越的条件来支持和帮助智库专家进行研究，并不断提高智库的凝聚力。需要建立一种报告成果的机制，以快速促进智库工作的研究成果并促进成果的应用。智库应基于教育和经济社会发展的战略现实要求。主要研究课题是公共教育政策，综合教育改革，教育规则的建设，教育与经济社会发展的关系以及社会结构矛盾主要以以研究，专家咨询，政策建议，人才培训和项目介绍的形式开展研究和咨询活动。智库需要关注省教育和经济社会发展改革的迫切需要，加强协作与创新，不断推出新的深入的研究成果，努力组建新型智库。要反思教育与经济社会发展过程中的不足，对症下药，反馈跟踪，使之成为在全国具有重要影响的新型智库，做出自己的特色。深化教育领域的综合改革，使之办成群众满意的教育，为的发展和现代化作出必要的贡献。

* + 1. 国内外研究现状

现代智库经历了三个发展阶段，它具有实践研究，种类繁多，专业功能齐全，综合多样的国际化先进视野等特点。 需要利用总结好世界智库发展的经验，并融合中国现实，需要明确定位，改进和发展具有中国特色的智库。 在当今社会中，智库对政府决策，企业发展和社会舆论都会产生重大影响。 在一些西方媒体看来，智库是继政府， 法律和新闻之后的“第四部门”，它们构成了该国“软实力”的重要组成部分。 进入改革与发展时代的中国，由于国内外发展环境复杂多变，改革发展面临艰巨挑战，有望发挥智库作用。

中国当前正处于伟大复兴的关键时期，任何政策的出台都需要深思熟虑，都需要满足人民群众利益和期待，为此，智库需要充分介入政策制定的前端、中端和后端的过程中来，要以人民群众乐于接受的方式，来对社会公众解读国家政策和社会事件， 让智库真正成为国家决策与社会舆论之间的媒介。

为此，当前中国需要智库需要广泛参与国家事务并且建言建策，积极宣讲中国的核心价值观和经济社会发展的成功经验，要通过与国外的智库的广泛深入的交流合作， 致力于推动全球层面对挑战，在中国社会发展战略与发展道路的深进行广泛而深入探讨。中国智库还需要积极培养符合治理的国际化人才，因为人才竞争是国际竞争的核心焦点，是制定新的国际规则的核心，也是中国成为世界强国的重要基础条件。对于政府充分发挥职能作用，管理好智库提出了更高的要求。而智库管理系统对于这一发展期望起到了不可替代的作用，一个实用于省经济社会发展智库的管理系统应用而生。

* + 1. 研究的意义

为顺应省委、省政府对教育工作的期望，对经济社会发展的需要，推动教育领域改革、提高教育质量、实现教育强省、助推绿色崛起等方面发挥着重要作用。为高效、有序的进行教育与经济社会发展智库专家管理、会议活动管理及智库成果管理，拟开发智库管理信息系统进行支撑。该系统采用 B/S 架构，支持 J2EE 框架，采用数据库，与省教育厅统一门户融合。该系统功能设计包括有专家库、会议管理、活动管理、工作联络管理、成果库等功能模块。应充分利用好信息化手段等手段，建设好教育与经济社会发展智库管理信息系统，系统实现会议与活动管理、工作联络管理、专家资料管理、专家库管理、专家维度管理、成果库管理等，为省厅领导及时了解会议活动成果、专家库的活动记录、及献言献策内容的有效的管理， 为领导科学决策提供基础数据支撑。

### 系统目标

总体目标：

本系统基于 B/S 架构，采用 J2EE 技术进行开发，利用 MVC 的思想，实现数据的数据访问层、业务层、控制层的分离，满足模块化开发的要求，建设教育与经济社会发展智库管理信息系统，系统实现会议与活动管理、工作联络管理、专家资料管理、专家库管理、专家维度管理、成果库管理等，为省厅领导及时了解会议活动成果、专家库的活动记录、及献言献策内容的有效的管理，为领导科学决策提供基础数据支撑。

具体目标：

本次项目建设的具体目标包括建设“领导端、秘书端、联络员端和管理员端”。领导端是指领导登录智库系统后，可以查询到专家会议活动情况，会议的相关资

料、专家参会考勤情况、各个专家的详细信息及专家发表的成果等信息。

联络员端是指系统中设置联络员联系跟踪专家，提供专家需要的工作辅助。

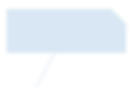
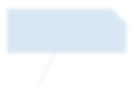
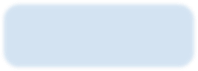
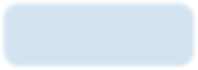
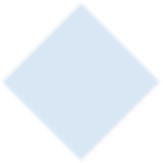
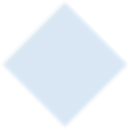
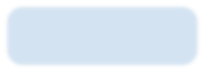
秘书端是秘书登录智库系统后，对智库工作所需要召开的活动开展服务工作，例如，会议纪要、专家意见、会议资料的上传与下载，活动的考勤等信息进行维护。

管理员端是指维护系统运行所需的数据各种功能，具体包括：用户管理、角色管理、日志管理、标签维度维护和专家联系配置，管理员拥有本系统的所有权限。

# 需求分析

### 业务需求

1. 能够实现专家资料展示及修改，专家成果展示及修改，专家标签维度管理，工作联络管理，专家活动管理等功能，对专家信息及活动的全过程进行管理。
2. 实现对专家活动过程中产生的所有业务数据的管理，如会议记录、资料记录、专家意见、会议考勤等管理与维护。
3. 具有与专家信息展示相配套的系统后台管理维护功能，能够对专家资料信息、专家成果信息、用户操作的系统日志等信息进行管理与维护，并能够进行对应的权限管理。
4. 有着高效，界面美观及响应速度合理的优点，安全性较高。
5. 系统运行稳定，对用户操作的过程记录满足对错误操作进行复原。
6. 系统运行必须安全可靠，特别是用户密码等信息须加密处理，防止信息泄露。
   * 1. 主要业务流程
7. 用户登录活动图，如图 2-1 所示：



登录页面

提交

否

用户名及密码存入

cookie

是

获取用户信息

数据库验证

是否保存用户名及密码

用户信息压入

session

更新登录时间或记

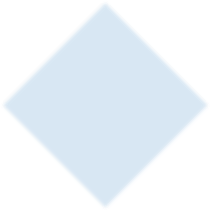
录登录日志

告知用户成功登录

并跳转到首页

图 2-1 用户登录活动图

1. 用户角色新增活动图，如图 2-2 所示：



系统管理员登录

系统管理

用户管理

新增角色

关闭

新增角色成功

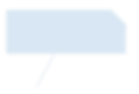
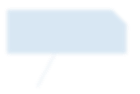
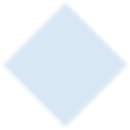
保存

填写角色信息

保存或关闭

图 2-2 用户角色新增活动图

1. 新增专家库并调入专家信息活动图，如图 2-3 所示：



管理员角色或领导

角色进入首页

专家库管理

新增

关闭

详情

生成专家库信息

保存

信息填写

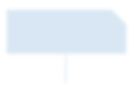
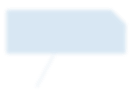
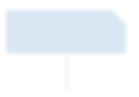
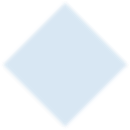
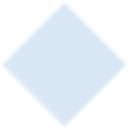
保存或关闭

调入专家信息

生成专家库内容信息

图 2-3 新增专家库并调入专家信息活动图

1. 新增专家资料并设置标签活动图，如图 2-4 所示：



管理员或领导角色

登录系统

专家资料管理

专家资料维护

新增

关闭

选择标签内容

设置标签

生成专家资料信息

保存

专家信息填写

关闭

保存或关闭

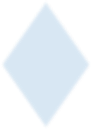
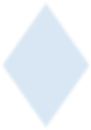
保存

生成专家标签信息

保存或关闭

图 2-4 新增真假资料并设置标签活动图

1. 会议资料上传与下载活动图，如图 2-5 所示：



下载资料成功

保存

下载

保存或关闭

关闭

秘书，领导或管理

员角色登录系统

活动管理

选择会议主题

管理

会议资料

关闭

上传

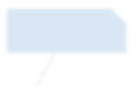
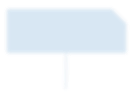
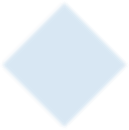
上传资料成功

保存

保存或关闭

图 2-5 会议资料上传与下载活动图

1. 设置联络员活动图，如图 2-6 所示：



联络员，领导登录

系统

工作联络管理

工作联络设置

新增

选择专家和联络员

关闭

设置成功

保存

保存或关闭

图 2-6 设置联络员活动图

1. 查找工作日志活动图，如图 2-7 所示：



管理员登录

系统管理

用户日志管理

输入要查找的用户

名

显示查找结果

图 2-7 查找工作日志活动图

### 功能需求

* + 1. 角色分析

以下从省教育与经济社会发展智库管理信息系统的实际需求分析出发，智库系统涉及到以下角色，如表 2-1 所示：

表 2-1 系统角色表

|  |  |
| --- | --- |
| 用户 | 说明 |
| 系统管理员 | 拥有整个系统所有权限，维护系统数据运行 |
| 领导 | 对智库的专家，成果，活动，工作联络等信息管理 |

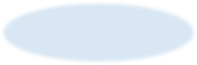
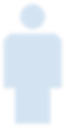
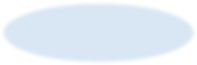
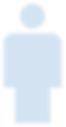
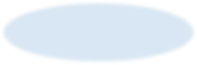
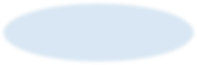
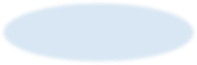
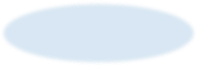
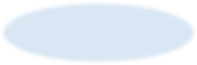
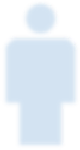
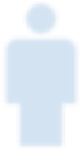
续表 2-1

|  |  |
| --- | --- |
| 秘书 | 对智库的活动进行安排和记录 |
| 联络员 | 对智库专家进行联络 |

* + 1. 业务功能

利用信息化手段，建设教育与经济社会发展智库管理信息系统，系统实现会议与活动管理、工作联络管理、专家资料管理、专家库管理、专家维度管理、成果库管理等，为省厅领导及时了解会议活动成果、专家库的活动记录、及献言献策内容的有效的管理，为领导科学决策提供基础数据支撑。

以下从业务角度出发，给出了系统的总体用例图，包含系统管理、专家库管理、成果库管理、标签管理、专家资料管理、活动管理、工作联络管理等用例，如图 2-8 所示：



系统管理

专家库管理

专家资料管理

管理员

领导

标签维度管理

工作联络管理

活动管理

联络员

成果库管理

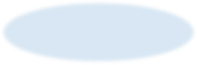
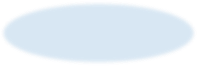
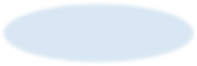
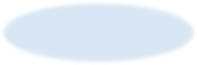
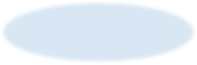
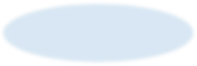
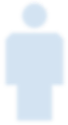
秘书

图 2-8 系统总体用例图

主要业务分为以下几个：

1. 系统管理

系统管理用例图如图 2-9 所示：



用户新增

<<包含>>

用户查询

<<包含>>

用户管理

<<包含>>

用户修改

<<包含>>

<<包含>>

用户删除

系统管理

管理员

<<包含>>

用户日志管理 <<包含>>

用户日志查询

图 2-9 系统管理用例图

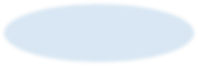
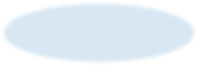
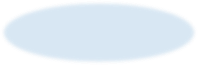
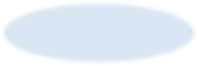
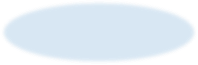
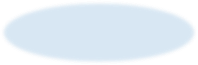
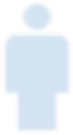
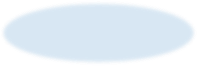
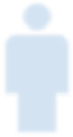
日志管理中用户日志查询用例如表 2-2 所示：

表 2-2 用户日志查询用例表

|  |  |
| --- | --- |
| 用例名称 | 用户日志查询 |
| 用例标识符 | YL01 |
| 参与者 | 管理员 |
| 简要说明 | 管理员查询用户信息操作信息 |
| 前置条件 | 管理员成功登陆后台 |
| 基本事件流 | 1. 管理员点击首页左栏系统管理，点击用户日志管理 2. 在输入框输入要查找的操作人，点击查询 3. 网站显示查询结果 |
| 后置条件 | 无 |

1. 专家资料管理

专家资料管理用例图如图 2-10 所示:



专家资料查询

专家资料新增

<<包含>>

管理员

专家资料查看

<<包含>>

专家资料管理

<<包含>>

设置专家标签

<<包含>>

<<包含>>

专家资料删除

领导

专家资料修改

图 2-10 专家资料管理用例图

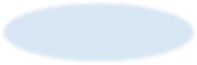
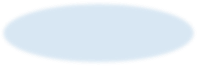
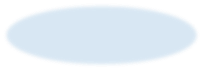
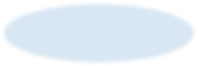
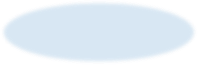
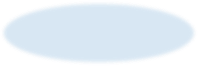
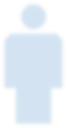
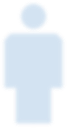
专家资料管理中专家资料新增用例如表 2-3 所示：

表 2-3 专家资料新增用例表

|  |  |
| --- | --- |
| 用例名称 | 专家资料新增 |
| 用例标识符 | YL02 |
| 参与者 | 管理员、领导 |
| 简要说明 | 管理员或领导新增专家资料信息，系统对新增信息验证后，保存信息 |
| 前置条件 | 管理员或领导成功登陆后台 |
| 基本事件流 | 1. 管理员点击首页左栏专家资料管理，点击专家资料维护 2. 点击新增按钮 3. 输入专家新增信息 4. 系统对输入信息进行验证是否合法 |
| 后置条件 | 新增成功 |

1. 专家库管理

专家库管理用例图如图 2-11 所示:



专家库新增

<<包含>>

管理员

专家库查询

<<包含>>

查询专家

专家库管理

<<扩展>>

<<包含>>

专家库查看 <<扩展>>

调入专家

<<包含>>

<<扩展>>

调出专家

领导

专家库删除

图 2-11 专家库管理用例图

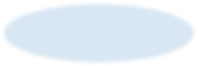
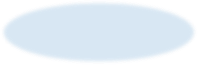
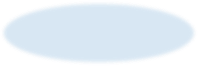
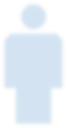
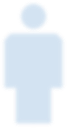
专家库管理中调入专家信息用例如表 2-4 所示：

表 2-4 专家库调入专家信息用例表

|  |  |
| --- | --- |
| 用例名称 | 专家库调入专家信息 |
| 用例标识符 | YL3 |
| 参与者 | 管理员、领导 |
| 简要说明 | 管理员或领导调入专家库内专家资料信息 |
| 前置条件 | 管理员或领导成功登陆后台，专家库已创建，专家资料已创建 |
| 基本事件流 | 1. 管理员点击首页左栏专家库管理，点击专家库维护 2. 找到要查询专家库，点击详情 3. 点击调入 4. 选择专家 |
| 后置条件 | 设置成功 |

1. 成果库管理

成果库管理用例图如图 2-12 所示:



成果新增

<<包含>>

管理员

成果查询

<<包含>>

成果库管理

<<包含>>

成果查看

<<包含>>

领导

成果删除

图 2-12 成果库管理用例

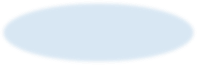
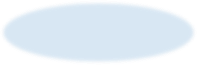
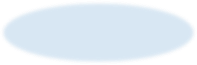
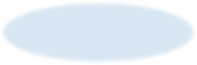
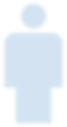
成果库管理中调入专家信息用例如表 2-5 所示：

表 2-5 成果查看用例表

|  |  |
| --- | --- |
| 用例名称 | 成果查看 |
| 用例标识符 | YL4 |
| 参与者 | 管理员、领导 |
| 简要说明 | 管理员或领导查看成果信息 |
| 前置条件 | 管理员或领导成功登陆后台，成果信息已创建在本系统 |
| 基本事件流 | 1. 管理员点击首页左栏成果库管理，点击成果库维护 2. 找到要查看的成果信息，点击查看 |
| 后置条件 | 跳转至成果信息展示界面 |

1. 标签维度管理

标签维度管理用例图如图 2-13 所示:



标签维度新增

<<包含>>

标签维度管理 <<包含>>

标签维度修改

<<包含>>

管理员

标签维度查询

<<包含>>

标签维度修改

图 2-13 标签维度管理用例图

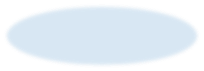
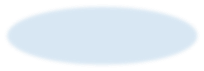
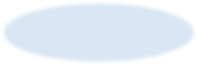
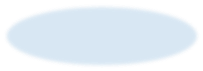
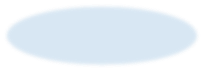
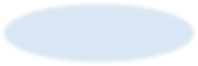
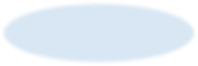
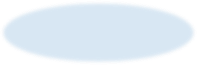
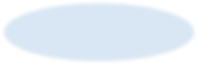
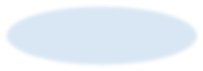
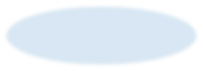
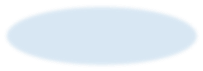
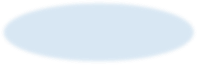
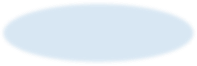
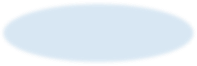
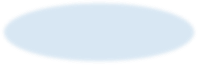
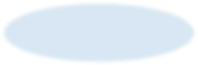
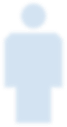
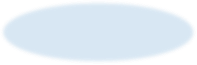
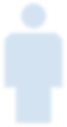
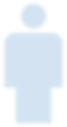
标签维度管理中标签维度修改用例如表 2-6 所示：

表 2-6 标签维度修改用例表

|  |  |
| --- | --- |
| 用例名称 | 标签维度修改 |
| 用例标识符 | YL5 |
| 参与者 | 管理员 |
| 简要说明 | 管理员修改标签维度，保存信息 |
| 前置条件 | 管理员成功登陆后台，要修改的标签维度已存在本系统 |
| 基本事件流 | 1. 管理员点击首页左栏标签维度管理，点击标签维度维护 2. 找到要修改的标签，点击修改按钮 3. 输入修改标签信息，保存信息 |
| 后置条件 | 标签信息更新 |

1. 活动管理

活动管理用例图如图 2-14 所示:



专家意见查询

<<包含>>

会议纪要查询

<<包含>>

活动主题查询

<<包含>>

会议资料查询

<<包含>>

<<包含>>

会议考勤查询

<<包含>>

管理员

<<包含>>

<<包含>>

<<包含>>

专家意见新增

<<包含>>

活动管理

<<包含>>

活动主题管理 <<包含>>

<<包含>>

会议纪要新增

会议考勤新增

领导

<<包含>>

<<包含>>

<<包含>>

<<包含>>

活动主题新增

<<包含>>

会议资料上传

<<包含>>

<<包含>>

<<包含>>

会议资料下载

活动主题查看

<<包含>>

会议纪要删除

秘书

会议资料删除

活动主题删除

专家意见删除

会议考勤删除

图 2-14 活动管理用例图

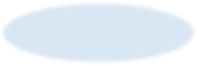
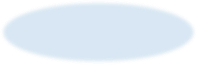
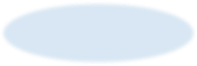
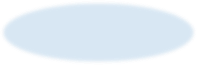
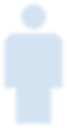
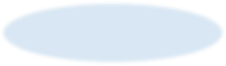
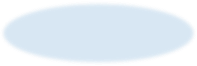
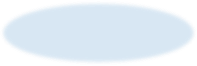
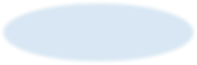
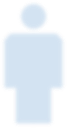
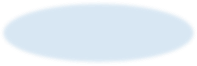
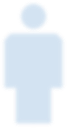
活动管理中会议资料上传用例如表 2-7 所示：

表 2-7 会议资料上传用例表

|  |  |
| --- | --- |
| 用例名称 | 会议资料上传 |
| 用例标识符 | YL6 |
| 参与者 | 管理员、领导、秘书 |
| 简要说明 | 操作员上传会议资料 |
| 前置条件 | 管理员、领导或秘书成功登陆后台，会议主题已存在本系统中 |
| 基本事件流 | 1. 操作人员点击首页左栏活动主题管理管理，点击活动主题维护 2. 选择活动主题，点击管理按钮 3. 点击会议资料，点击上传按钮 4. 从本机中选择上传资料，点击保存 |
| 后置条件 | 上传成功 |

1. 联络工作管理

工作联络管理用例图如图 2-15 所示:



查询联络信息

<<包含>>

工作联络设置

<<包含>>

管理员

<<包含>>

<<包含>>

联络员与专家联络新增

工作联络管理

联络员与专家联络删除

<<包含>>

工作联络新增

<<包含>>

领导

工作联络记录

<<包含>>

工作联络查询

<<包含>>

<<包含>>

工作联络修改

<<包含>>

工作联络删除

联络员

工作联络查看

图 2-15 工作联络管理用例图

工作联络管理中工作联络记录修改用例如表 2-8 所示：

表 2-8 工作联络记录修改用例表

|  |  |
| --- | --- |
| 用例名称 | 工作联络记录修改 |
| 用例标识符 | YL7 |
| 参与者 | 联络员 |
| 简要说明 | 联络员修改联络记录信息，保存信息 |
| 前置条件 | 联络员成功登陆后台 |
| 基本事件流 | 1. 联络人员点击首页左栏工作联络管理管理，点击工作联络记录 2. 找到要修改的工作联络记录信息，点击修改按钮 3. 输入修改信息，点击保存 |
| 后置条件 | 修改成功 |

### 非功能需求

* + 1. 环境需求

服务器端：

表 2-9 服务器端环境

|  |  |
| --- | --- |
| 系统所使用数据库 | Mysql 5.0 |
| 操作系统 | Windows10 专业版 |
| Web 服务器 | Tomcat 8.0 |

客户端：

表 2-10 客户端环境

|  |  |
| --- | --- |
| 操作系统 | Windows7 或其以上版本操作系统 |
| 浏览器 | IE 6.0 以及其以上版本（IE7.0、IE8.0），Chrome 浏览器  等主流浏览器 |
| 分辨率 | 最佳效果为 1028×960 像素 |

* + 1. 性能需求

1. 可维护性

维护方法：系统能够真正的做到使自身在数据遭恶意操作下损毁的情况下将查询到操作记录，实现其数据的恢复，提高系统的实用性和可靠性。

1. 易操作性

界面设计：系统应提供实用，简洁和美观的用户管理界面，充分考虑本系统为省教育厅所属的网站，界面应该严肃，简单易学，便于操作。

1. 可扩展性

功能扩展：一个系统的功能不是一成不变的，随着时代前进，日后将会有更多的功能点加入进来，因此，开发人员在系统的设计过程中应充分的考虑到日后的工作， 提前预留出数据接口，动态管理好每个模块，尽量不要写死模块，方便日后的工作拓展。

软硬件升级：系统应能及时的进行软件和硬件的升级工作，包括服务器的优化， 数据库的优化和软件代码的优化等内容，保证系统能够跟上时代步伐，促进智库系统的升级改造。

1. 开放性

系统的数据格式应充分符合国家党政办公软件的使用标准，在标准内确保应用程序具有良好的互操作性，要注意与其他软件接口的开放，使数据充分整合起来。

* + 1. 安全需求

1. 容错性

当用户产生错误操作时，能够快速的并且友好的提示用户进行下一步操作，防止产生非法数据，例如提示用户重新输入数据等操作

1. 系统的外在环境安全

系统的外在安全是每个企业级应用程序开发者都应注意的内容，要充分考虑到网络安全的复杂性，建立多措施处理机制。包括防火墙的设置，检测 ip 地址，提供非法用户频繁登录的报警机制，防止未经授权的异常用户非法访问系统。

1. 系统内部安全

开发者不应只关注系统外的非法用户对系统的入侵，还应确保合法的授权用户操作的安全性，防止系统内个人用户信息遭受泄露。

1. 系统运行安全

系统运行安全是每一个软件系统最重要的环节。在硬件方面，采用市面上好的且有安全保障的服务器，需要能提供数据资源的备份，在极端情况下，保证数据资料能够找回，挽回用户最大的利益。在软件方面，做好某些数据的加密传输，特别是用户密码等信息。

# 总体设计

### 系统设计的原则

在充分考虑到本系统的总体需求目标和系统的运行环境前提下，本系统的设计将严格的遵循以下的系统设计原则：

1. 统一设计原则

在系统设计的过程中，每个代码模块尽量做到风格统一，方便日后的系统维护。包括使用技术的框架，数据封装格式，存储结构都要做到统一。

1. 结构稳定性原则

系统结构稳定是一个系统能否取得成功的必要因素，一个系统若经常发生数据格式错误，用户界面点击错误等情况，轻微的可能到宕机，严重的则会发生数据泄露的危险。因此，在系统设计中，开发人员应确保每个功能点都处于运行稳定状态。

1. 先进性原则

开发人员在使用框架技术开发时，必须采用成熟的，稳定的技术，在这一前提条件下，做到使用符合国内外先进的软件技术，借鉴已有框架的成熟模式，保证系统有较强的生命力和扩展能力。

1. 可扩展性

信息系统设计要充分考虑到系统业务未来发展的需要，满足未来的扩展功能需求， 预留出数据接口，需要尽可能的设计的简明，降低各个功能模块耦合度，并充分考虑系统的兼容性。系统需要能够支持对多种格式数据的存储，是十分必要的。

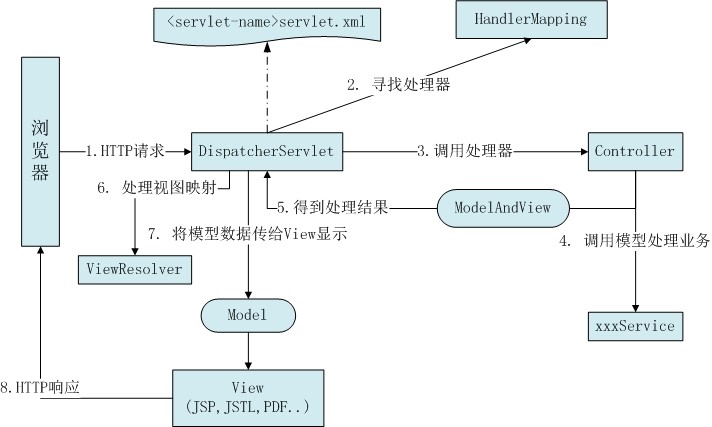
### 系统体系结构设计

在目前的系统设计中，主要采用的体系结构由 B/S 结构、C/S 结构以及 B/S 和 C/S 混合结构，本系统采用基于 Maven 的 SSM 总体架构设计的 B/S 结构，maven 是专门用于构建和管理 java 相关项目的工具，使用 maven 管理的 java 项目都有着相同的项目结果，利用 pom.xml 维护当前项目使用了那些 jar 包，将所有 jar 包都放在本地“仓库”里，方便项目的更新迭代。SSM 框架是 Spring 框架，Spring MVC 框架，MyBatis 框架的集成缩写，是目前比较主流的 Java EE 企业级框架，它能够适用于搭建各种大型的企业级复杂的系统。

其中， Spring 框架是为了解决开发企业级应用程序的复杂过程产生的框架。它能够动态的管理好每个对象，不用一个一个的 new 出对象，它不仅适用于服务器端开发，在后端开发的过程中，能够是 Java 程序运行速度更快，因为这些底层代码都是经过不断的迭代更新，能够最大限度保证数据的安全性。Spring 框架是一个轻量级的控制反转（IOC）和面向切面（AOP）的容器框架。 IOC 也称为控制翻转，它使用工厂模

板将对象交付给容器管理，程序员只需要在 Spring 配置文件中设置相应的 bean 并定义相关属性，然后让 spring 容器生成实例对象和类管理对象。 面向切面（AOP）完善和完善的面向对象（OOP）技术，在该系统中，采用动态代理技术，通过截取消息来修改消息，以获取对象原始行为的执行。

Spring MVC 框架是 Spring 家族的另一个框架，它利用 MVC 的设计思想，能够快速实现 java web 技术，能够实现灵活的对数据进行验证。它主要分为三层，分别由控制器层，业务层，数据持久层组成，控制器层是对前台页面转来的字符串数据进行解析。业务层是对控制器层转来的封装好的数据进行逻辑处理，进行再次封装。数据持久层是对业务层的数据进行处理，筛选出想要的数据，进行数据库事务操作。

Spring MVC 框架系统结构如图 3-1 所示：

流程说明：

图 3-1 Spring MVC 系统结构图

* + 1. 客户端（浏览器）向服务器发送请求，直接请求到 DispatcherServlet。
    2. DispatcherServlet 根据请求的信息调用 HandlerMapping，解析请求对应的 Handler 组件。
    3. 解析到对应的 Handler 组建之后，开始转而由 HandlerAdapter 适配器处理。
    4. HandlerAdapter 会根据 Handler 来调用真正的处理器开始处理请求，并处理相应的业务逻辑版块。
    5. 处理器处理完业务后，会返回一个 ModelAndView 对象，Model 是返回的数据对象，View 是个逻辑上的视图。
    6. ViewResolver 会根据逻辑上的 View 来查找实际的 View。
    7. DispaterServlet 把返回的 Model 传给 View。
    8. 最后通过 View 返回给请求者（浏览器）

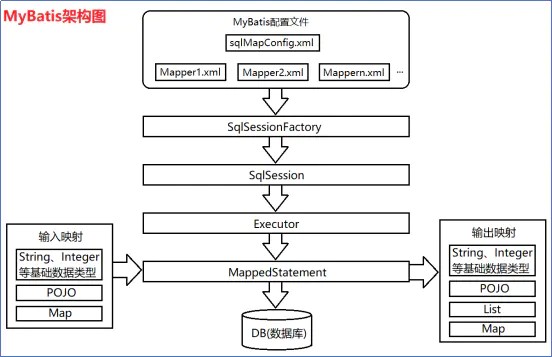
MyBatis 框架是一个十分优秀的数据持久层框架，它利用 xml 的方式对数据事务进行管理，不用频繁的对数据库进行连接操作和断开操作，减少了代码的冗余度。不像jdbc 技术那样，每次数据库操作都需要写冗长的注册驱动，创建事务等等操作，只需让开发者关注 SQL 语句本身，对事务操作内容进行动态的管理，还能写出动态 SQL， 让数据库 SQL 操作执行的速度更快。底层有专门的安全管理，保证了数据库操作的安全性。系统结构图如 3-2 所示：

图 3-2 Mybatis 系统结构图

MyBatis 框架相对于 JDBC 的好处：

1. 使用传统方式的 JDBC 技术访问数据库：

①使用 JDBC 技术访问数据库会产生大量重复代码，比如注册驱动数据库、获取连接数据库、获取传输器、释放资源等。

②JDBC 自身没有数据库连接池的，还会频繁的创建连接和关闭连接，代码执行效率低。

③SQL 是写死在程序中，一旦修改 SQL 语句，需要对整个代码重新编译，加重维护过程。

④对查询 SQL 执行后返回的 ResultSet 对象，需要手动处理，不易处理。

1. 使用 MyBatis 框架访问数据库：

①Mybatis 框架对 JDBC 技术对了封装，可以简化 JDBC 代码。

②Mybatis 框架支持数据库连接池（也可以配置其他的连接池），因此可以提高程序的执行效率。

③Mybatis 框架是将 SQL 配置在 mapper 文件中，修改 SQL 只是修改配置文件中的事务语句，类不需要重新编译。

④对查询 SQL 执行后返回的 ResultSet 对象，Mybatis 会帮我们处理，转换成 Java 对象。

### 系统功能结构设计

本系统总包图如下图 3-3 所示:

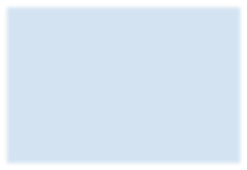
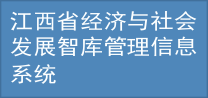
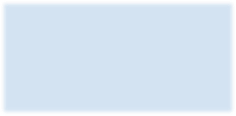


图 3-3 系统总包图



如上图所示，根据需求分析得到结果，系统一共分为七个子系统模块，以下详细说明：

1. 系统管理

系统管理分为用户管理和用户日志管理，用户管理包括用户信息新增，用户信息修改，用户信息查询，用户信息删除。用户日志管理是对用户在本系统操作的记录， 包括用户日志查询。系统管理是只有管理员才有权限处理的模块。

1. 专家库管理

专家库管理主要是对专家库中专家信息的维护，包括新增专家库，查询专家库， 修改专家库，删除专家库，查看专家库详情。在专家库中，可以调入系统中的专家至专家库，也可以从专家库中将专家调出，专家库包含不同类型的专家，主要根据专家标签来区分，能够查询专家资料。专家库管理是只有管理员，领导才有权限处理的模块。

1. 专家资料管理

专家资料管理主要是对专家资料的维护，包括新增专家资料的信息，查询专家资料的信息，修改专家资料的信息，查看专家资料的信息，删除专家资料的信息，根据标签库中的标签对专家设置标签。专家资料管理是只有管理员，领导才有权限处理的模块。

1. 成果库管理

成果库管理主要是对专家的成果信息进行维护，包括新增成果信息，查询成果信息，查看成果信息，删除成果信息。成果库管理是只有管理员和领导才有权限处理的模块。

1. 活动管理

活动管理主要是对专家活动信息的维护，包括新增活动主题，查询活动主题，查看活动主题，删除活动主题和管理活动内容。其中管理活动内容包括活动的会议纪要、会议资料、专家意见及会议考勤的维护。会议纪要维护包括查询会议纪要，新增会议纪要，删除会议纪要，查看会议纪要。会议资料维护包括查询会议资料，上传会议资料，下载会议资料，删除会议资料。专家意见维护包括查询专家意见，新增专家意见， 查看专家意见，删除专家意见。会议考勤维护包括查询会议考勤，新增会议考勤，删除会议考勤。活动管理是只有管理员、领导和秘书才有权限处理的模块。

1. 工作联络管理

工作联络管理主要是对工作联络模块的维护，包括设置工作联络相匹配的专家和联络员，这一功能只有管理员、领导才能有权限处理。工作联络记录维护包括查询联络记录、查看联络记录、删除联络记录、修改联络记录，这一功能可以由领导、管理员和联络员一起处理。

1. 标签维度管理

标签维度管理主要是对标签维度的维护，包括标签查询，标签新增，标签删除，

标签修改。标签维度管理只有管理员才有权限进行处理的模块。

# 数据库设计

### 概念结构设计

* + 1. 设计思路

数据库设计中概念设计遵循 E-R 模式，首先设计局部实体 E-R 关系，再讲局部中各个子 E-R 模式综合成一个全局 E-R 模式，经过实体关系解耦之后，最终得到一个优化的E-R 模式。根据本系统的用户实体，即管理员、领导、秘书及联络员角色的各种操作模块，以及模块与模块之间的联系进行设计。管理员用户能够设置标签维度，管理其他用户，管理系统日志。领导角色可以对专家库，专家资料，活动信息，成果信息，专家联络信息等进行管理，秘书对活动信息进行管理，联络员可以对工作联络进行管理。专家资料组成专家库。根据这些基本信息进行本系统的概念设计。

* + 1. E-R 图

根据前期设计得到的实体来建立各个实体之间的联系，本系统的实体繁多，包括管理员，成果库，系统日志，标签维度，活动主题，会议纪要，会议资料，会议考勤，专家意见，工作联络，专家资料等等，先按照系统模块中实体之间的联系，构建出局部的实体关系，并区分出一对一关系，一对多关系，多对多关系。并且需要对多对多关系进行解耦，方便之后的数据库设计中表需要解耦。如果一个实体内部还存在联系，需要划分出子实体。本系统的实体关系如下：

* + - 1. 本系统的实体关系图，如图 4-1 所示。

专家意见

1

设置

N

1

管理

N

系统日志

标签维度

管理员

N

1

N

1

N

1

设置

浏览

1

N

编写

N

N

N

维护

维护

N

专家库

N

1

1

专家资料

联络记录

用户

专家资料-专家库

联络记录

工作联络关系

N

N

包含

N

1

N

会议考勤

活动信息

会议资料

成果信息

会议纪要

图 4-1 实体关系图

* + - 1. 用户信息的实体图，如图 4-2 所示。实体：用户

属性：编号，用户名，密码，邮箱，创建时间，用户类别，手机号。

手机号

用户类别

邮箱

密码

用户名

图 4-2 用户实体图

编号

用户

创建时间

* + - 1. 专家资料信息的实体图，如图 4-3 所示。实体：专家资料信息

属性：用户编号，用户名，密码，工作单位，照片，联系地址，国籍，标签，职称， 出生日期，创建时间，证件号。

照片

联系地址

工作单位

证件号

密码

国籍

创建时间

用户名

标签

职称

出生日期

编号

专家

图 4-3 专家资料实体图

* + - 1. 专家库的实体图，如图 4-4 所示。实体：专家库

属性：编号，专家库名，备注，创建时间。

备注

创建时间

专家库

图 4-4 专家库实体图

编号

专家库名

* + - 1. 成果库的实体图，如图 4-5 所示。实体：成果库

属性：编号，成果名称，专家编号，成果信息，创建时间。

专家编号

创建时间

成果库

成果信息

成果名称

编号

图 4-5 成果库实体图

* + - 1. 活动主题的实体图，如图 4-6 所示。实体：活动主题

属性：编号，活动主题名称，内容信息，举办时间，举办地址，举办天数，嘉宾姓名，举办单位，主题类型，创建时间。

举办地址

内容信息

主题类型

举办时间

嘉宾姓名

举办天数

主办单位

活动主题名称

创建时间

编号

活动主题

图 4-6 活动主题实体图

* + - 1. 会议资料的实体图，如图 4-7 所示。实体：会议资料

属性：编号，活动主题编号，资料名称，文件类型，创建人编号，资料地址，创建时间。

资料地址

创建人编号

文件类型

资料名称

活动主题编号

创建时间

会议资料

图 4-7 会议资料实体图

编号

* + - 1. 专家意见的实体图，如图 4-8 所示。实体：专家意见

属性：编号，活动主题编号，专家编号，意见内容，创建时间。

意见内容

专家编号

创建时间

专家意见

图 4-8 专家意见实体图

编号

活动主题编号

* + - 1. 会议纪要的实体图，如图 4-9 所示。实体：会议纪要

属性：编号，活动主题编号，会议纪要名称，阶段，会议纪要内容，创建时间。

阶段

会议纪要内容

会议纪要名称

图 4-9 会议纪要实体图

创建时间

编号

活动主题编号

会议纪要

* + - 1. 会议考勤的实体图，如图 4-10 所示。实体：会议考勤

属性：编号，活动编号，专家编号，总人数，实到人数，创建时间。

总人数

实到人数

活动编号

图 4-10 会议考勤实体图

创建时间

编号

专家编号

会议考勤

* + - 1. 工作联络的实体图，如图 4-11 所示。实体：工作联络

属性：编号，专家编号，联络员编号，创建时间。

联络员编号

工作联络

专家编号

图 4-11 工作联络实体图

编号

创建时间

* + - 1. 工作联络记录的实体图，如图 4-11 所示。实体：工作联络记录

属性：编号，专家编号，联络员编号，联络方式，联络主题，联络信息，创建时间。

联络信息

创建时间

工作联络记录

联络主题

专家编号

联络员编号

### 逻辑结构设计

联络方式

编号

图 4-12 工作联络记录实体图

* + 1. 设计思路

1. 将需求转化为数据库的逻辑模型
2. 通过 E-R 图的型式对逻辑模型进行展示
3. 同所用的具体 DBMS 系统无关
   * 1. 逻辑模型

用户表：（编号，用户名，密码，邮箱，创建时间，用户类别，电话，删除状态） 专家资料表：（用户编号，用户名，密码，工作单位，照片，联系地址，国籍，标签，

职称，出生日期，创建时间，证件号，删除状态）

专家库表：（编号，专家库名，备注，创建时间）

成果库表：（编号，成果名称，专家编号，成果信息，创建时间）

活动主题表：（编号，活动主题名称，内容信息，举办时间，举办地址，举办天数，嘉宾姓名，举办单位，主题类型，创建时间）

会议资料表：（编号，活动主题编号，资料名称，文件类型，创建人编号，资料地址，创建时间）

日志表：（编号，用户编号，内容，模块，创建时间）

专家库-专家关联表：（编号，专家库编号，专家编号，创建时间） 专家-标签管理表：（编号，专家编号，标签编号，创建时间）

专家意见表：（编号，活动主题编号，专家编号，意见内容，创建时间）

会议纪要表：（编号，活动主题编号，会议纪要名称，阶段，会议纪要内容，创建时间）

会议考勤表：（编号，活动编号，专家编号，总人数，实到人数，创建时间） 工作联络表：（编号，专家编号，联络员编号，创建时间）

工作联络记录：（编号，专家编号，联络员编号，联络方式，联络主题，联络信息，创建时间）

### 物理结构设计

* + 1. 存取方式

本文研究采用 MYSQL 数据库进行数据的存储。

现行的 DBMS 一般地都提供了多种存取方法如索引法、HASH 法等。本文中用的是索引法。在数据库中，索引能够允许应用程序快速地找到表中的数据，而不必扫描整个数据库。索引就是表中数据和相应的存储位置的列表。使用索引能够大大减少数据的查询时间。

* + 1. 存储结构

1. 用户表 user，如表 4-15 所示：

表 4-15 用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 11 | 主键 |
| 2 | username | 用户名 | varchar | 20 |  |
| 3 | password | 密码 | varchar | 200 |  |
| 4 | email | 邮箱 | varchar | 20 |  |
| 5 | mobile | 联系电话 | varchar | 20 |  |
| 6 | type | 用户类型 | tinyint | 4 |  |
| 7 | CreateTime | 创建时间 | datetime | 0 |  |
| 8 | deleted | 是否删除 | tinyint | 4 |  |

1. 专家资料表 expert，如表 4-16 所示：

表 4-16 专家资料表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 11 | 主键 |
| 2 | work\_title | 职称 | varchar | 255 |  |
| 3 | organization | 工作单位 | varchar | 255 |  |
| 4 | information | 介绍 | varchar | 255 |  |
| 5 | name | 姓名 | varchar | 255 |  |
| 6 | nation | 国籍 | varchar | 255 |  |
| 7 | idnumber | 证件号 | varchar | 255 |  |
| 8 | birth | 出生日期 | vrchar | 255 |  |
| 9 | address | 联系地址 | varchar | 255 |  |
| 10 | file\_path | 照片路径 | varchar | 255 |  |
| 11 | createtime | 创建时间 | datetime | 0 |  |
| 12 | deleted | 是否删除 | tinyint | 4 |  |

1. 专家库表 expert\_group，如表 4-17 所示：

表 4-17 专家库表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | name | 专家库名 | varchar | 255 |  |
| 3 | remark | 备注 | varchar | 255 |  |
| 4 | createtime | 创建时间 | datetime | 0 |  |

1. 成果表 achievements，如表 4-18 所示：

表 4-18 专家库表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | title | 专家库名 | varchar | 255 |  |
| 3 | expert\_id | 专家编号 | int | 11 |  |
| 4 | information | 成果内容 | text | 0 |  |
| 5 | createtime | 创建时间 | datetime | 0 |  |

1. 活动表 activity，如表 4-19 所示：

表 4-19 活动表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | type | 活动类型 | varchar | 255 |  |

续表 4-19

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | theme | 活动主题 | varchar | 255 |  |
| 4 | information | 成果内容 | text | 0 |  |
| 5 | organization | 举办单位 | varchar | 255 |  |
| 6 | guest | 嘉宾 | varchar | 255 |  |
| 7 | holdtime | 举办时间 | varchar | 255 |  |
| 8 | days | 天数 | int | 4 |  |
| 9 | address | 活动地址 | varchar | 255 |  |
| 10 | createtime | 创建时间 | datetime | 0 |  |

1. 会议资料表 file，如表 4-20 所示：

表 4-20 会议资料表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | activity\_id | 活动编号 | int | 11 |  |
| 3 | name | 资料名称 | varchar | 255 |  |
| 4 | type | 资料内容 | varchar | 255 |  |
| 5 | user\_id | 创建人编号 | varchar | 255 |  |
| 6 | file\_path | 文件路径 | varchar | 255 |  |
| 7 | createtime | 创建时间 | datetime | 0 |  |

1. 日志表 log，如表 4-21 所示：

表 4-21 日志表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | user\_id | 用户编号 | int | 11 |  |
| 3 | content | 操作内容 | varchar | 255 |  |
| 4 | modular | 模块 | varchar | 255 |  |
| 5 | createtime | 创建时间 | datetime | 0 |  |

1. 专家库-专家关联表 expert\_group\_rela，如表 4-22 所示：

表 4-22 专家库-专家关联表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | expert\_id | 专家编号 | int | 11 |  |
| 3 | group\_id | 专家库编号 | int | 11 |  |
| 4 | createtime | 创建时间 | datetime | 0 |  |

1. 专家-标签关联表 expert\_group\_rela，如表 4-23 所示：

表 4-23 专家-标签关联表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | expert\_id | 专家编号 | int | 11 |  |
| 3 | tag\_id | 标签编号 | int | 11 |  |
| 4 | createtime | 创建时间 | datetime | 0 |  |

1. 专家意见表 idea，如表 4-24 所示：

表 4-24 专家意见表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | expert\_id | 专家编号 | int | 11 |  |
| 3 | activity\_id | 活动编号 | int | 11 |  |
| 4 | information | 意见内容 | varchar | 255 |  |
| 5 | createtime | 创建时间 | datetime | 0 |  |

1. 会议纪要表 summary ，如表 4-25 所示：

表 4-25 会议纪要表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | activity\_id | 活动编号 | int | 11 |  |
| 3 | title | 纪要标题 | varcahr | 11 |  |
| 4 | stage | 会议阶段 | varchar | 255 |  |
| 5 | information | 纪要内容 | varchar | 255 |  |
| 6 | createtime | 创建时间 | datetime | 0 |  |

1. 标签维度表 tag，如表 4-26 所示：

表 4-26 标签维度表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | name | 标签信息 | varchar | 255 |  |
| 3 | remark | 纪要标题 | varchar | 255 |  |
| 4 | create | 创建时间 | datetime | 0 |  |

1. 联络表 liaison，如表 4-27 所示：

表 4-27 联络表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | expert\_id | 专家编号 | int | 11 |  |

续表 4-27

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | liaison\_id | 联络人编号 | int | 11 |  |
| 4 | createTime | 创建时间 | datetime | 0 |  |
| 3 | liaison\_id | 联络人编号 | int | 11 |  |
| 4 | createTime | 创建时间 | datetime | 0 |  |

1. 工作联络表 liaison\_record，如表 4-28 所示：

表 4-28 工作联络表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | expert\_id | 专家编号 | int | 11 |  |
| 3 | liaison\_id | 联络人编号 | int | 11 |  |
| 4 | method | 联络方式 | varchar | 255 |  |
| 5 | title | 主题 | varchar | 255 |  |
| 6 | content | 联络内容 | varchar | 255 |  |
| 7 | createtime | 创建时间 | datetime | 0 |  |

1. 考勤表 attendance，如表 4-29 所示：

表 4-29 考勤表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 字段含义 | 类型 | 长度 | 备注 |
| 1 | id | 编号 | int | 10 | 主键 |
| 2 | activity\_id | 活动编号 | int | 11 |  |
| 3 | expert\_id | 专家编号 | int | 11 |  |
| 4 | attend\_actual | 应到人数 | varchar | 4 |  |
| 5 | attend\_should | 实到人数 | varchar | 4 |  |
| 6 | createtime | 创建时间 | datetime | 0 |  |

# 界面设计

### 界面关系图或工作流图

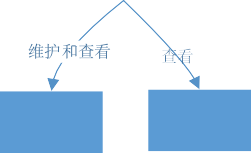
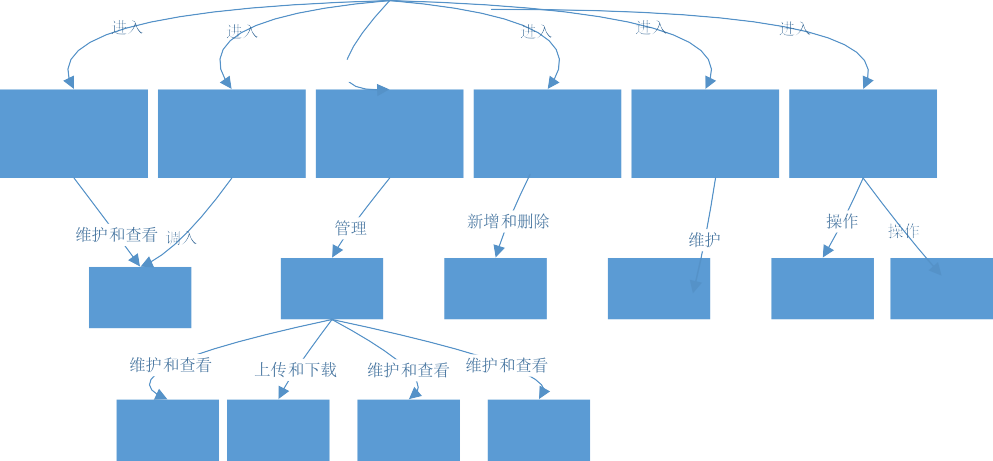
1. 管理员工作流图，如下图 5-1 所示：

开始



管理员

后台首页界面



进入

专家资料管理界面 专家库管理界面 活动库管理界面 标签维度管理界面 成果库管理

工作联络管理

用户信息

系统日志

专家信息

活动主题

标签信息

成果信息

工作联络设 工作联络记

置 录

会议纪要 会议资料

专家意见

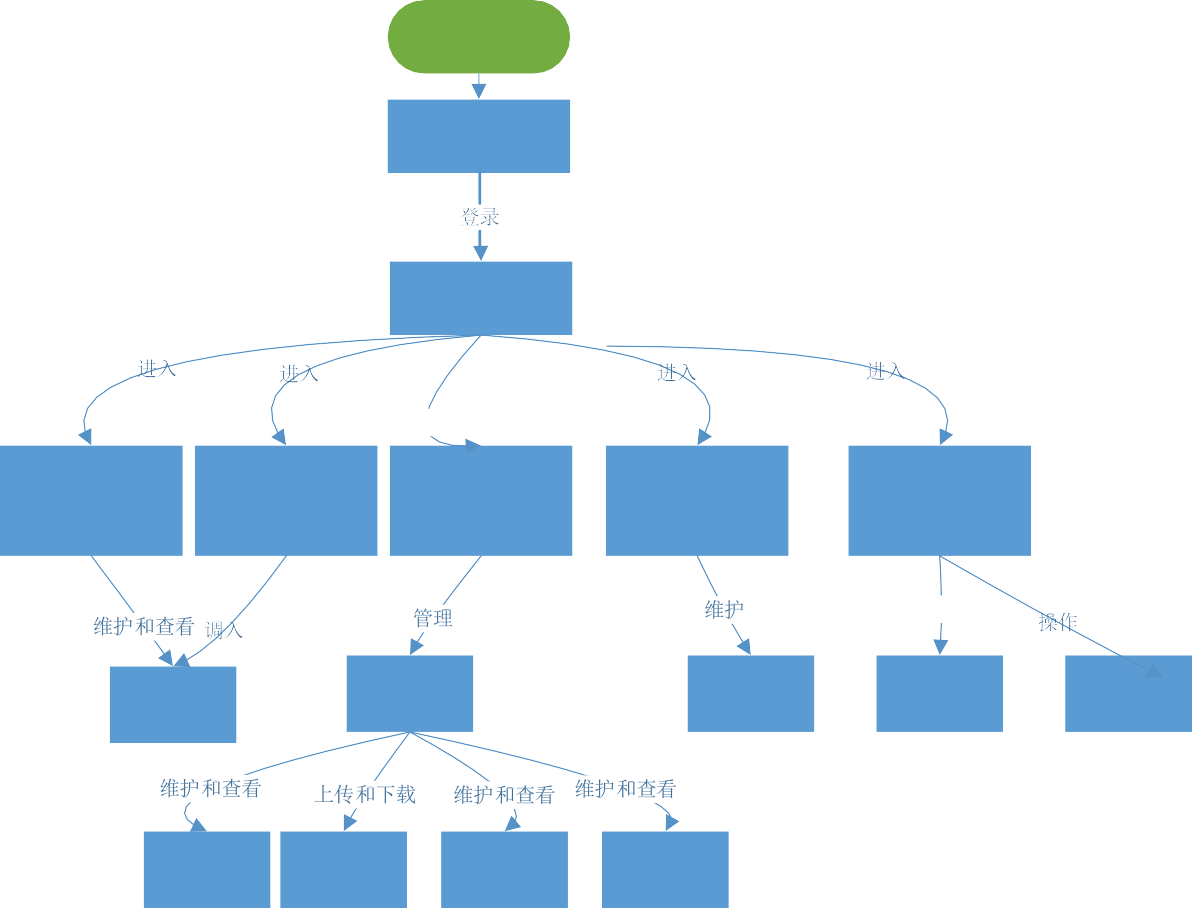
会议考勤

结束

系统管理界面

图 5-1 管理员工作流图

1. 领导工作流图，如下图 5-2 所示：



开始

领导

后台首页界面

进入

专家资料管理界面 专家库管理界面 活动库管理界面

成果库管理

工作联络管理

操作

专家信息

活动主题

成果信息

工作联络设

置

工作联络记

录

会议纪要

会议资料

专家意见

会议考勤

结束

图 5-2 领导工作流图

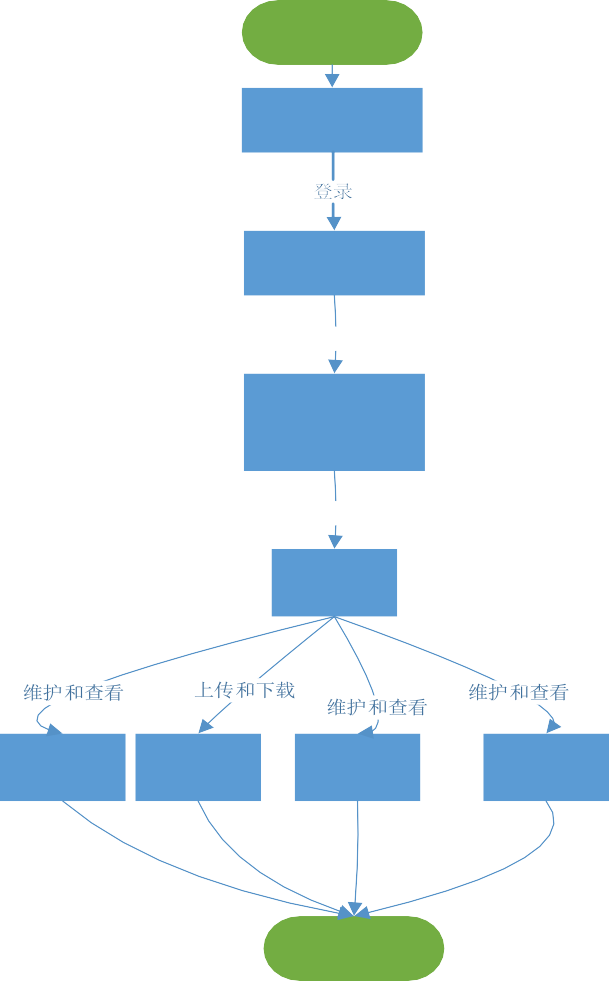
1. 秘书工作流图，如下图 5-3 所示：



操作

工作联络记

录



开始

秘书

后台首页界面

进入

活动库管理界面

管理

活动主题

会议纪要

会议资料

专家意见

会议考勤

结束

图 5-3 秘书工作流图

1. 联络员工作流图，如下图 5-4 所示：

开始



联络员



进入

工作联络管理

后台首页界面



结束

图 5-4 联络员工作流图

### 界面设计成果

* + 1. 主界面

后台主界面，如图 5-5 所示：

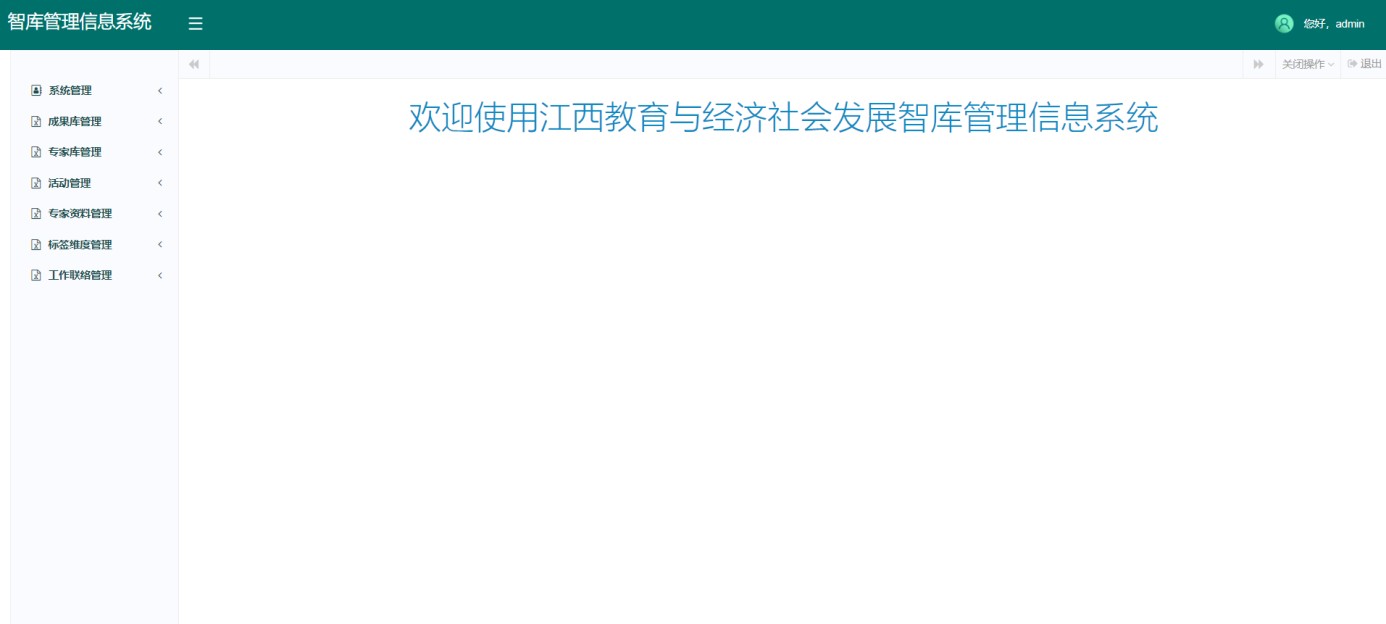


图 5-5 系统主界面

* + 1. 子界面

1. 用户管理子界面，如图 5-6 所示：

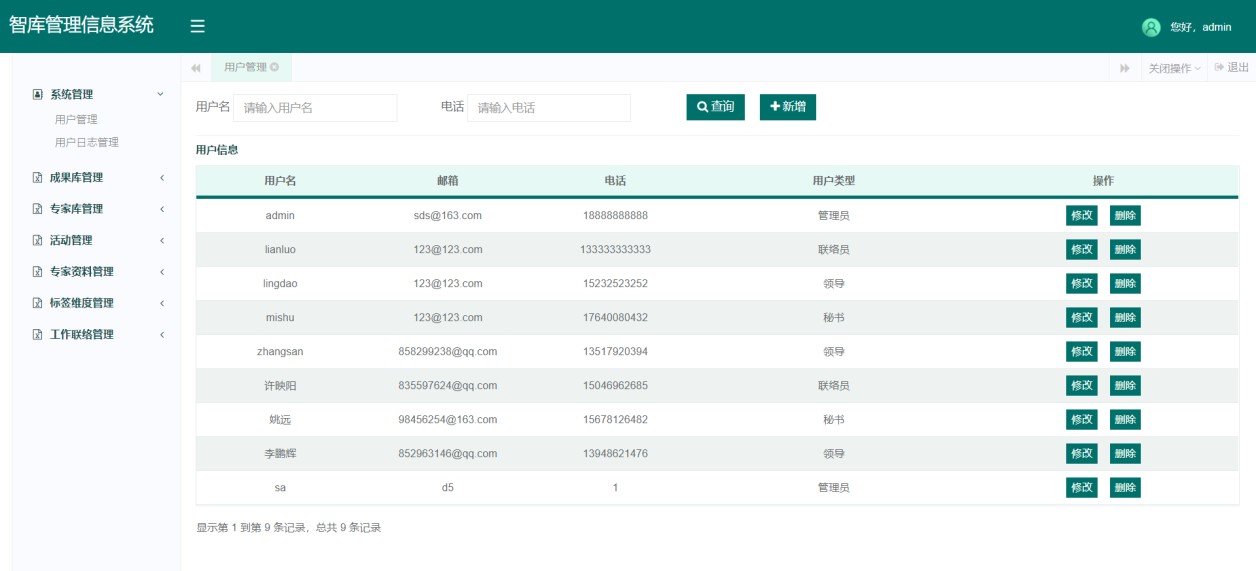


图 5-6 用户管理操作界面

1. 日志管理子界面，如图 5-7 所示：

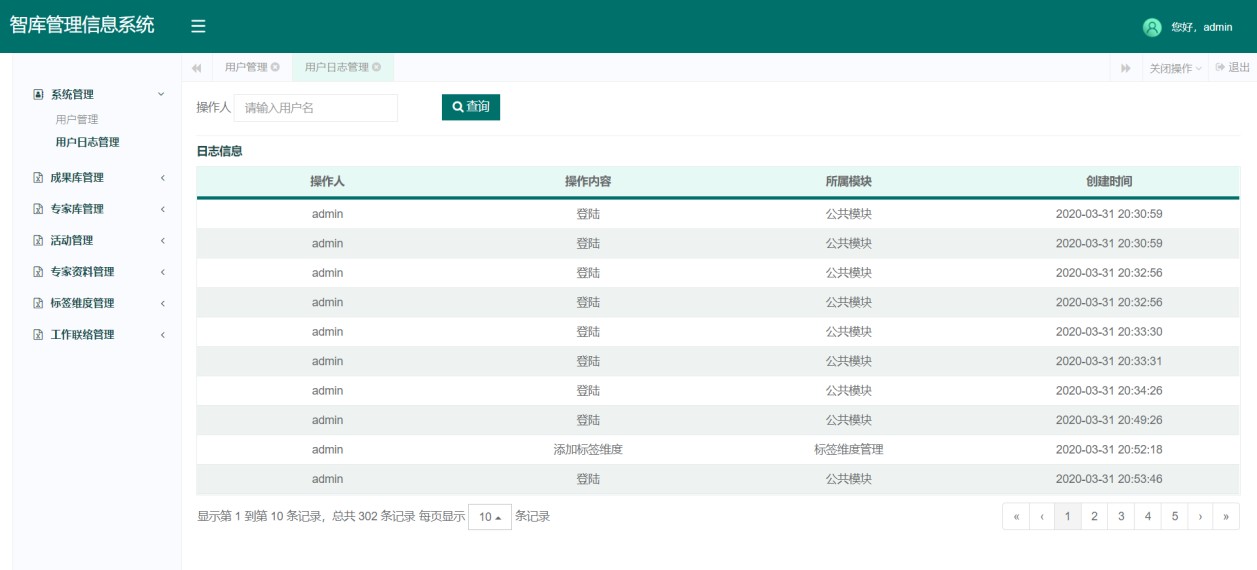


图 5-7 日志管理操作界面

1. 成果库管理子界面，如图 5-8,5-9 所示：

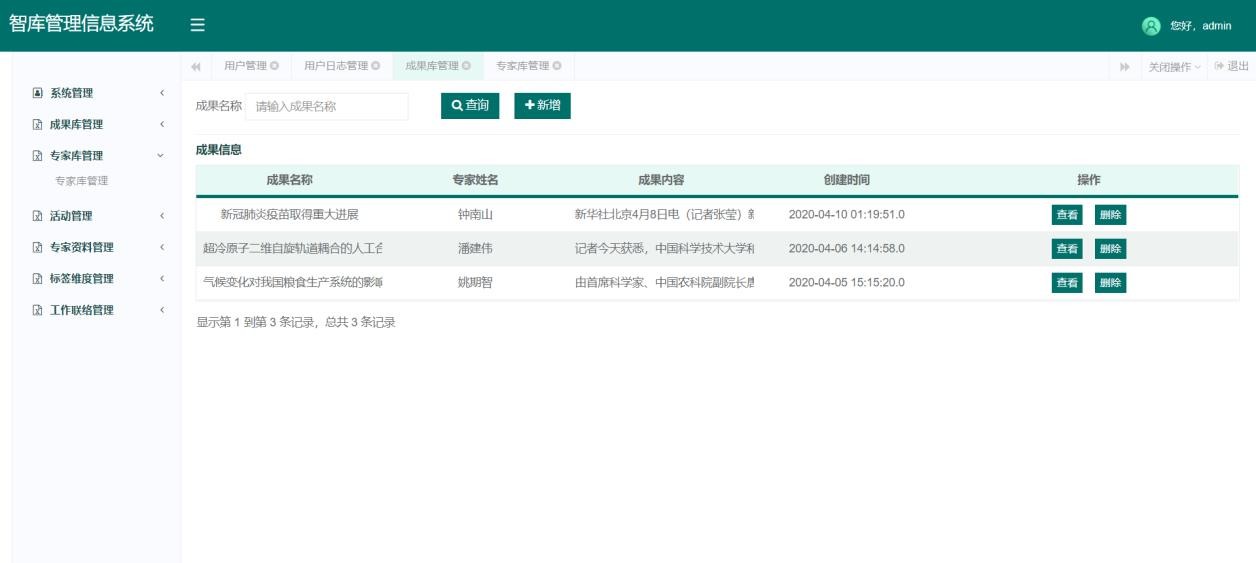


图 5-8 成果管理操作界面



图 5-9 成果详情界面

1. 专家库管理子界面，如图 5-10 所示：

图 5-10 专家库详情作界面

（5）活动管理子界面，如图 5-11,5-12 所示：



图 5-11 活动管理操作作界面

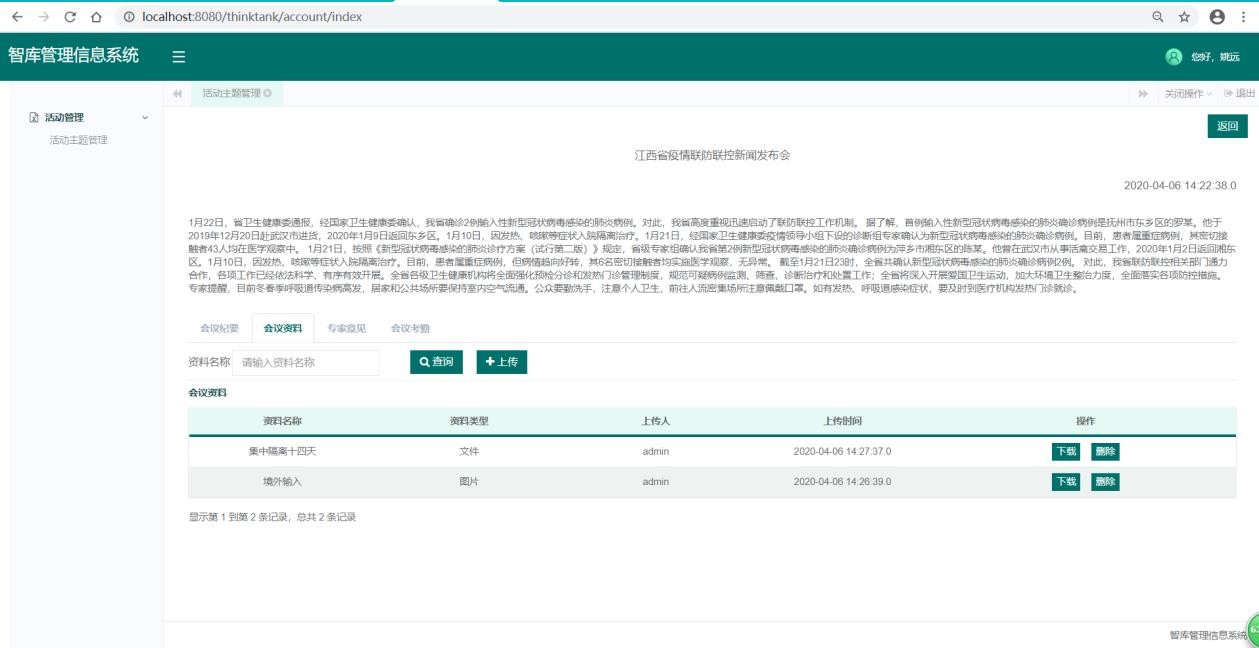


图 5-12 活动详情界面

1. 专家资料管理子界面，如图 5-13,5-14 所示：

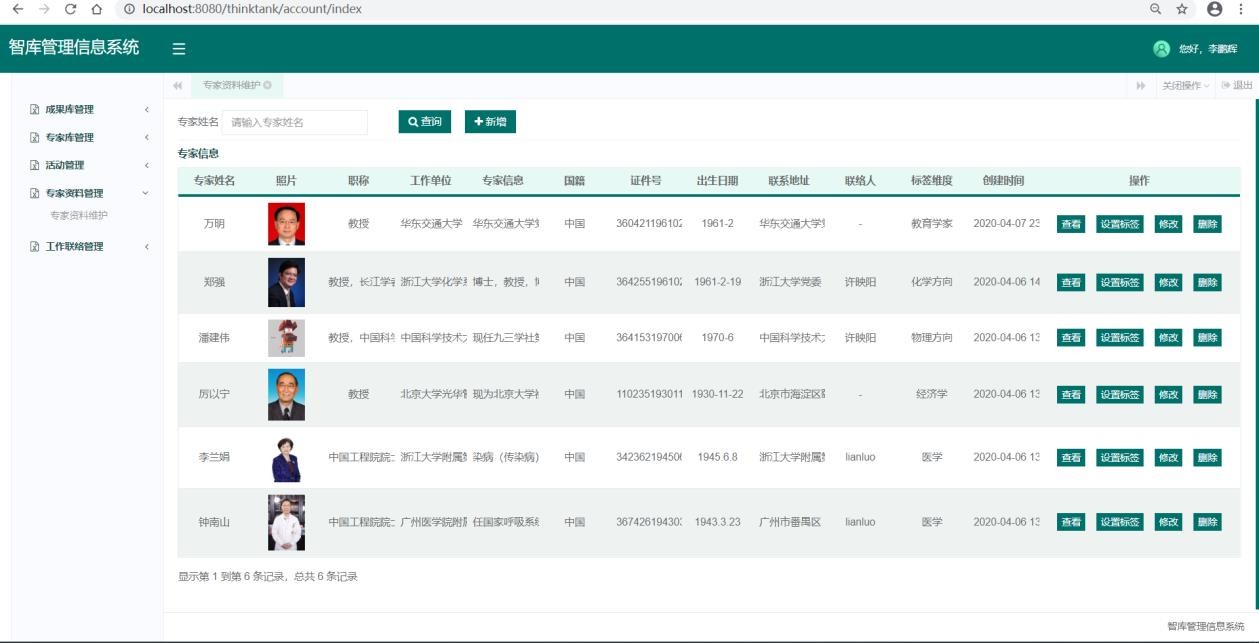


图 5-13 专家资料操作界面

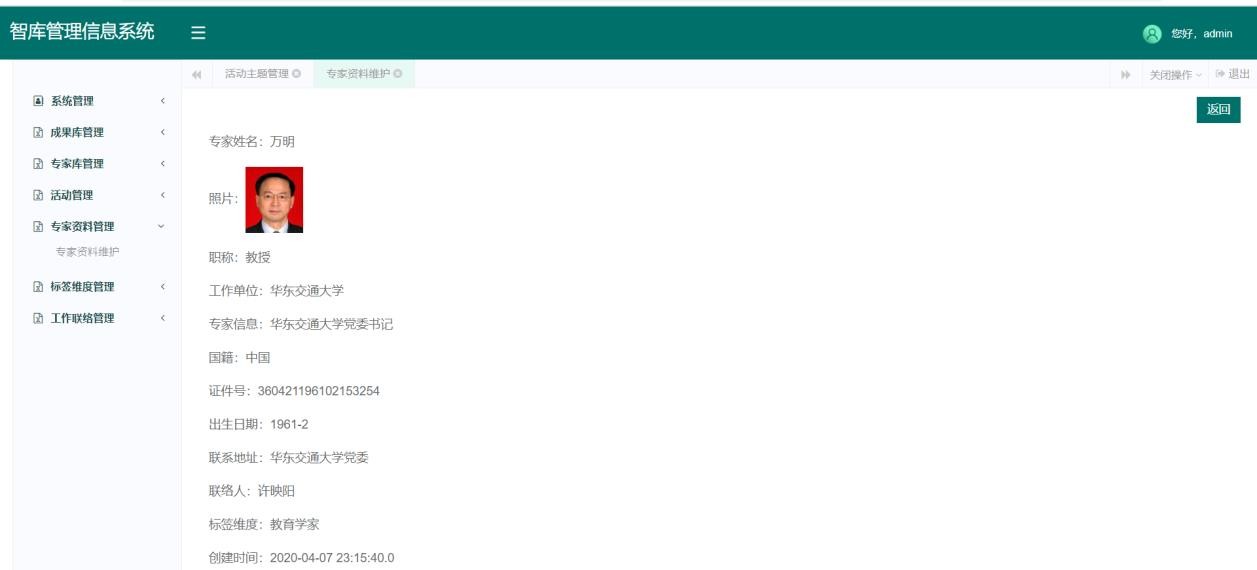


图 5-14 专家资料详情界面

1. 标签维度管理子界面，如图 5-15 所示：

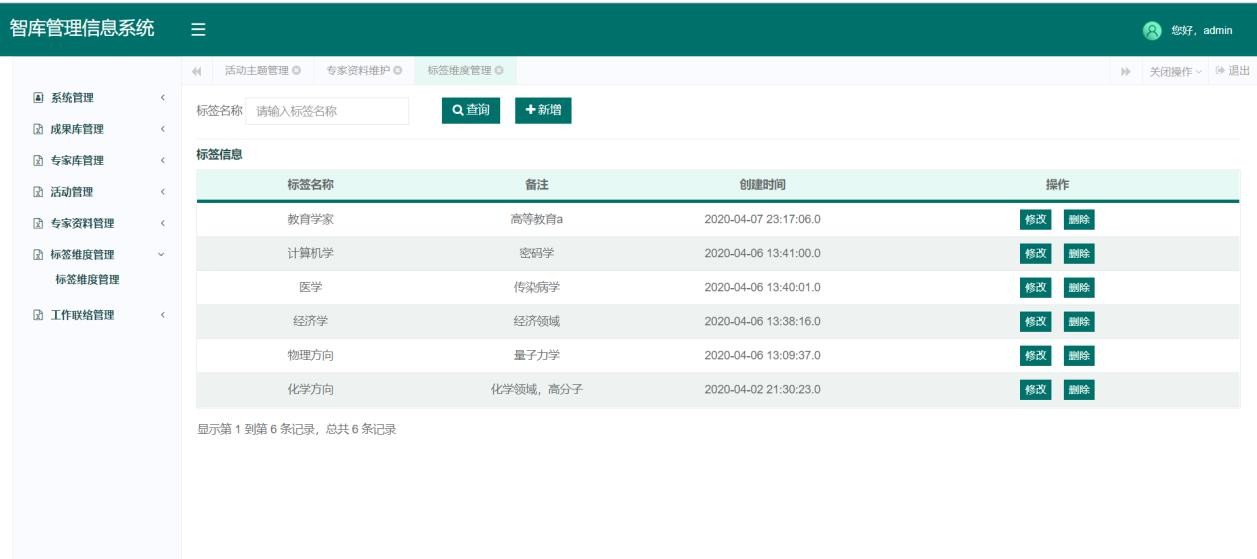
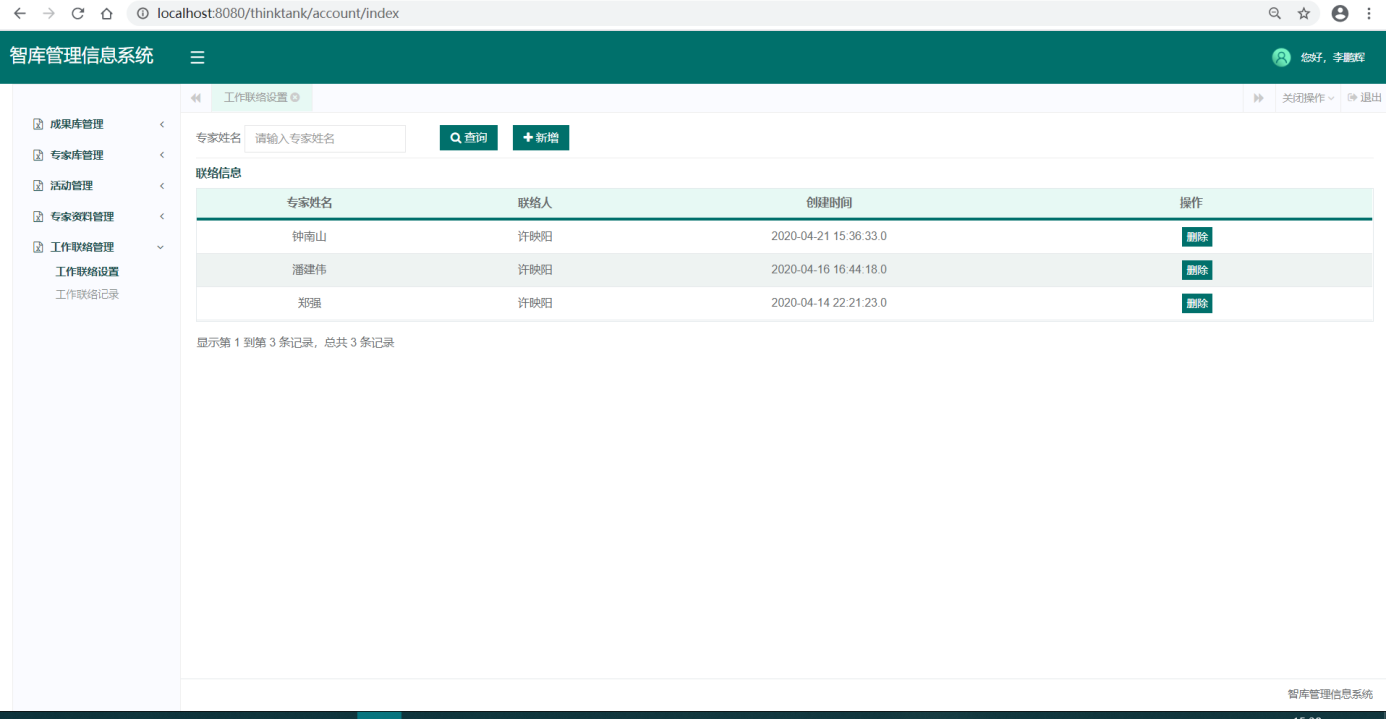


图 5-15 标签维度操作界面

1. 工作联络设置界面，如图 5-16 所示：



* 1. 工作联络设置界面

1. 工作联络记录界面，如图 5-17 所示：



* 1. 工作联络记录界面

# 详细设计

### 系统主要功能模块介绍

本子系统的功能模块主要包括:

1. 系统管理:

系统管理分为用户管理和用户日志管理，用户管理包括用户信息新增，用户信息修改，用户信息查询，用户信息删除。用户日志管理是对用户在本系统操作的记录，包括用户日志查询。系统管理是只有管理员才有权限处理的模块。

1. 专家库管理：

专家库管理主要是对专家库信息的维护，包括新增专家库，查询专家库，修改专家库，删除专家库，查看专家库详情。在专家库中，可以调入系统中的专家至专家库，也可以从专家库中将专家调出，能够查询专家资料。专家库管理是只有管理员，领导才有权限处理的模块。

1. 专家资料管理：

专家资料管理主要是对专家资料的维护，包括新增专家资料的信息，查询专家资料的信息，修改专家资料的信息，查看专家资料的信息，删除专家资料的信息，根据标签库中的标签对专家设置标签。专家资料管理是只有管理员，领导才有权限处理的模块。

1. 成果库管理：

成果库管理主要是对专家的成果信息进行维护，包括新增成果信息，查询成果信息， 查看成果信息，删除成果信息。成果库管理是只有管理员和领导才有权限处理的模块。

1. 活动管理：

活动管理主要是对专家活动信息的维护，包括新增活动主题，查询活动主题，查看活动主题，删除活动主题和管理活动内容。其中管理活动内容包括活动的会议纪要、会议资料、专家意见及会议考勤的维护。会议纪要维护包括查询会议纪要，新增会议纪要， 删除会议纪要，查看会议纪要。会议资料维护包括查询会议资料，上传会议资料，下载会议资料，删除会议资料。专家意见维护包括查询专家意见，新增专家意见，查看专家意见，删除专家意见。会议考勤维护包括查询会议考勤，新增会议考勤，删除会议考勤。活动管理是只有管理员、领导和秘书才有权限处理的模块。

1. 工作联络管理：

工作联络管理主要是对工作联络模块的维护，包括设置工作联络相匹配的专家和联络员，这一功能只有管理员、领导才能有权限处理。工作联络记录维护包括查询联络记录、查看联络记录、删除联络记录、修改联络记录，这一功能可以由领导、管理员和联络员一起处理。

1. 标签维度管理：

标签维度管理主要是对标签维度的维护，包括标签查询，标签新增，标签删除，标签修改。标签维度管理只有管理员才有权限进行处理的模块。

### 活动管理模块设计

* + 1. 功能结构设计

活动主题管理包括了四个子模块，分别是会议资料管理，用于活动内容中的会议资料的上传与下载；会议纪要管理，用于活动中的记录的会议纪要信息；会议考勤管理， 用于参与活动的人员考勤，便于统计数据；专家意见管理，用于与会专家的专家意见记录在案，为智库发展提供方向。包图如 6-1 所示：

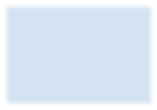


图 6-1 活动主题管理包图

* 1. .2 类图设计

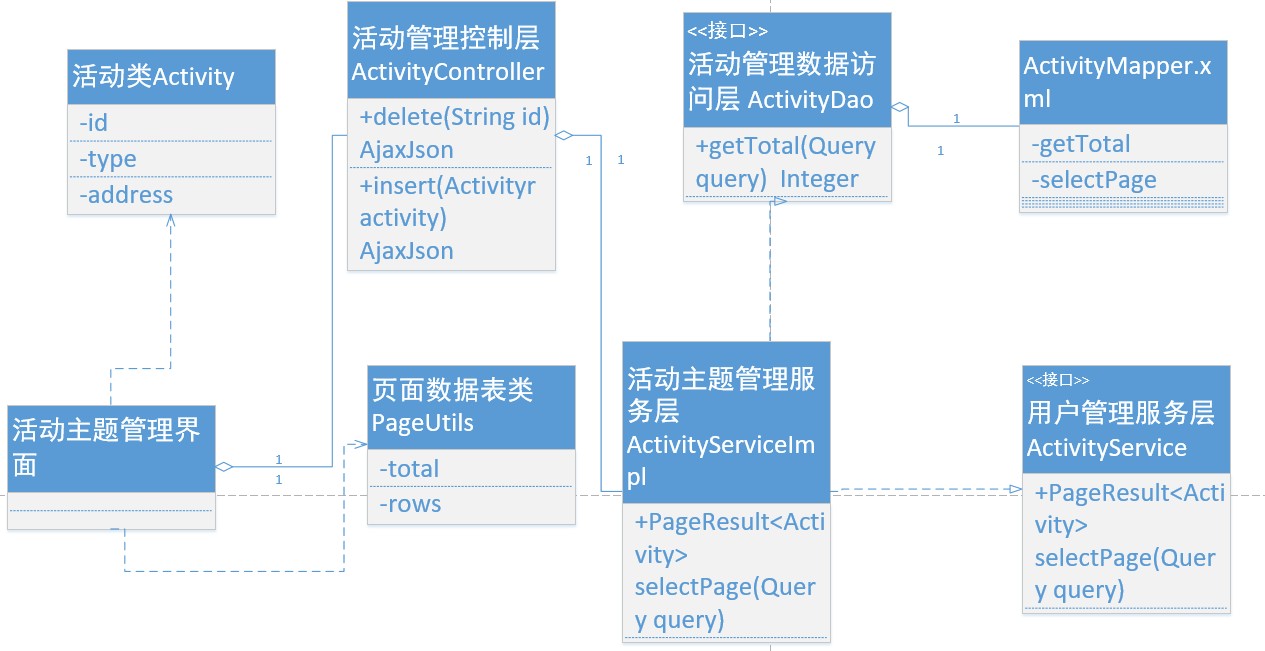
活动主题管理类图如图 6-2 所示：

图 6-2 活动主题管理类图

会议纪要管理类图如图 6-3 所示：

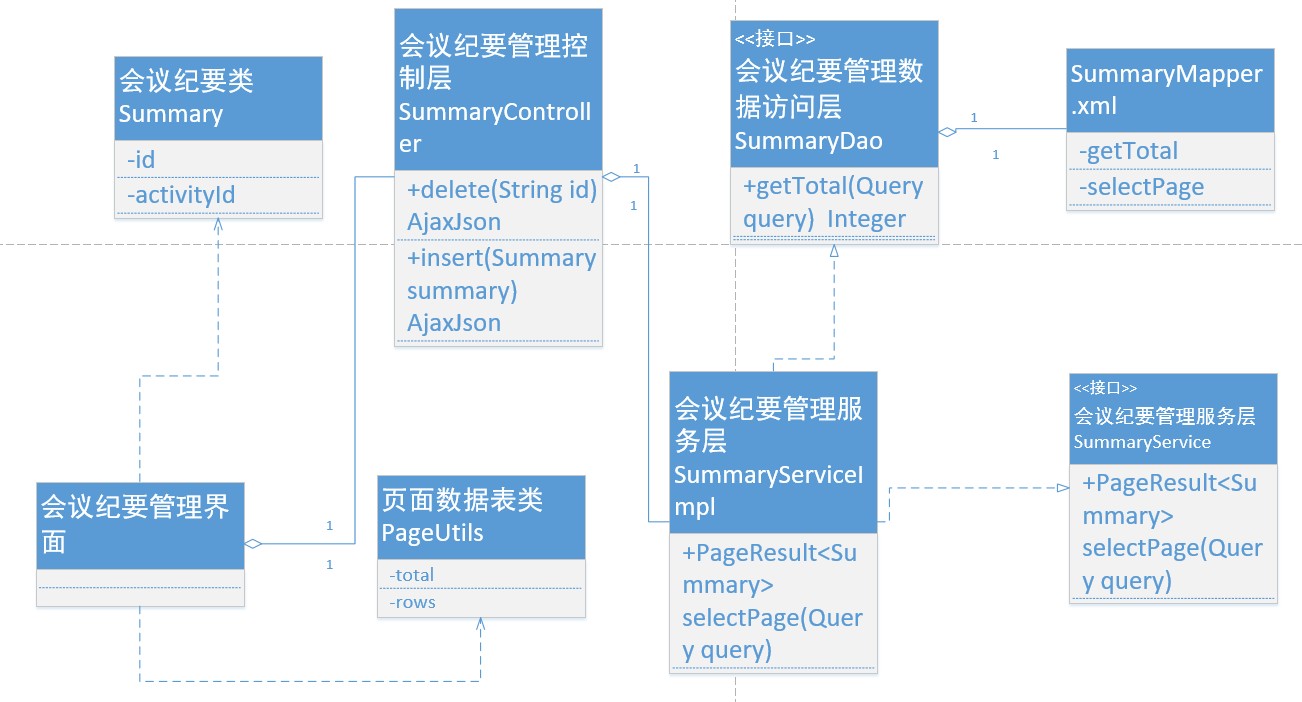


图 6-3 会议纪要管理类图

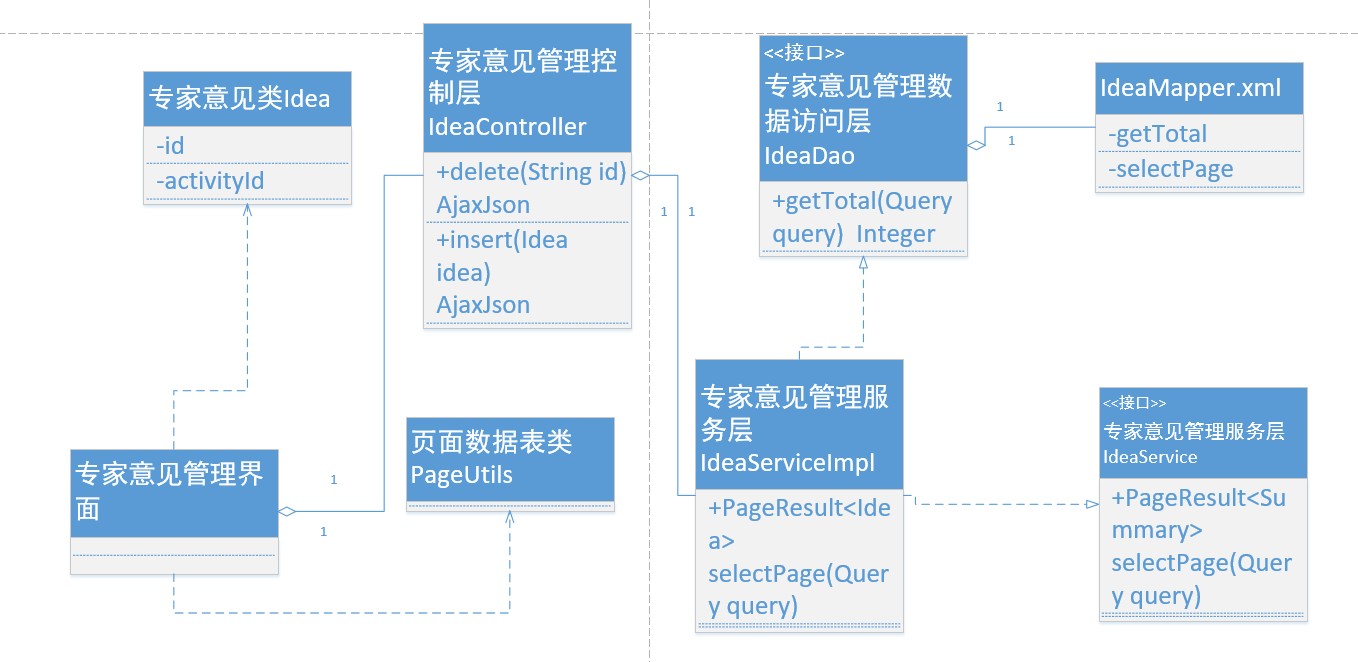
专家意见管理类图如 6-4 所示：

图 6-4 专家意见管理类图

会议资料管理类图如图 6-5 所示：

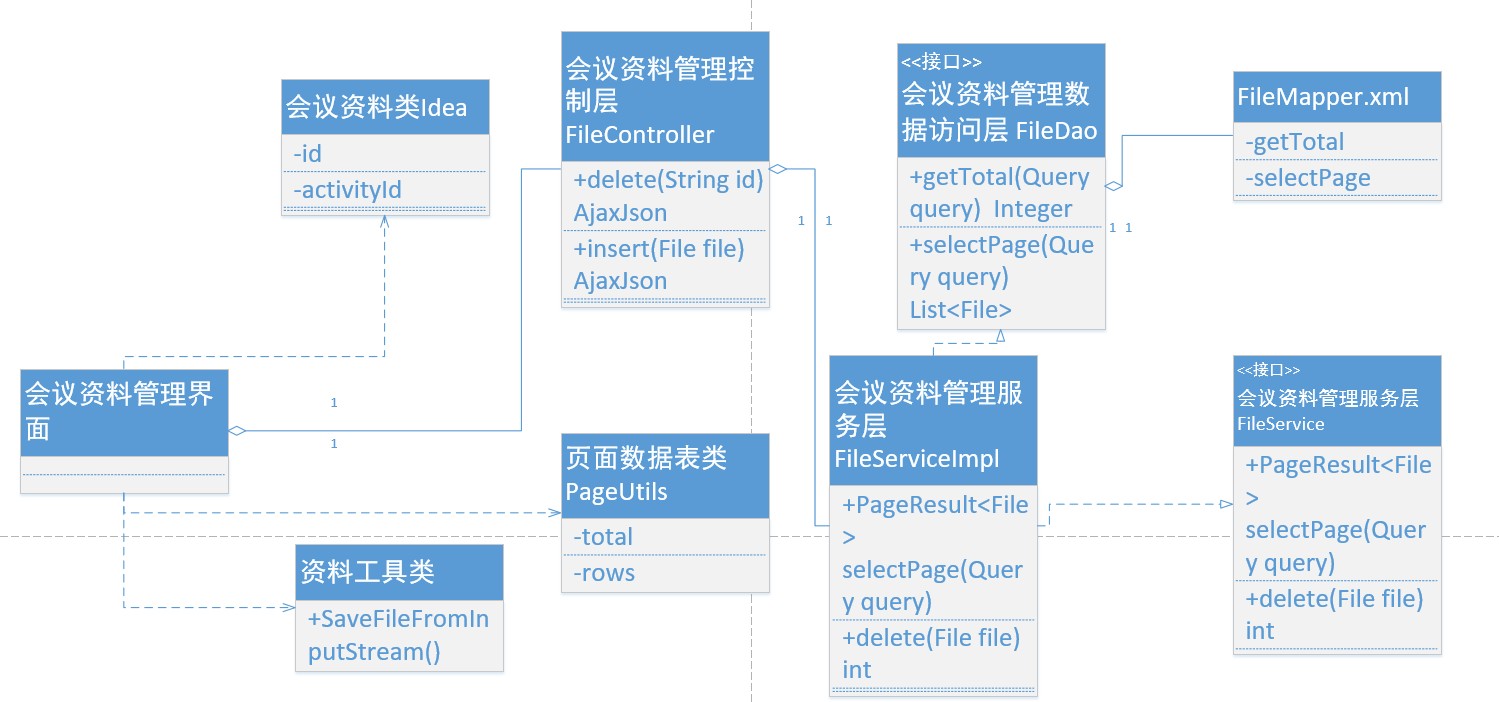


图 6-5 会议资料管理类图

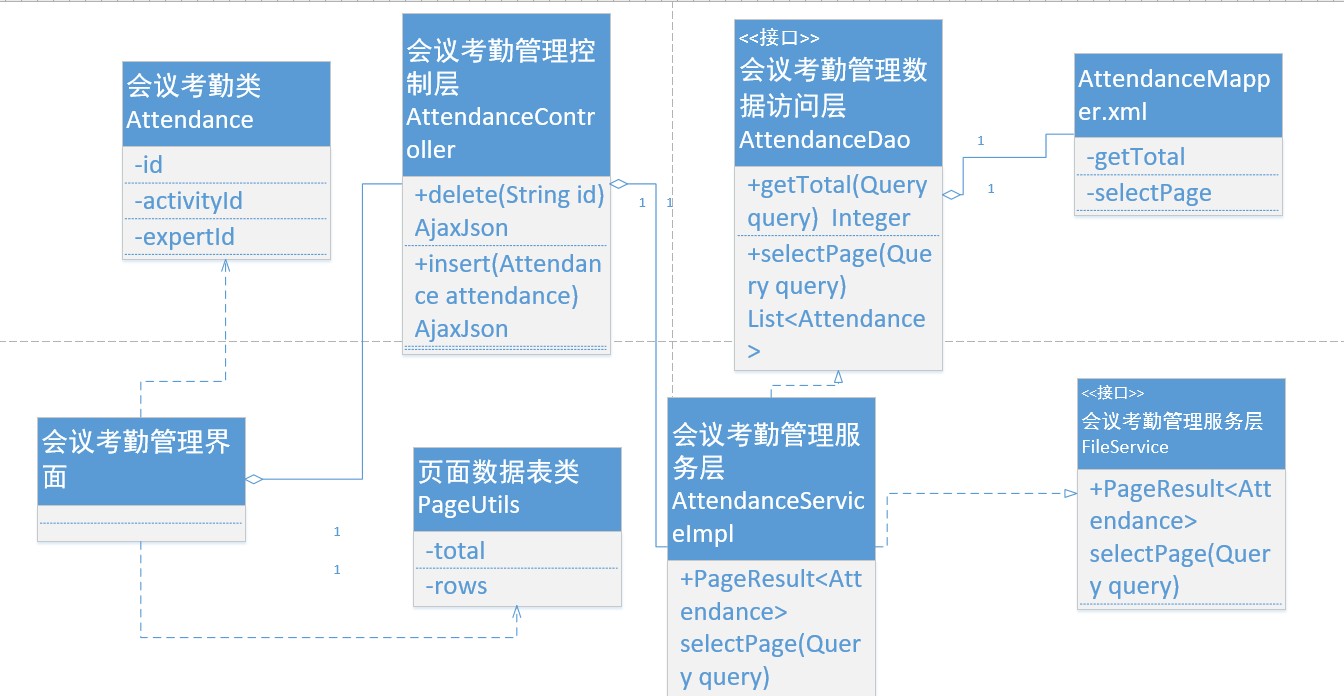
会议考勤管理类图如图 6-6 所示：

图 6-6 会议考勤管理类图

活动主题界面类如表 6-1 所示：

表 6-1 活动主题界面类描述

|  |  |  |
| --- | --- | --- |
| 活动主题界面类 | | |
| 类的描述：负责响应页面发出的活动主题信息获取请求，负责活动信息的维护 | | |
| 操作描述 | Insert(Activity  activity) | 新增活动信息 |

续表 6-1

|  |  |  |
| --- | --- | --- |
| 操作描述 | Insert(Activity  activity) | 新增活动信息 |
| Query(Activity  activity) | 查询活动信息 |
| Delete(Activity  activity) | 删除活动信息 |
| Update(Activity  activity) | 更新活动信息 |
| View(Activity  activity) | 查看活动信息 |

活动实体类描述如表 6-2 所示：

表 6-2 活动实体类描述

|  |  |  |
| --- | --- | --- |
| 活动实体类 | | |
| 类的描述：活动的实体类信息 | | |
| 属性描述 | Id | 活动编号 |
| Type | 活动类型 |
| Theme | 活动主题标题 |
| Information | 活动信息内容 |
| Organization | 活动举办单位 |
| Guest | 活动嘉宾 |
| HoldTime | 活动举办时间 |
| Days | 天数 |
| Address | 活动地址 |
| CreateTime | 创建时间 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

活动主题管理控制层类描述如表 6-3 所示：

表 6-3 活动主题管理控制层类描述

|  |  |  |
| --- | --- | --- |
| 活动主题管理控制层类 | | |
| 类的描述：活动主题管理控制层，处理前台转来的数据 | | |
| 操作描述 | Delete(Activity  activity) | 删除活动主题信息 |
| Insert(Activity  activity) | 新增活动主题信息 |
| Query(Activity  activity) | 查询活动主题信息 |
| View(Activity  activity) | 查看活动主题信息 |

活动主题管理服务层接口描述如表 6-4 所示：

表 6-4 活动主题管理服务层接口描述

|  |  |  |
| --- | --- | --- |
| 活动主题管理服务层接口 | | |
| 类的描述：活动主题管理服务层接口，处理控制层转来的数据 | | |
| 操作描述 | SelectPage(Activity  activity) | 分页查询活动信息 |
| Info(Activity  activity) | 活动内容信息 |
| Delete(Activity  activity) | 删除活动主题信息 |
| Insert(Activity  activity) | 新增活动主题信息 |

活动主题管理服务层类描述如表 6-5 所示：

表 6-5 活动主题管理服务层类描述

|  |  |  |
| --- | --- | --- |
| 活动主题管理服务层类 | | |
| 类的描述：活动主题管理服务层，处理控制层转来的数据 | | |
| 操作描述 | SelectPage(Activity  activity) | 分页查询活动信息 |
| Info(Activity  activity) | 活动内容信息 |

续表 6-5

|  |  |  |
| --- | --- | --- |
| 操作描述 | Delete(Activity  activity) | 删除活动主题信息 |
| Insert(Activity  activity) | 新增活动主题信息 |

活动主题管理数据访问层接口描述如表 6-6 所示：

表 6-6 活动主题管理数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 活动主题管理数据访问层接口 | | |
| 类的描述：活动主题管理数据访问层接口，处理服务层层转来的数据，在数据库中进行操作 | | |
| 操作描述 | GetTotal(Activity  activity) | 获取数据库中的总信息 |
| SelectPage(Activity  activity) | 获取数据库中的分页查询活动信息 |
| Delete(Activity  activity) | 删除数据库中的活动主题信息 |
| Insert(Activity  activity) | 新增数据库中的活动主题信息 |

会议纪要界面类描述如表 6-7 所示：

表 6-7 会议纪要界面类描述

|  |  |  |
| --- | --- | --- |
| 会议纪要界面类 | | |
| 类的描述：负责响应页面发出的活动主题信息获取请求，负责专家意见的维护 | | |
| 操作描述 | Insert(Activity  activity) | 新增会议纪要信息 |
| Query(Activity  activity) | 查询会议纪要信息 |
| Delete(Activity  activity) | 删除会议纪要信息 |
| 操作描述 | Update(Activity  activity) | 更新会议纪要信息 |
| View(Activity  activity) | 查看会议纪要信息 |

会议纪要实体类描述如表 6-8 所示：

表 6-8 会议纪要实体类描述

|  |  |  |
| --- | --- | --- |
| 会议纪要实体类 | | |
| 类的描述：会议纪要实体类信息 | | |
| 属性描述 | Id | 会议纪要编号 |
| Id | 会议纪要编号 |
| ActivityId | 活动主题编号 |
| Title | 会议纪要标题 |
| Stage | 会议纪要阶段 |
| Information | 会议纪要信息 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

会议纪要控制层类描述如表 6-9 所示：

表 6-9 会议纪要控制层类描述

|  |  |  |
| --- | --- | --- |
| 会议纪要控制层类 | | |
| 类的描述：会议纪要控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | Insert(Summary  summary) | 新增会议纪要 |
| Query(Summary summary) | 查询会议纪要信息 |
| Delete(Summary  summary) | 删除会议纪要信息 |

会议纪要服务层类描述如表 6-10 所示：

表 6-10 会议纪要服务层接口描述

|  |  |  |
| --- | --- | --- |
| 会议纪要服务层接口 | | |
| 类的描述：会议纪要服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Summary  summary) | 新增会议纪要 |
| Query(Summary summary) | 查询会议纪要信息 |
| Delete(Summary  summary) | 删除会议纪要信息 |

续表 6-10

|  |  |  |
| --- | --- | --- |
| 操作描述 | Info(Summary summary) | 操作会议纪要内容信息 |

会议纪要服务层描述如表 6-11 所示：

表 6-11 会议纪要服务层描述

|  |  |  |
| --- | --- | --- |
| 会议纪要服务层类 | | |
| 类的描述：会议纪要服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Summary  summary) | 新增会议纪要 |
| Query(Summary summary) | 查询会议纪要信息 |
| Delete(Summary  summary) | 删除会议纪要信息 |
| Info(Summary summary) | 操作会议纪要内容信息 |

会议纪要数据访问层接口描述如表 6-12 所示：

表 6-12 会议纪要数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 会议纪要数据访问层接口 | | |
| 类的描述：会议纪要接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(Summary  summary) | 新增数据库中会议纪要 |
| Query(Summary summary) | 查询数据库中会议纪要信息 |
| Delete(Summary  summary) | 删除数据库中会议纪要信息 |
| Info(Summary summary) | 操作数据库中会议纪要内容信息 |

专家意见界面类描述如表 6-13 所示：

表 6-13 专家意见界面类描述

|  |  |  |
| --- | --- | --- |
| 专家意见界面类 | | |
| 类的描述：负责响应页面发出的专家意见信息获取请求，负责专家意见的维护 | | |
| 操作描述 | Insert(Idea idea) | 新增专家意见信息 |
| Query(Idea idea) | 查询专家意见信息 |
| Delete(Idea idea) | 删除专家意见信息 |

续表 6-13

|  |  |  |
| --- | --- | --- |
| 操作描述 | Update(Idea idea) | 更新专家意见信息 |
| View(Idea idea) | 查看专家意见信息 |

专家意见实体类描述如表 6-14 所示：

表 6-14 专家意见实体类描述

|  |  |  |
| --- | --- | --- |
| 专家意见实体类 | | |
| 类的描述：专家意见实体类信息 | | |
| 属性描述 | Id | 专家意见编号 |
| ActivityId | 活动主题编号 |
| ExpertId | 专家编号 |
| ExpertName | 专家姓名 |
| Information | 专家意见信息 |
| createTime | 专家意见创建时间 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

专家意见控制层类描述如表 6-15 所示：

表 6-15 专家意见控制层类描述

|  |  |  |
| --- | --- | --- |
| 专家意见控制层类 | | |
| 类的描述：专家意见控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | Insert(Idea idea) | 新增专家意见 |
| Query(Idea idea) | 查询专家意见信息 |
| Delete(Idea idea) | 删除专家意见信息 |

专家意见服务层接口描述如表 6-16 所示：

表 6-16 专家意见服务层接口描述

|  |  |  |
| --- | --- | --- |
| 专家意见服务层接口 | | |
| 类的描述：专家意见服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Idea idea) | 新增专家意见 |

续表 6-16

|  |  |  |
| --- | --- | --- |
| 操作描述 | Query(Idea idea) | 查询专家意见信息 |
| Delete(Idea idea) | 删除专家意见信息 |
| Info(Idea idea) | 操作专家意见内容信息 |

专家意见服务层描述如表 6-17 所示：

表 6-17 专家意见服务层描述

|  |  |  |
| --- | --- | --- |
| 专家意见服务层类 | | |
| 类的描述：专家意见服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Idea idea) | 新增专家意见 |
| Query(Idea idea) | 查询专家意见信息 |
| Delete(Idea idea) | 删除专家意见信息 |
| Info(Idea idea) | 操作专家意见内容信息 |

专家意见数据访问层接口描述如表 6-18 所示：

表 6-18 专家意见数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 专家意见数据访问层接口 | | |
| 类的描述：会议纪要接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(Idea idea) | 新增数据库中专家意见 |
| Query(Idea idea) | 查询数据库中专家意见信息 |
| Delete(Idea idea) | 删除数据库中专家意见信息 |
| Info(Idea idea) | 操作数据库中专家意见内容信息 |

会议资料界面类描述如表 6-19 所示：

表 6-19 会议资料界面类描述

|  |  |  |
| --- | --- | --- |
| 会议资料界面类 | | |
| 类的描述：负责响应页面发出的会议资料获取请求，负责会议资料的维护 | | |
| 操作描述 | Insert(File file) | 新增会议资料信息 |
| Upload(File file) | 上传会议资料 |
| Select(File file) | 查询会议资料 |
| Download(File file) | 下载会议资料 |

会议资料实体类描述如表 6-20 所示：

表 6-20 会议资料实体类描述

|  |  |  |
| --- | --- | --- |
| 会议资料实体类 | | |
| 类的描述：会议资料实体类信息 | | |
| 属性描述 | Id | 会议资料编号 |
| ActivityId | 活动主题编号 |
| UserId | 用户编号 |
| ExpertName | 专家姓名 |
| UserName | 用户姓名 |
| FilePath | 文件路径 |
| FileType | 文件类型 |
| CreateTime | 创建时间 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

会议资料控制层类描述如表 6-21 所示：

表 6-21 会议资料控制层类描述

|  |  |  |
| --- | --- | --- |
| 会议资料控制层类 | | |
| 类的描述：专家意见控制层类对前台界面转发来的请求进行操作 | | |
| 操作描述 | Insert(File file) | 新增会议资料 |
| Query(File file) | 查询会议资料信息 |
| Delete(File file) | 删除会议资料信息 |
| Upload(File file) | 上传会议资料 |
| Download(File file) | 下载会议资料 |

会议资料服务层接口描述如表 6-22 所示：

表 6-22 会议资料服务层接口描述

|  |
| --- |
| 会议资料服务层接口 |
| 类的描述：会议资料服务层接口对控制层转发来的请求进行操作 |

续表 6-22

|  |  |  |
| --- | --- | --- |
| 操作描述 | Insert(File file) | 新增会议资料 |
| Query(File file) | 查询会议资料信息 |
| Delete(File file) | 删除会议资料信息 |
| Upload(File file) | 上传会议资料 |
| Download(File file) | 下载会议资料 |

会议资料服务层描述如表 6-23 所示：

表 6-23 会议资料服务层描述

|  |  |  |
| --- | --- | --- |
| 会议资料服务层类 | | |
| 类的描述：会议资料服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(File file) | 新增会议资料 |
| Query(File file) | 查询会议资料文件 |
| Delete(File file) | 删除会议资料文件 |
| Upload(File file) | 上传会议资料 |
| Download(File file) | 下载会议资料 |

会议资料数据访问层接口描述如表 6-24 所示：

表 6-24 会议资料数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 会议资料数据访问层接口 | | |
| 类的描述：会议资料接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(File file) | 新增数据库中会议资料文件 |
| Query(File file) | 查询数据库中会议资料文件 |
| Delete(File file) | 删除数据库中会议资料文件 |
| Upload(File file) | 上传会议资料 |
| Download(File file) | 下载会议资料 |

会议考勤界面类描述如表 6-25 所示：

表 6-25 会议考勤界面类描述

|  |
| --- |
| 会议考勤界面类 |
| 类的描述：负责响应页面发出的会议考勤信息获取请求，负责专家意见的维护 |

续表 6-25

|  |  |  |
| --- | --- | --- |
| 操作描述 | Insert(Attendance  attendance) | 新增会议考勤信息 |
| Query(Attendance  attendance) | 查询会议考勤信息 |
| Delete(Attendance  attendance) | 删除会议考勤信息 |
| Update(Attendance  attendance) | 更新会议考勤信息 |
| View(Attendance  attendance) | 查看会议考勤信息 |

会议考勤实体类描述如表 6-26 所示：

表 6-26 会议考勤实体类描述

|  |  |  |
| --- | --- | --- |
| 会议考勤实体类 | | |
| 类的描述：会议考勤实体类信息 | | |
| 属性描述 | Id | 会议考勤编号 |
| ActivityId | 活动主题编号 |
| ExpertName | 专家姓名 |
| AttendActual | 会议应到人数 |
| Attendshould | 会议实到人数 |
| ExpertId | 专家编号 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

会议考勤控制层类描述如表 6-27 所示：

表 6-27 会议考勤控制层类描述

|  |  |  |
| --- | --- | --- |
| 会议考勤控制层类 | | |
| 类的描述：会议考勤控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | Insert(Attendance  attendance) | 新增会议考勤 |

续表 6-27

|  |  |  |
| --- | --- | --- |
| 操作描述 | Query(Attendance  attendance) | 查询会议考勤信息 |
| Delete(Attendance  attendance) | 删除会议考勤信息 |

会议考勤服务层接口描述如表 6-28 所示：

表 6-28 会议考勤服务层接口描述

|  |  |  |
| --- | --- | --- |
| 会议考勤服务层接口 | | |
| 类的描述：会议考勤服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Attendance  attendance) | 新增会议考勤 |
| Query(Attendance  attendance) | 查询会议考勤信息 |
| Delete(Attendance  attendance) | 删除会议考勤信息 |
| Info() | 操作会议考勤内容信息 |

会议考勤服务层描述如表 6-29 所示：

表 6-29 会议考勤服务层描述

|  |  |  |
| --- | --- | --- |
| 会议考勤服务层类 | | |
| 类的描述：会议考勤服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Attendance  attendance) | 新增会议考勤 |
| Query(Attendance  attendance) | 查询会议考勤信息 |
| Delete(Attendance  attendance) | 删除会议考勤信息 |
| Info() | 操作会议考勤内容信息 |

会议考勤数据访问层接口描述如表 6-30 所示

表 6-30 会议考勤数据访问层接口描述

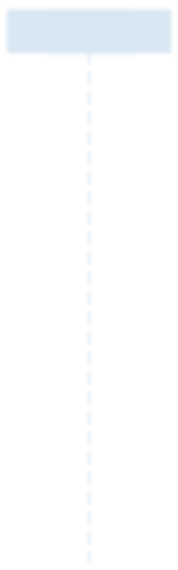
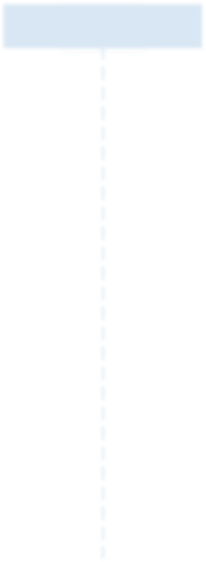
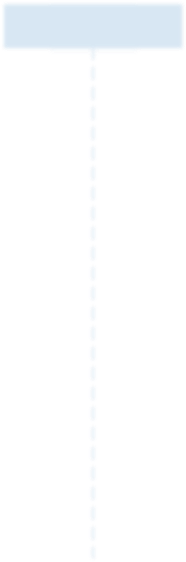
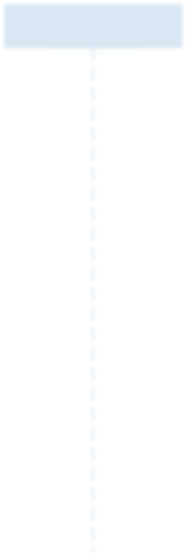
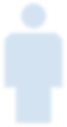
会议考勤数据访问层接口

续表 6-30

|  |  |  |
| --- | --- | --- |
| 类的描述：会议考勤接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(Attendance  attendance) | 新增数据库中会议考勤 |
| Query(Attndance  attendance) | 查询数据库中会议考勤信息 |
| Delete(Attendance  attendance) | 删除数据库中会议考勤信息 |
| Info() | 操作数据库中会议考勤内容信息 |

* + 1. 顺序图设计

会议主题管理顺序图如图 6-7 所示：



**活动管理界面类**

**活动管理控制层**

**活动管理服务层类**

**活动管理数据访问类**

2新增一活动主题，活动信息提交给控

制层

3.向服务层提交活动信息数据

4.在数据库中新增活动数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.选择某一主题，点击管理

13.将查询的数据信息返回前台界面

9.向服务层发送请求数据信

息

12.将查询的数据信息返回控制层

10.在数据库中查询数据信

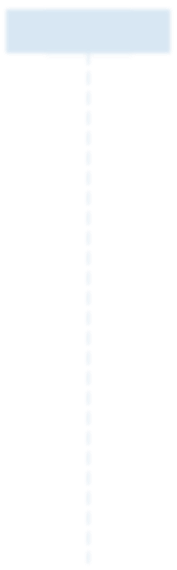
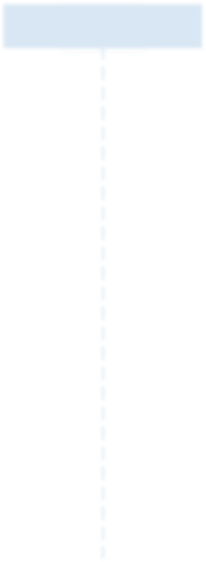
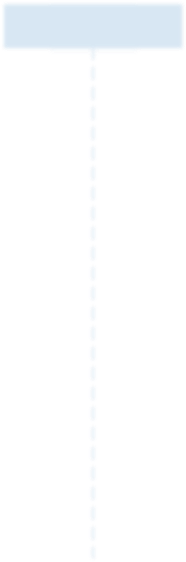
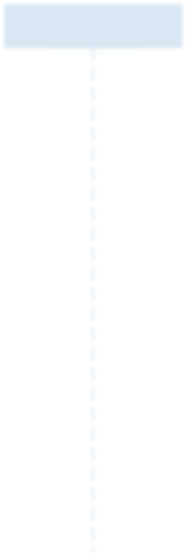
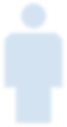
息

11.将查询的数据信息返回

服务层

图 6-7 会议主题管理顺序图

会议纪要管理顺序图如图 6-8 所示：



**会议纪要界面类**

**会议纪要控制层**

**会议纪要服务层类**

**会议纪要数据访问类**

2新增一活动主题，在该主题中新增会

议纪要信息，将数据提交给控制层

3.向服务层提交活动信息数据

4.在数据库中新增会议纪要

数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.在输入框中输入会议纪要

关键字，点击查询

10.在数据库中查询数据信

息

13.将查询的数据信息返回

前台界面

9.向服务层发送查询数据信

息

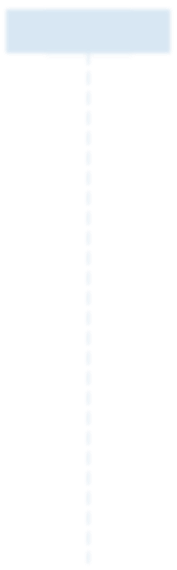
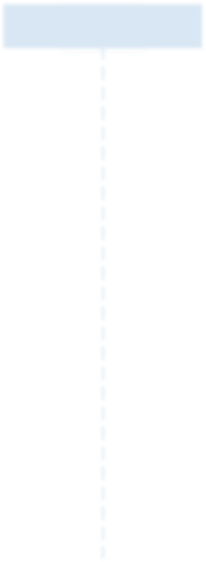
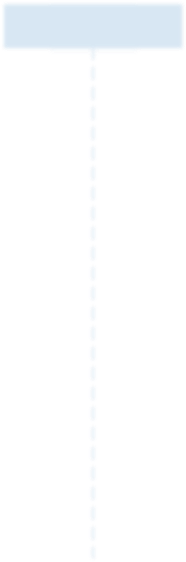
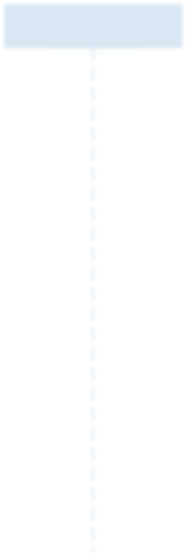
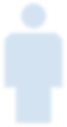
12.将查询的数据信息返回控制层

11.将查询的数据信息返回

服务层

图 6-8 会议纪要管理顺序图

会议资料管理顺序图如图 6-9 所示：



**会议资料界面类**

**会议资料控制层**

**会议资料服务层类**

**会议资料数据访问类**

2新增一会议资料信息，点击上传数据

3.向服务层上传信息数据

4.在数据库中新增会议资料

数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.选择某一会议资料，点击

下载

10.在数据库中查询文件数

据路径

13.将下载的文件返回前台

界面

9.向服务层发送下载文件信

息请求

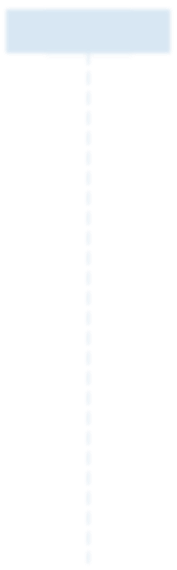
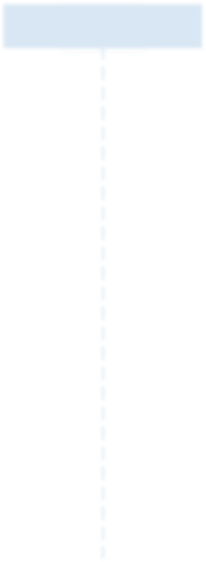
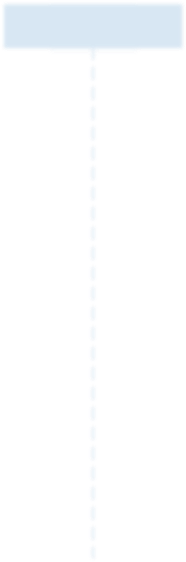
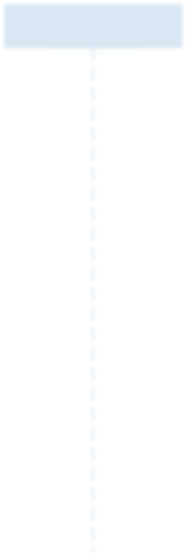
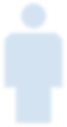
12.将查询的文件路径返回控制层，下载

11.将查询的文件路径返回

服务层

图 6-9 会议资料管理顺序图

专家意见管理顺序图如图 6-10 所示：



**专家意见界面类**

**专家意见控制层**

**专家意见数据访问类**

2新增一活动主题，在该主题中新增专

家意见信息，将数据提交给控制层

3.向服务层提交活动信息数据

4.在数据库中新增专家意见

数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.在输入框中输入专家意见

关键字，点击查询

10.在数据库中查询数据信

息

13.将查询的数据信息返回

前台界面

9.向服务层发送查询数据信

息

12.将查询的数据信息返回控制层

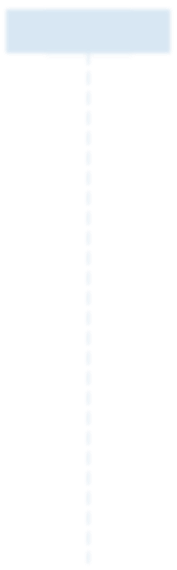
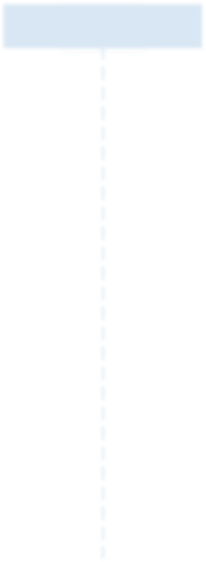
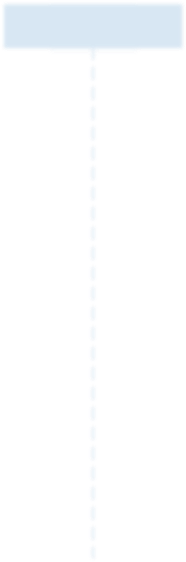
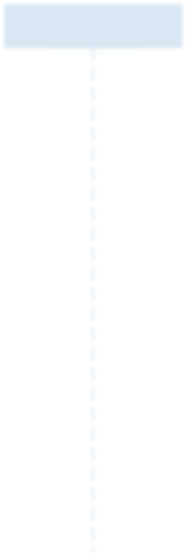
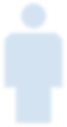
11.将查询的数据信息返回

服务层

**专家意见服务层类**

图 6-10 专家意见管理顺序图

会议考勤管理顺序图如图 6-11 所示：



**会议考勤界面类**

**会议考勤控制层**

**会议考勤数据访问类**

2新增一活动主题，在该主题中新增会

议考勤信息，将数据提交给控制层

3.向服务层提交活动信息数据

4.在数据库中新增会议考勤

数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.在输入框中输入会议考勤

关键字，点击查询

10.在数据库中查询数据信

息

13.将查询的数据信息返回

前台界面

9.向服务层发送查询数据信

息

12.将查询的数据信息返回控制层

11.将查询的数据信息返回

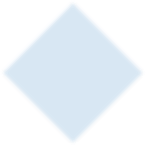
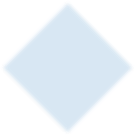
服务层

**会议考勤服务层类**

图 6-11 会议考勤管理顺序图

* + 1. 核心处理流程设计

活动主题新增查询顺序图如图 6-12 所示：



活动管理

新增活动主题

在数据库中查询

否

数据库是否有该活动信息

是

显示该活动信息

显示无信息页面

是

是否继续搜索

否

图 6-12 活动主题新增查询顺序图

会议资料上传与下载顺序图如图 6-13 所示:



下载资料成功

保存

下载

保存或关闭

关闭

秘书，领导或管理

员角色登录系统

活动管理

选择会议主题

管理

会议资料

关闭

上传

上传资料成功

保存

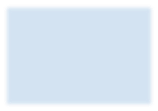
图 6-13 会议资料上传与下载顺序图

保存或关闭

### 成果库管理模块设计

* + 1. 功能结构设计

成果库管理是对专家的成果信息进行维护，包括新增成果信息，查询成果信息，删除成果信息，查看成果信息等。包图如 6-14 所示：



* + 1. 类图设计

图 6-14 成果库管理包图

成果库管理类图如图 6-15 所示：

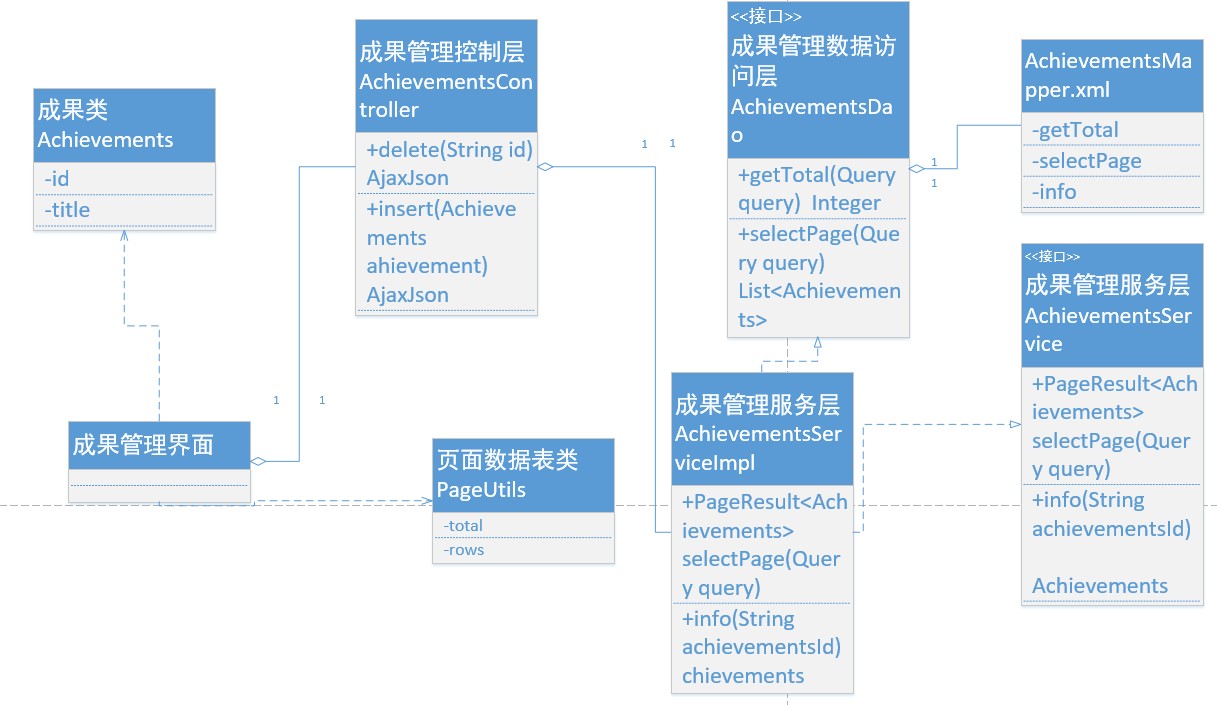


图 6-15 成果库管理类图

成果库界面类描述如表 6-31 所示：

表 6-31 成果库界面类描述

|  |  |  |
| --- | --- | --- |
| 成果库界面类 | | |
| 类的描述：负责响应页面发出的成果信息获取请求，负责成果库的维护 | | |
| 操作描述 | Insert(Achievements  achievements) | 新增成果信息 |
| Insert(Achievements  achievements) | 新增成果信息 |
| Query(Achievements  achievements) | 查询成果信息 |
| Delete(Achievements  achievements) | 删除成果信息 |
| View(Achievements  achievements) | 查看成果信息 |

成果实体类描述如表 6-32 所示：

表 6-32 成果实体类描述

|  |  |  |
| --- | --- | --- |
| 成果实体类 | | |
| 类的描述：成果实体类信息 | | |
| 属性描述 | Id | 成果编号 |
| Title | 成果标题 |
| ExpertId | 专家编号 |
| ExpertName | 成果名称 |
| Information | 成果信息 |
| Information | 成果信息 |
| CreateTime | 创建时间 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

成果库控制层类描述如表 6-33 所示：

表 6-33 成果库控制层类描述

成果库控制层类

续表 6-33

|  |  |  |
| --- | --- | --- |
| 类的描述：成果库控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | Insert(Achievements  achievements) | 新增成果信息 |
| Query(Achievements  achievements) | 查询成果信息 |
| 操作描述 | Delete(Achievements  achievements) | 删除成果信息 |

成果库服务层接口描述如表 6-34 所示：

表 6-34 成果库服务层接口描述

|  |  |  |
| --- | --- | --- |
| 成果库服务层接口 | | |
| 类的描述：成果库服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Achievements  achievements) | 新增成果信息 |
| Query(Achievements  achievements) | 查询成果信息 |
| Delete(Achievements  achievements) | 删除成果信息 |
| Info(Achievements  achievements) | 操作会成果信息 |

成果库服务层描述如表 6-35 所示：

表 6-35 成果库服务层描述

|  |  |  |
| --- | --- | --- |
| 成果库服务层类 | | |
| 类的描述：成果库服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Achievements  achievements) | 新增成果信息 |
| Query(Achievements  achievements) | 查询成果信息 |
| Delete(Achievements  achievements) | 删除成果信息 |
| Info(Achievements  achievements) | 操作成果内容信息 |

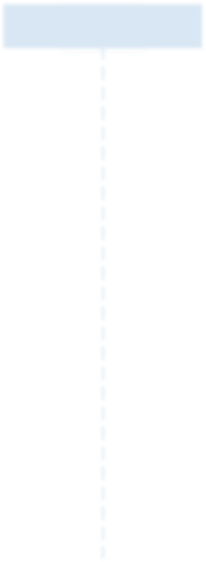
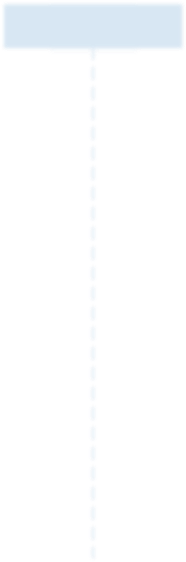
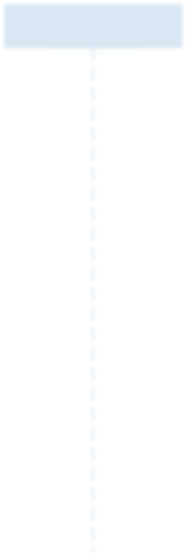
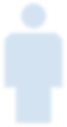
成果库数据访问层接口描述如表 6-36 所示;

表 6-36 成果库数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 成果库数据访问层接口 | | |
| 类的描述：成果库接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(Achievements  achievements) | 新增数据库中成果信息 |
| Query(Achievements  achievements) | 查询数据库中成果信息 |
| Query(Achievements  achievements) | 查询数据库中成果信息 |
| Delete(Achievements  achievements) | 删除数据库中成果信息 |
| Info(Achievements  achievements) | 操作数据库中成果内容信息 |

* + 1. 顺序图设计

成果库管理顺序图如图 6-16 所示：



**成果库界面类**

**成果库服务层类**

**成果库数据访问类**

2新增成果信息，将数据提交给控制层

3.向服务层提交成果信息数据

4.在数据库中新增成果数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.在输入框中输入成果关键

字，点击查询

13.将查询的数据信息返回前台界面

9.向服务层发送查询数据信

息

12.将查询的数据信息返回控制层

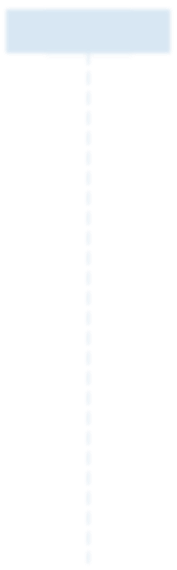
10.在数据库中查询数据信

息

11.将查询的数据信息返回

服务层

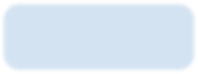
**成果库控制层**



* + 1. 核心处理流程设计

图 6-16 成果库管理顺序图

成果库管理活动图如图 6-17 所示：



成果管理

新增成果

查询成果

生成成果信息

删除成果信息

查看成果信息

成果信息是否存在

是

跳转至详细成果页

面

显示成果信息

图 6-17 成果库管理活动图

### 标签维度管理模块设计

* + 1. 功能结构设计

标签维度管理包括标签维度维护和标签维度设置两个子模块，标签维度维护是对专家的标签信息进行新增和修改操作，标签维度设置是由管理员对专家进行标签设置。包图如 6-18 所示：



* + 1. 类图设计

图 6-18 标签维度管理包图

标签维度类图如图 6-19 所示：

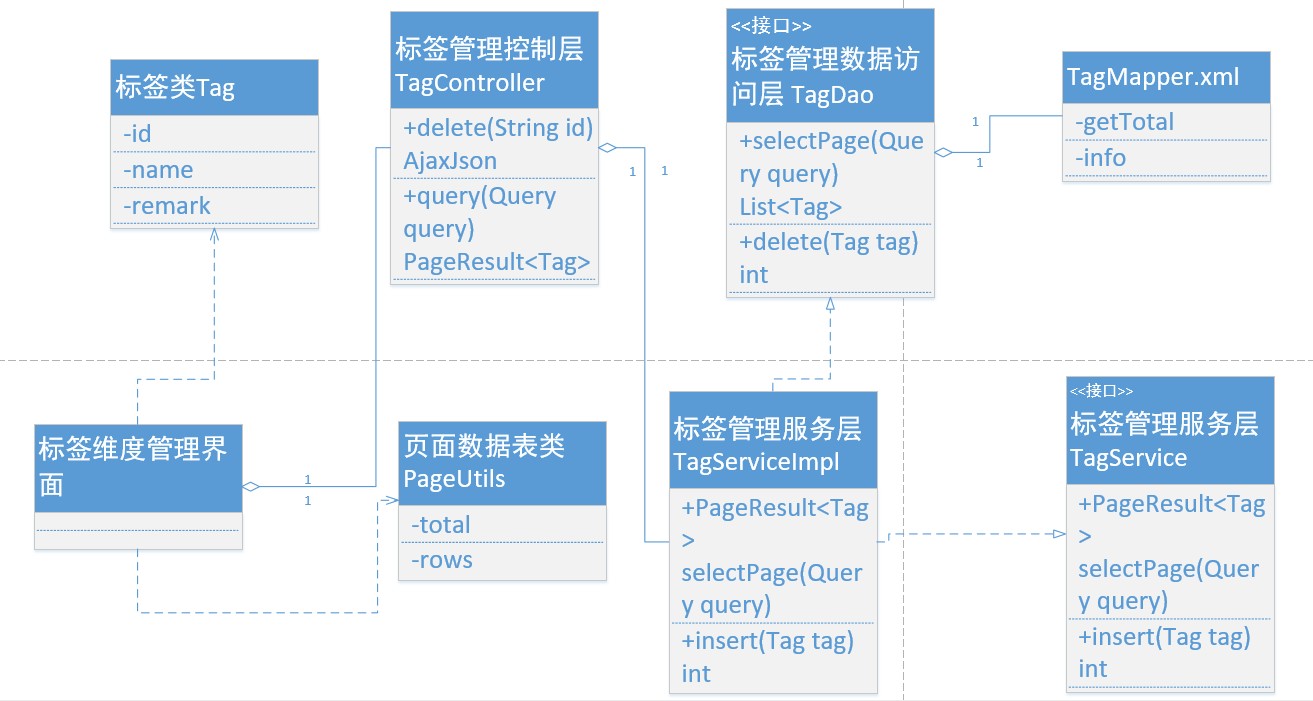


图 6-19 标签维度类图

标签维度界面类描述如表 6-37 所示：

表 6-37 标签维度界面类描述

|  |  |  |
| --- | --- | --- |
| 标签维度界面类 | | |
| 类的描述：负责响应页面发出的标签信息获取请求，负责标签维度的维护 | | |
| 操作描述 | Insert(Tag tag) | 新增标签信息 |
| Query(Tag tag) | 查询标签信息 |
| Delete(Tag tag) | 删除标签信息 |
| Update(Tag tag) | 更新标签信息 |

标签实体类描述如表 6-38 所示：

表 6-38 标签实体类描述

|  |  |  |
| --- | --- | --- |
| 标签实体类 | | |
| 类的描述：标签类信息 | | |
| 属性描述 | Id | 标签编号 |
| Name | 标签名 |
| Remark | 备注 |
| CreateTime | 创建时间 |

续表 6-38

|  |  |  |
| --- | --- | --- |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

标签管理控制层类描述如表 6=39 所示：

表 6-39 标签管理控制层类描述

|  |  |  |
| --- | --- | --- |
| 标签管理控制层类 | | |
| 类的描述：标签管理控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | Insert(Tag tag) | 新增标签信息 |
| Query(Tag tag) | 查询标签信息 |
| Delete(Tag tag) | 删除标签信息 |
| Update(Tag tag) | 修改标签信息 |

标签管理服务层接口描述如表 6-40 所示：

表 6-40 标签管理服务层接口描述

|  |  |  |
| --- | --- | --- |
| 标签管理服务层接口 | | |
| 类的描述：标签管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Tag tag) | 新增标签信息 |
| Query(Tag tag) | 查询标签信息 |
| Delete(Tag tag) | 删除标签信息 |
| Update(Tag tag) | 修改标签信息 |

标签管理业务层描述如表 6-41 所示：

表 6-41 标签管理业务层描述

|  |  |  |
| --- | --- | --- |
| 标签管理业务层类 | | |
| 类的描述：标签管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Tag tag) | 新增标签信息 |
| Query(Tag tag) | 查询标签信息 |
| Delete(Tag tag) | 删除标签信息 |
| Update(Tag tag) | 修改标签信息 |

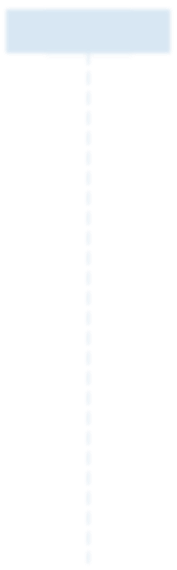
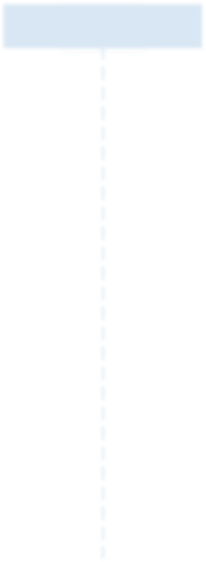
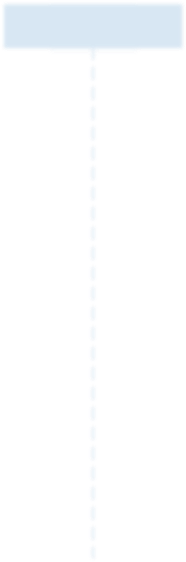
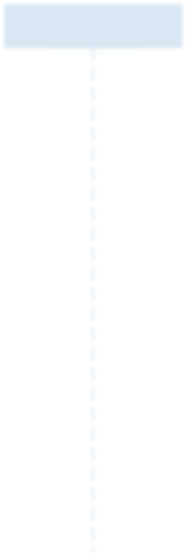
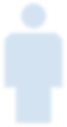
标签管理数据访问层接口描述如表 6-42 所示：

表 6-42 标签管理数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 标签管理数据访问层接口 | | |
| 类的描述：标签管理数据访问接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(Tag tag) | 新增数据库中标签信息 |
| Query(Tag tag) | 查询数据库中标签信息 |
| Delete(Tag tag) | 删除数据库中标签信息 |
| Update(Tag tag) | 修改数据库中标签信息 |

* + 1. 顺序图设计

标签维度管理顺序图如图 6-20 所示：



**标签维度界面类**

**标签管理服务层类**

2新增标签信息，将数据提交给控制层

3.向服务层提交标签数据

4.在数据库中新增标签数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控 5.将数据新增结果返回给服

制层 务层

管理员

8.在输入框中输入标签关键

字，点击查询

10.在数据库中查询数据信

息

13.将查询的数据信息返回

前台界面

9.向服务层发送查询数据信

息

12.将查询的数据信息返回控制层

11.将查询的数据信息返回

服务层

**标签管理数据访问类**

**标签管理控制层**

* + 1. 核心处理流程设计

图 6-20 标签维度管理顺序图

标签维度管理活动图如图 6-21 所示：



标签维度管理

新增标签

查询标签

生成标签信息

删除标签信息

修改成果信息

标签信息是否存在

是

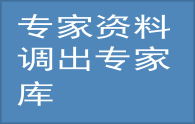
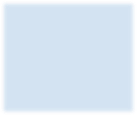
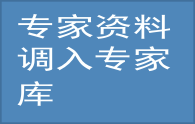
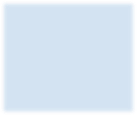
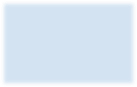
显示标签信息

图 6-21 标签维度管理活动图

### 专家资料管理模块设计

* + 1. 功能结构设计

专家资料管理主要包括了专家资料调入与调出专家库，专家资料维护是对专家资料信息进行新增，修改，删除及展示工作。包图如 6-22 所示：



* + 1. 类图设计

图 6-22 专家资料管理包图

专家资料类图如图 6-23 所示：

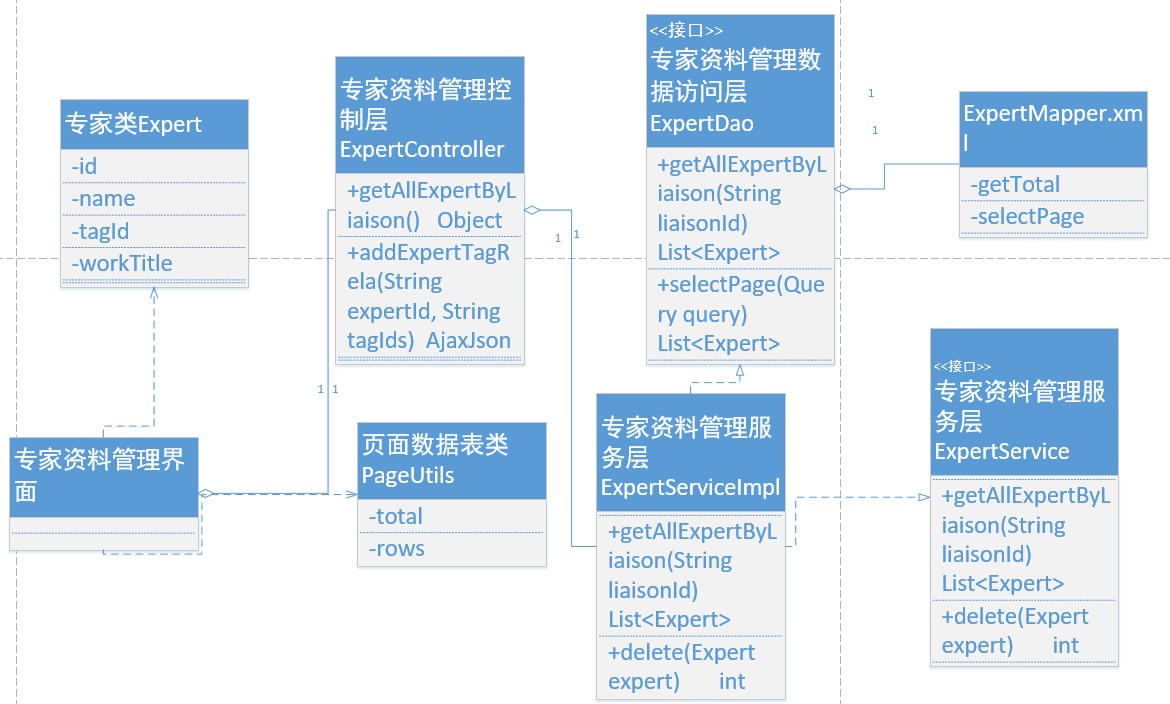


图 6-23 专家资料类图

专家资料管理界面描述如表 6-43 所示：

表 6-43 专家资料管理界面描述

|  |  |  |
| --- | --- | --- |
| 专家资料界面类 | | |
| 类的描述：负责响应页面发出的专家资料获取请求，负责专家资料的维护 | | |
| 操作描述 | Insert(Expert expert) | 新增专家资料信息 |
| 操作描述 | Query(Expert expert) | 查询专家资料信息 |
| Delete(Expert expert) | 删除专家资料信息 |
| Update(Expert expert) | 更新专家资料信息 |
| View(Expert expert) | 查看专家资料信息 |

专家资料实体类描述如表 6-44 所示：

表 6-44 专家资料实体类描述

|  |  |  |
| --- | --- | --- |
| 专家资料实体类 | | |
| 类的描述：专家资料类信息 | | |
| 属性描述 | Id | 标签编号 |

续表 6-44

|  |  |  |
| --- | --- | --- |
| 属性描述 | Name | 标签名 |
| TagId | 备注 |
| TagName | 创建时间 |
| WorkTitle | 职称 |
| Organization | 工作单位 |
| Information | 个人简介 |
| Nation | 国籍 |
| IdNumber | 证件号 |
| Birth | 出生日照 |
| Address | 工作地址 |
| LiaisonId | 联络员编号 |
| LiaisonName | 联络员姓名 |
| filePath | 照片路径 |
| createTime | 创建时间 |
| 操作描述 | Set()/Get() | 各个属性的设置和获取 |
| toString() | 各个属性的字符串读取 |

专家资料管理控制层类描述如表 6-45 所示：

表 6-45 专家资料管理控制层类描述

|  |  |  |
| --- | --- | --- |
| 专家资料管理控制层类 | | |
| 类的描述：专家资料管理控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | Insert(Expert expert) | 新增标签信息 |
| Query(Expert expert) | 查询标签信息 |
| Delete(Expert expert) | 删除标签信息 |
| Update(Expert expert) | 修改标签信息 |
| View(Expert expert) | 查看专家信息 |
| SetLiaison(Expert  expert) | 设置联络员 |

专家资料管理服务层接口描述如表 6-46 所示：

表 6-46 专家资料管理服务层接口描述

|  |  |  |
| --- | --- | --- |
| 专家资料管理控制层类 | | |
| 类的描述：专家资料管理控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | Insert(Expert exert) | 新增标签信息 |
| Query(Expert expert) | 查询标签信息 |
| Delete(Expert expert) | 删除标签信息 |
| Update(Expert expert) | 修改标签信息 |
| getAllTag(Expert  expert) | 获取所有标签 |
| getAllExpertOutOfGroup  (Expert expert) | 获取专家库外专家信息 |
| getAllExpertOutOfLiais  on(Expert expert) | 获取专家库外联络员 |
| getAllExpertByLiaison(  Expert expert) | 根据联络员获取专家信息 |
| addExpertTagRela(Exper  t expert) | 给专家添加标签 |

专家资料管理服务层描述如表 6-47 所示：

表 6-47 专家资料管理服务层描述

|  |  |  |
| --- | --- | --- |
| 专家资料管理服务层类 | | |
| 类的描述：专家资料管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Expert expert) | 新增专家信息 |
| Query(Expert expert) | 查询专家信息 |
| 操作描述 | Delete(Expert expert) | 删除专家信息 |
| Update(Expert expert) | 修改专家信息 |
| getAllTag(Expert  expert) | 获取所有标签 |
| getAllExpertOutOfGroup  (Expert expert) | 获取专家库外专家信息 |
| getAllExpertOutOfLiais  on(Expert expert) | 获取专家库外联络员 |

续表 6-47

|  |  |  |
| --- | --- | --- |
| 操作描述 | getAllExpertByLiaison(  Expert expert) | 根据联络员获取专家信息 |
| addExpertTagRela(Exper  t expert) | 给专家添加标签 |

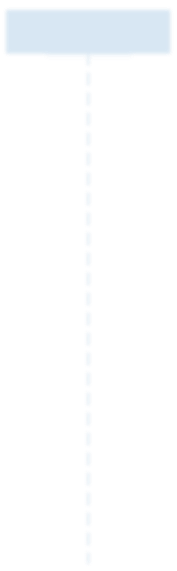
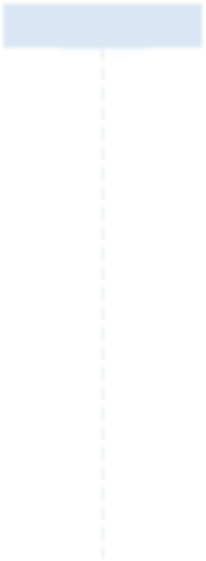
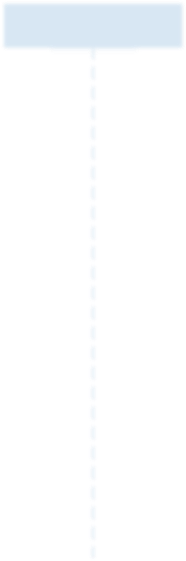
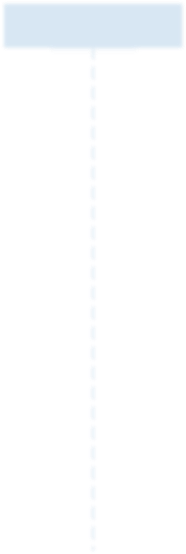
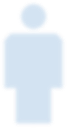
专家资料管理数据访问层接口描述如表 6-48 所示：

表 6-48 专家资料管理数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 专家资料管理数据访问层接口 | | |
| 类的描述：专家资料管理数据访问接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(Expert expert) | 新增数据库中标签信息 |
| Query(Expert expert) | 查询数据库中标签信息 |
| Delete(Expert expert) | 删除数据库中标签信息 |
| Update(Expert expert) | 修改数据库中标签信息 |
| getAllExpertOutOfGroup  (Expert expert) | 在数据库中获取专家库外专家信息 |
| getAllExpertOutOfLiais  on(Expert expert) | 在数据库中获取专家库外联络员 |
| getAllExpertByLiaison(  Expert expert) | 在数据库中根据联络员获取专家信  息 |
| addExpertTagRela(Exper  t expert) | 在数据库中给专家添加标签 |

* + 1. 顺序图设计

专家资料新增与查询顺序图如图 6-24 所示：



**专家资料界面类**

**专家资料管理控**

**制层**

**专家资料管理服务**

**层类**

**专家资料管理数据访**

**问类**

2新增专家信息，将专家资料数据提交

给控制层

3.向服务层提交专家资料数据

4.在数据库中新增专家资料

数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.在输入框中输入专家姓名

关键字，点击查询

10.在数据库中查询数据信

息

13.将查询的数据信息返回

前台界面

9.向服务层发送查询数据信

息

12.将查询的数据信息返回

控制层

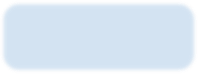
11.将查询的数据信息返回

服务层

* + 1. 核心处理流程设计

图 6-24 专家资料新增与查询顺序图

专家资料管理活动图如图 6-25 所示：



专家资料管理

新增专家资料

查询成果

生成专家资料信息

修改成果信息

删除成果信息

查看专家信息

成果信息是否存在

是

跳转至详细成果页

面

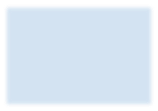
显示成果信息

图 6-25 专家资料管理活动图

### 专家库管理模块设计

* + 1. 功能结构设计

专家库管理模块包括了专家库维护和专家库调入与调出专家两个子模块，专家库维护是对专家库进行新建，查询，删除，查看等操作。调入与调出专家资料是根据系统内的专家资料进行统一管理。包图如 6-26 所示：



* + 1. 类图设计

图 6-26 专家库管理包图

专家库类图如图 6-27 所示：

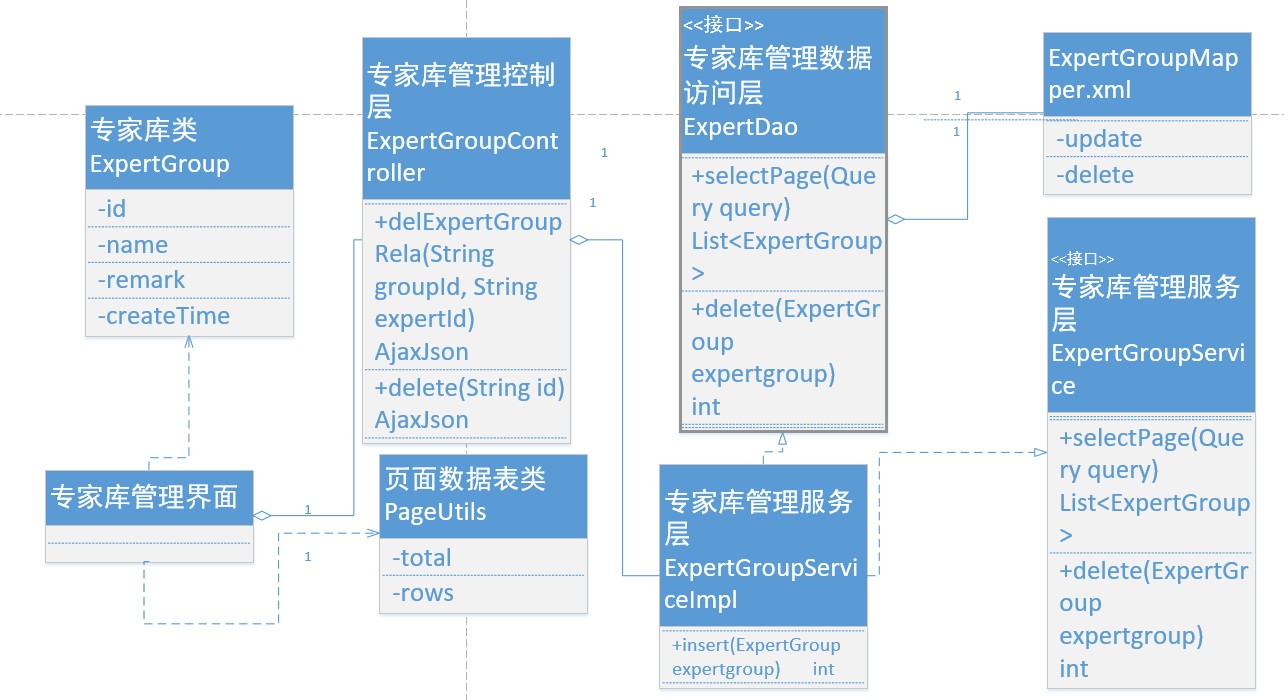


图 6-27 专家库类图

专家库管理界面描述如表 6-49 所示：

表 6-49 专家库管理界面描述

|  |  |  |
| --- | --- | --- |
| 专家库界面类 | | |
| 类的描述：负责响应页面发出的专家库获取请求，负责专家库的维护 | | |
| 操作描述 | Insert(ExpertGroup  expertgroup) | 新增专家库内容信息 |
| Query(ExpertGroup  expertgroup) | 查询专家库内容信息 |
| Delete(ExpertGroup  expertgroup) | 删除专家库内容信息 |
| Update(ExpertGroup  expertgroup) | 更新专家库内容信息 |

专家资料实体类描述如表 6-50 所示：

表 6-50 专家资料实体类描述

|  |  |  |
| --- | --- | --- |
| 专家库实体类 | | |
| 类的描述：专家库类信息 | | |
| 属性描述 | Id | 专家库编号 |
| Name | 专家库名 |
| Remark | 备注 |
| createTime | 创建时间 |
| 操作描述 | Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |
| toString() | 属性的字符串读取 |

专家库管理控制层类描述如表 6-51 所示：

表 6-51 专家库管理控制层类描述

|  |  |  |
| --- | --- | --- |
| 专家库管理控制层类 | | |
| 类的描述：专家库管理控制层类对前台界面转发来的数据进行操作 | | |
| 操作描述 | delExpertGroupRela(Exp ertGroup expertgroup) | 删除专家库中专家信息 |
| addExpertGroupRela(Exp  ertGroup expertgroup) | 添加专家库中专家信息 |

续表 6-51

|  |  |  |
| --- | --- | --- |
| 操作描述 | Delete(ExpertGroup  expertgroup) | 删除专家库中专家信息 |
| Update(ExpertGroup  expertgroup) | 修改专家库中专家信息 |
| Inset(ExpertGroup  expertgroup) | 新增专家库中专家信息 |
| Query(ExpertGroup  expertgroup) | 查询专家库信息 |

专家库管理服务层接口描述如表 6-52 所示：

表 6-52 专家库管理服务层接口描述

|  |  |  |
| --- | --- | --- |
| 专家库管理服务层接口 | | |
| 类的描述：专家库管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(ExpertGroup  expertgroup t) | 新增专家库中专家信息 |
| SelectPage(ExpertGroup expertgroup) | 查询专家库页面信息 |
| Delete(ExpertGroup expertgroup) | 删除专家库中专家信息 |
| Update(ExpertGroup  expertgroup) | 修改专家库中专家信息 |
| Inset(ExpertGroup expertgroup) | 新增专家库中专家信息 |
| Query(ExpertGroup  expertgroup) | 查询专家库中专家信息 |

专家库管理服务层描述如表 6-53 所示：

表 6-53 专家库管理服务层描述

|  |  |  |
| --- | --- | --- |
| 专家库管理服务层类 | | |
| 类的描述：专家库管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(ExpertGroup  expertgroup) | 新增专家库中专家信息 |

续表 6-53

|  |  |  |
| --- | --- | --- |
| 操作描述 | SelectPage(ExpertGroup expertgroup) | 查询专家库页面信息 |
| Delete(ExpertGroup  expertgroup) | 删除专家库中专家信息 |
| Update(ExpertGroup  expertgroup) | 修改专家库中专家信息 |

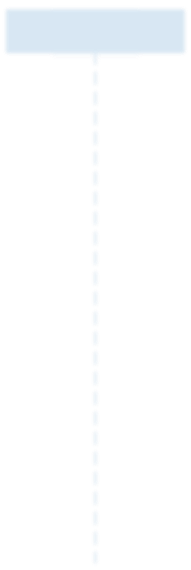
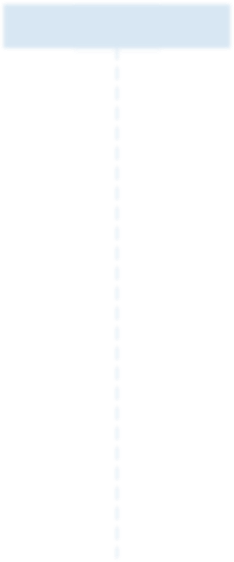
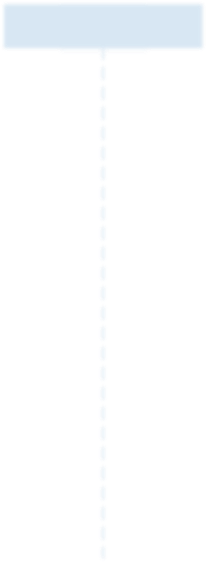
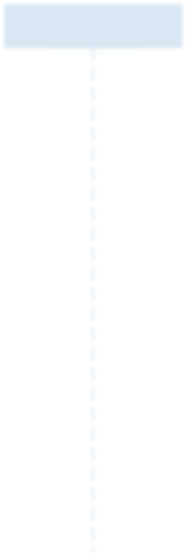
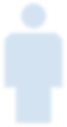
专家库管理数据访问层接口描述如表 6-54 所示：

表 6-54 专家库管理数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 专家库管理数据访问层接口 | | |
| 类的描述：专家库管理数据访问接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(ExpertGroup  expertgroup) | 新增数据库中专家库内容信息 |
| Query(ExpertGroup  expertgroup) | 查询数据库中专家库内容信息 |
| Delete(ExpertGroup  expertgroup) | 删除数据库中专家库内容信息 |
| Update(ExpertGroup  expertgroup) | 修改数据库中专家库内容信息 |
| selectPage(ExpertGroup  expertgroup) | 在数据库中查找出专家库页面 |
| getAllExpertOutOfGroup  (String groupId) | 在数据库中获取专家库外专家信息 |
| getAllExpertOutOfLiais  on() | 在数据库中根据专家库之外的联络  员获取专家信息 |
| info(String expertId) | 在数据库中操作专家库信息 |
| getAllExpertByLiaison(  String liaisonId) | 在数据库中根据联络员获取所有专  家 |

* + 1. 顺序图设计

专家库新增与调入专家顺序图如图 6-28 所示：



**专家库界面类**

**专家库管理控制层**

**专家库管理服务层类**

**专家库管理数据访问类**

2新增专家库信息，将专家库数据提交

给控制层

3.向服务层提交专家库数据

4.在数据库中新增专家库数

据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

管理员

8.专家库内调入专家信息

13.将调入的数据信息返回前台界面

9.向服务层发送调入的专家

信息

12.将调入的数据信息返回控制层

10.在专家库中往数据库新

增专家信息

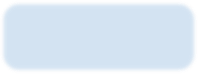
11.将调入的专家库专家数

据信息返回服务层

* + 1. 核心处理流程设计

图 6-28 专家库新增与调入专家顺序图

专家库管理活动图如图 6-29 所示：



专家库管理

新增专家库

查询专家库

生成专家库信息

调入专家信息

调出专家信息

查看专家库信息

专家库信息是否存在

是

跳转至详细专家库

页面

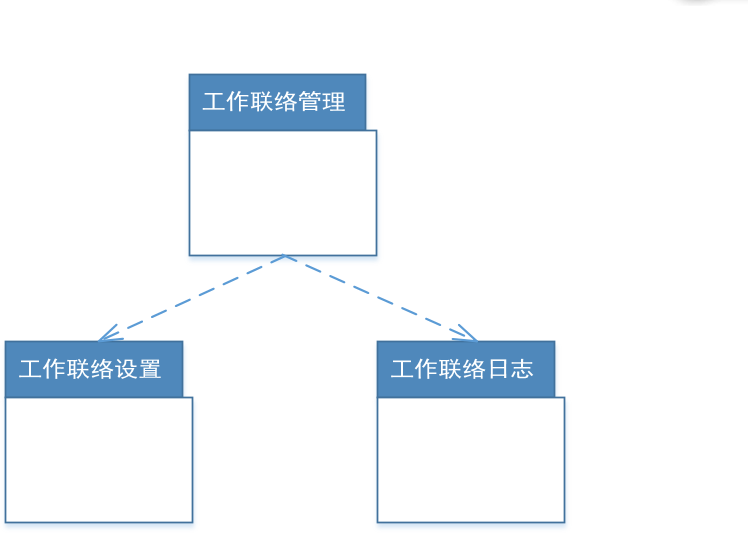
显示专家库信息

图 6-29 专家库管理活动图

### 工作联络管理模块设计

* + 1. 功能结构设计

工作联络管理模块包括工作联络设置和工作联络日志子模块，工作联络设置模块是设置智库专家与对应的联络员，工作联络日志子模块是联络员联络专家的记录信息。包图如 6-30 所示：



* + 1. 类图设计

图 6-30 工作联络管理包图

工作联络管理类图如图 6-31 所示：

图 6-31 工作联络管理类图

工作联络管理界面描述如表 6-55 所示：

表 6-55 工作联络管理界面描述

工作联络管理界面类

续表 6-55

|  |  |  |
| --- | --- | --- |
| 类的描述：负责响应页面发出的工作联络管理获取请求，负责工作联络的维护 | | |
| 操作描述 | Insert(Liaison  liaision) | 设置专家与联络员 |
| Query(Liaison  liaision) | 查询工作联络信息 |
| Delete(Liaison  liaision) | 删除工作联络信息 |
| Update(Liaison  liaision) | 更新工作联络信息 |
| View(Liaison liaision) | 查看工作联络信息 |

工作联络实体类描述如表 6-56 所示：

表 6-56 工作联络实体类描述

|  |  |  |
| --- | --- | --- |
| 工作联络实体类 | | |
| 类的描述：工作联络类信息 | | |
| 属性描述 | Id | 工作联络编号 |
| ExpertId | 专家编号 |
| LiaisonId | 联络员编号 |
| ExpertName | 专家姓名 |
| createTime | 创建时间 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

工作联络记录实体类描述如表 6-57 所示：

表 6-57 工作联络记录实体类描述

|  |  |  |
| --- | --- | --- |
| 工作联络实体类 | | |
| 类的描述：工作联络类信息 | | |
| 属性描述 | Id | 工作联络编号 |
| ExpertId | 专家编号 |
| LiaisonId | 联络员编号 |

续表 6-57

|  |  |  |
| --- | --- | --- |
| 属性描述 | ExpertName | 专家姓名 |
| createTime | 创建时间 |
| Method | 联络方式 |
| Method | 联络方式 |
| Content | 联络内容 |
| Title | 联络标题 |
| LiaisonName | 联络员姓名 |
| 操作描述 | toString() | 属性的字符串读取 |
| Set() | 各个属性的设置 |
| Get() | 各个属性的获取 |

工作联络管理控制层类描述如表 6-58 所示：

表 6-58 工作联络管理控制层类描述

|  |  |  |
| --- | --- | --- |
| 工作联络管理服务层接口 | | |
| 类的描述：工作联络管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Liaison  liaision) | 新增联络员与专家的联络信息 |
| SelectPage(Liaison  liaision) | 查询所有工作联络信息页数据 |
| Delete(Liaison  liaision) | 删除工作联络信息 |

工作联络管理服务层接口描述如表 6-59 所示：

表 6-59 工作联络管理服务层接口描述

|  |  |  |
| --- | --- | --- |
| 工作联络管理服务层接口 | | |
| 类的描述：工作联络管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Liaison  liaision) | 新增联络员与专家的联络信息 |
| SelectPage(Liaison  liaision) | 查询所有工作联络信息页数据 |
| Delete(Liaison  liaision) | 删除工作联络信息 |

续表 6-59

|  |  |  |
| --- | --- | --- |
| 操作描述 | GetAll() | 获取所有工作联络信息 |
| GetTotal() | 获取总页数 |

工作联络管理服务层描述如表 6-60 所示：

表 6-60 工作联络管理服务层描述

|  |  |  |
| --- | --- | --- |
| 工作联络管理服务层类 | | |
| 类的描述：工作联络管理服务层接口对控制层转发来的数据进行操作 | | |
| 操作描述 | Insert(Liaison  liaision) | 新增联络员与专家的联络信息 |
| SelectPage(Liaison  liaision) | 查询所有工作联络信息页数据 |
| 操作描述 | Delete(Liaison  liaision) | 删除工作联络信息 |
| GetAll() | 获取所有工作联络信息 |
| GetTotal() | 获取总页数 |

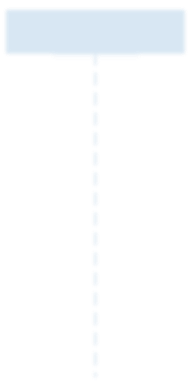
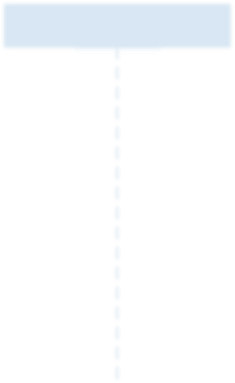
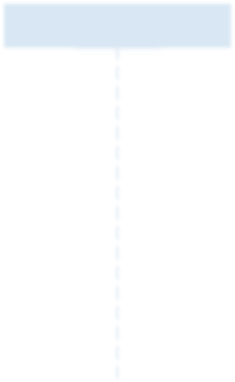
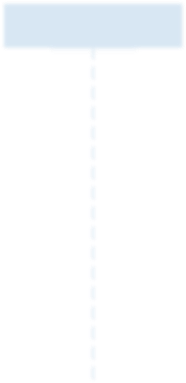
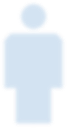
专家库管理数据访问层接口描述如表 6-61 所示：

表 6-61 专家库管理数据访问层接口描述

|  |  |  |
| --- | --- | --- |
| 工作联络管理数据访问层接口 | | |
| 类的描述工作联络管理数据访问接口对服务层转发来的数据进行在数据库中进行操作 | | |
| 操作描述 | Insert(Liaison  liaision) | 新增数据库中工作联络信息 |
| 操作描述 | Delete(Liaison  liaision) | 删除数据库中工作联络信息 |
| getTotal(Query query) | 获取数据库中所有工作联络信息 |
| getAll() | 在数据库中操作工作联络信息 |
| SelectPage(Liaison  liaision) | 在数据库中查询工作联络数据页信息 |

* + 1. 顺序图设计

联络员-专家设置顺序图如 6-32 所示：



**工作联络设置控制**

**层**

2设置联络员与专家，将数据提交控制

层

3.向服务层提交联络员与专家数

据

4.在数据库中新增联络员与

专家数据

1.访问页面

7.将数据新增结果返回给用

户操作界面

6.将数据新增结果返回给控

制层

5.将数据新增结果返回给服

务层

联络员

**工作联络设置数据访问**

**类**

**工作联络设置服务层类**

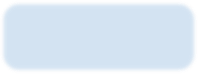
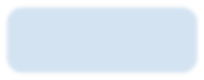
**工作联络设置界面**

**类**

* + 1. 核心处理流程设计

图 6-32 联络员-专家设置顺序图

工作联络记录活动图如图 6-33 所示：



工作联络记录

新增工作联络记录

查询联络记录

生成工作联络记录

信息

修改联络记录

删除联络记录

查看专联络记录信

息

联络记录信息是否存在

是

跳转至详细联络记

录页面

显示联络记录信息

图 6-33 工作联络记录活动图

# 编码

### 代码实现与核心算法

* + 1. 日志管理模块

本模块对所有用户的日志进行查询，将要查找的用户名在数据库中日志表进行查找， 将查找结果用数据表类进行封装，返回给前台。

|  |
| --- |
| @Controller @RequestMapping("/log")  public class LogController extends BaseController {  private static final Logger *logger* = LoggerFactory.*getLogger*(LogController.class); @Autowired  private LogService logService;  @RequestMapping(value = "/query", method = RequestMethod.*GET*) public String query(Model model) {  return "log";  }  @ResponseBody  @RequestMapping(value = "/query", method = RequestMethod.*POST*) public PageResult<Log> query(Query query) {  PageResult<Log> pageResult = logService.selectPage(query); return pageResult;  }  } |

* + 1. 活动管理模块

本模块是对用户新增和删除活动进行操作，新增活动是将前台 AjaxJson 传来的活动信息字符串进行解析封装，将数据发送给服务层，服务层对数据再进行判断发给数据持久层，写入数据库，将写入数据库的结果进行原路返回，最后将 AjaxJson 数据返回给前台页面。删除活动是将删除活动的 id 发送给控制层，控制层将数据发送给服务层，服务层将要删除的数据信息传输给数据持久层，通过 SQL 语句进行删除，包括此活动的所有子信息全部删除。

|  |
| --- |
| @SuppressWarnings("rawtypes") @ResponseBody  @RequestMapping(value = "/insert", method = RequestMethod.*POST*) public AjaxJson insert(Activity activity) {  try {  activityService.insert(activity); Log log=new Log(); |

|  |
| --- |
| log.setUserId(getUser().getId()); log.setContent("添加活动"); log.setModular("活动管理"); logService.insert(log);  } catch (Exception e) {  *logger*.error(e.getMessage(), e);  return new AjaxJson(false, e.getMessage());  }  return new AjaxJson(true, "成功");  }  @SuppressWarnings("rawtypes") @ResponseBody  @RequestMapping(value = "/delete", method = RequestMethod.*POST*) public AjaxJson delete(String id) {  if (StringUtils.*isBlank*(id)) {  return new AjaxJson(false, "id 不能为空");  }  Activity activity = new Activity(); activity.setId(id);  try {  activityService.delete(activity); Log log=new Log(); log.setUserId(getUser().getId()); log.setContent("删除活动"); log.setModular("活动管理"); logService.insert(log);  } catch (Exception e) {  *logger*.error(e.getMessage(), e); return new AjaxJson(false, "失败");  }  return new AjaxJson(true, "成功");  } |

* + 1. 专家资料管理模块

本模块是对所有专家的联络员进行操作，通过前台传来的 AjaxJson 字符串数据对用户进行判断，获取所有专家对应的联络员信息，方便后续对工作联络设置中对专家-联络员进行判断设置。对专家进行标签设置，先对前台传来的 AjaxJson 字符串进行分割，对专家标签进行封装，然后添加到数据库中。

|  |
| --- |
| @ResponseBody |

|  |
| --- |
| @RequestMapping(value = "/getAllExpertByLiaison", method = RequestMethod.*POST*) public Object getAllExpertByLiaison() {  Map<String, Object> result = new HashMap<String, Object>(); User user=getUser(); if(user!=null&&"3".equals(user.getType())){  List<Expert> expertList = expertService.getAllExpertByLiaison(user.getId()); result.put("expertList", expertList);  }else {  result.put("expertList", null);  }  result.put("success", true); return result;  }  @SuppressWarnings("rawtypes") @ResponseBody  @RequestMapping(value = "/addExpertTagRela", method = RequestMethod.*POST*) public AjaxJson addExpertTagRela(String expertId, String tagIds) {  String[] ids=tagIds.split(","); expertTagRelaService.deleteByExpert(expertId); try {  for(String id:ids) {  ExpertTagRela etr=new ExpertTagRela(); etr.setTagId(id); etr.setExpertId(expertId); expertTagRelaService.insert(etr);  }  } catch (Exception e) { e.printStackTrace();  return new AjaxJson(false, "失败");  }  return new AjaxJson(true, "成功");  } |

* + 1. 工作联络管理模块

对工作联络中修改联络记录和新增联络记录操作，通过前台 AjaxJson 传来的工作联络信息字符串进行解析之后，对数据进行判断，字符不能为空之后，接着对联络员与专家的匹配信息在数据库中查询判断，最后进行修改联络信息操作，对操作信息发送给业务层，业务层进行判断之后，发给数据持久层进行数据 SQL 执行操作。最后将结果原路返回给控制层，控制层将结果进行封装之后，又通过 AjaxJson 返回给前台页面。

|  |
| --- |
| @SuppressWarnings("rawtypes") @ResponseBody  @RequestMapping(value = "/update", method = RequestMethod.*POST*)  public AjaxJson update(LiaisonRecord liaisonRecord) { |

|  |
| --- |
| if (liaisonRecord == null) {  return new AjaxJson(false, "参数错误");  }  if (StringUtils.*isBlank*(liaisonRecord.getId())) { return new AjaxJson(false, "id为空");  }  User user=getUser(); if(user!=null&&liaisonRecord.getLiaisonId().equals(user.getId())){  liaisonRecord.setLiaisonId(user.getId());  }else {  return new AjaxJson(false, "当前登录人不是该数据的联络员，操作失败");  }  try {  liaisonRecordService.update(liaisonRecord); Log log=new Log(); log.setUserId(getUser().getId()); log.setContent("修改联络记录"); log.setModular("联络记录管理"); logService.insert(log);  } catch (Exception e) { *logger*.error(e.getMessage(), e); return new AjaxJson(false, "失败");  }  return new AjaxJson(true, "成功");  } |

### 代码优化分析

使用 SSM 框架进行开发,对代码优化起到了很大帮助。

在 Spring 框架中立于控制反转（IOC）特性，对象之间的相互依赖完全转移到 Spring中，由 Spring 统一的管理各个对象，将对象之间的依赖关系完全交给了 Spring 控制， 从而便利了分离过程并简化了开发过程，利用 Spring 的面向切面（AOP）特性，对重复的功能进行集中处理，实现数据事务，代码日志等操作控制，此外，它还可以整合其他的优秀的开源框架，例如 hibernate 框架等。

Spring MVC 框架是 Spring 家族的另一个框架，它利用 MVC 的设计思想，能够快速实现 java web 技术，能够实现灵活的对数据进行验证。它主要分为三层，分别由控制器层，业务层，数据持久层组成，控制器层是对前台页面转来的字符串数据进行解析。业务层是对控制器层转来的封装好的数据进行逻辑处理，进行再次封装。数据持久层是对业务层的数据进行处理，筛选出想要的数据，进行数据库事务操作。

MyBatis 框架是对数据库进行操作的框架，它不像 jdbc 技术那样需要繁琐的数据库建立，断开操作。只需在 xml 文件中配置好相应的事务管理工作，就能够在 xml 文件或

者是 mapper 文件中进行 SQL 语句的封装，还支持动态 SQL，能够减轻开发人员的工作量， 减少编写大量重复的 SQL 语句，还提高了数据的安全性。

为了防止认为输入后台网址，进入后台页面进行操作，本系统设置了登录权限验证拦截器，集成了权限认证，需要匹配数据库中的 URL，用户访问验证是否具有权限，如果没有，则重定向到登录页面。优化代码如下：

|  |
| --- |
| @Override  public boolean preHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o) throws Exception {  String path = httpServletRequest.getServletPath();  if (!StringUtils.*isEmpty*(httpServletRequest.getQueryString())) { path = path + "?" + httpServletRequest.getQueryString();  }  HttpSession session = httpServletRequest.getSession(true); if (session.getAttribute(SessionKey.*sessionUser*) == null) { if (returnRelogin(httpServletResponse, o)) return false; httpServletResponse.sendRedirect("/account/login");  return false;  }  if (path.matches(*exclue*)) { return true;  }  User currentUser = (User) session.getAttribute(SessionKey.*sessionUser*); if (currentUser.getUsername().equalsIgnoreCase(Cons.*Admin*)) {  return true;  }  if(o.getClass().isAssignableFrom(HandlerMethod.class)&&((HandlerMethod) o).getMethod().isAnnotationPresent(BaseNoNeedPermission.class)) {  return true;  }  if (returnRelogin(httpServletResponse, o)) return false; httpServletResponse.sendRedirect("/account/login");  return false;  }  private boolean returnRelogin(HttpServletResponse httpServletResponse, Object o) throws IOException {  if (o instanceof HandlerMethod && ((HandlerMethod) o).getMethod().isAnnotationPresent(ResponseBody.class)) {  httpServletResponse.setContentType("application/json; charset=UTF-8"); |

|  |
| --- |
| PrintWriter out = httpServletResponse.getWriter(); out.flush();  out.println(new AjaxJson<>(false, "没有权限", null).toJson());  return true;  }  return false;  } |

使用了 shrio 框架进行开发，对用户密码进行了加密，保证管理员任意查看到其他用户的密码，保证了系统的安全。优化代码如下所示:

|  |
| --- |
| public static String MD5Encode(String origin) { String resultString = null;  try {  resultString = origin;  MessageDigest md = MessageDigest.*getInstance*("MD5");  resultString = *byteArrayToHexString*(md.digest(resultString.getBytes()));  } catch (Exception e) { e.printStackTrace();  }  return resultString;  } |

# 测试

### 测试方案设计

* + 1. 测试策略

测试策略应该严格遵循软件测试的规范和标准，测试策略有白盒测试，黑盒测试等。在本系统中，采用的是黑盒测试，因为黑盒测试能够在不清楚代码业务逻辑的情况下， 能够像在一个黑色的盒子里摸球一样，在很大程度上规避了了解逻辑代码的前提下对测试工作的影响。黑盒测试的典型错误是遗漏功能点，界面设计不合理等等。

* + 1. 测试进度安排

测试过程进度安排表如表 8-1 所示：

表 8-1 测试进度安排表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试任务 | 时间 | 执行人员 | 预计工作量（天） |
| 确定总体测试策略 | 2020-4-1 |  | 1 |
| 编写测试计划 | 2020-4-2 |  | 1 |
| 登录功能测试 | 2020-4-3 |  | 1 |
| 系统管理功能测试 | 2020-4-5 |  | 2 |
| 成果库管理功能测试 | 2020-4-7 |  | 1 |
| 活动管理功能测试 | 2020-4-8 |  | 3 |
| 专家资料功能测试 | 2020-4-11 |  | 2 |
| 标签维度管理功能测试 | 2020-4-13 |  | 1 |
| 专家库管理功能测试 | 2020-4-14 |  | 3 |
| 工作联络管理测试 | 2020-4-17 |  | 2 |
| 系统总体测试 | 2020-4-22 |  | 5 |
| 测试总结 | 2020-4-23 |  | 1 |
| 总计 |  |  | 22 |

* + 1. 测试资源

1. 人力资源:管理员和其他用户的角色都由我来扮演。
2. 工具资源:采用 Junit 来进行单元测试。
   * 1. 关键测试点
3. 系统用户分类的管理。
4. 成果库，专家信息，专家库，活动主题，标签维度的增删改查。
5. 会议资料的上传和下载，会议纪要、专家意见的增删改查等。
6. 专家库调入专家资料功能测试。

### 测试用例构建

* + 1. 测试用例编写约定

1. 测试用例的编号一般遵循项目名称+时间+编号的形式的原则，便于查找测试用例以及测试用例的跟踪。
2. 测试用例的标题应该简洁明了表达此次测试用例的用途。
   * 1. 测试用例设计

采用控制变量法来设计测试用例，例如，在用户登录这个功能点上，可以选择错误的账号正确的密码，正确的账号错误的密码，正确的账号正确的密码，错误的账号错误的密码，乱码，不同大小写等来测试该功能点，因为这样的测试用例它的覆盖度较高， 代表性好，测出来的数据也比较权威。

另外就是测试用例设计中需要注意边界值的点，要确保边界值有点，边界值的左右边也同样有测试用例，因为边界这附近能有效测试，还有就是不仅需要覆盖有效类，无效类也需要设计用例覆盖，如果用例设计中没有无效类，那么这次测试也很可能是一次失败的测试。

* + 1. 关键测试用例

表 8-2 测试用例 1

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 登录功能测试 | 用例编号 | 01 |
| 测试人员 |  | 测试时间 | 2020.4.1 |
| 测试项：验证后台登录功能，查看管理员是否能正常登陆。 | | | |
| 输入：   1. 打开智库系统网站 2. 找到登录页面 3. 输入正确账号及密码 4. 输入正确账号及错误密码 5. 输入错误账号及错误密码 6. 输入乱码账号及密码 7. 输入不同大小写账号及密码 | | | |
| 预期结果：当输入 1234qwer!时，能正确登录，其他错误； | | | |
| 测试结论： 正确 | | | |

表 8-3 测试用例 2

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 用户管理功能测试 | 用例编号 | 02 |
| 测试人员 |  | 测试时间 | 2020.4.2 |

续表 8-3

|  |
| --- |
| 测试项：管理员登录智库系统后，修改其他用户密码后，其他用户是否能正常登陆。 |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到用户页面，选择一用户为许映阳，点击修改 3. 输入修改密码为 12345qwer!，再次输入 12345qwer! 4. 返回用户登录界面，选择这许映阳的账号和密码来登录 |
| 预期结果：当输入用户名为许映阳和密码 12345qwer!时，能正确登录，其他错误； |
| 测试结论： 正确 |

表 8-4 测试用例 3

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 系统日志功能测试 | 用例编号 | 03 |
| 测试人员 |  | 测试时间 | 2020.4.3 |
| 测试项：管理员登录智库系统后，查看系统日志信息，日志信息是否更新。 | | | |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到用户日志管理，选择一其他用户为“李鹏辉”，用此账号登录，在活动管理中新增活动主题； 3. 在管理员账号中，查看系统日志信息 | | | |
| 预期结果：显示在此时间内李鹏辉新增活动主题信息，其他错误； | | | |
| 测试结论： 正确 | | | |

表 8-5 测试用例 4

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 成果库管理功能测试 | 用例编号 | 04 |
| 测试人员 |  | 测试时间 | 2020.4.5 |
| 测试项：管理员登录智库系统后，新增成果信息，输入成果关键字，查询成果信息。 | | | |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到成果库管理，点击新增，输入中医药成果信息后 3. 在搜索框中搜索关键字“中医药”，点击查询 | | | |
| 预期结果：显示出中医药成果信息，其他错误； | | | |
| 测试结论： 正确 | | | |

表 8-6 测试用例 5

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 专家信息管理功能测试 | 用例编号 | 05 |
| 测试人员 |  | 测试时间 | 2020.4.6 |
| 测试项：管理员登录智库系统后，新增专家信息，输入专家姓名，查询专家信息。 | | | |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到专家资料管理，点击新增，输入新增专家钟南山信息后 3. 在搜索框中搜索关键字“钟南山”，点击查询 | | | |
| 预期结果：显示出钟南山资料信息，其他错误； | | | |
| 测试结论： 正确 | | | |

表 8-7 测试用例 6

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 专家库管理功能测试 | 用例编号 | 06 |
| 测试人员 |  | 测试时间 | 2020.4.7 |
| 测试项：管理员登录智库系统后，新增专家库，在专家库调入增专家信息，输入专家姓名，查询是  否专家信息。 | | | |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到专家库管理，点击新增，输入新增库医学后 3. 调入医学专家钟南山 4. 搜索框输入关键字“钟南山”，点击查询 | | | |
| 预期结果：显示出钟南山资料信息，其他错误； | | | |
| 测试结论： 正确 | | | |

表 8-8 测试用例 7

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 会议资料上传下载功能  测试 | 用例编号 | 07 |
| 测试人员 |  | 测试时间 | 2020.4.10 |
| 测试项：管理员登录智库系统后，点击活动管理，在疫情活动中，上传会议资料，“新冠肺炎诊疗方案 word 资料”，然后店家下载，查看 word 文档是否发生错误 | | | |

续表 8-8

|  |
| --- |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到活动管理，点击疫情活动，选择会议资料 3. 点击上传，选择新冠肺炎诊疗方案 word 文本 4. 找到上传信息，点击下载 |
| 预期结果：显示出上传成功，下载成功，点开 word 查看无误，其他错误； |
| 测试结论： 正确 |

表 8-9 测试用例 8

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 标签维度管理功能测试 | 用例编号 | 08 |
| 测试人员 |  | 测试时间 | 2020.4.13 |
| 测试项：管理员登录智库系统后，点击标签维度管理，新增标签医学，在专家资料管理中设置专家  标签，查看是否设置成功 | | | |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到标签维度管理，点击新增，输入医学标签 3. 找到专家资料管理，找到“钟南山”，点击设置标签 4. 选择标签，点击设置 | | | |
| 预期结果：在选择标签中，有医学选项，设置成功后，点击“钟南山”详细信息，显示其中有医学  标签，其他均错误 | | | |
| 测试结论： 正确 | | | |

表 8-10 测试用例 9

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例名称 | 工作联络管理功能测试 | 用例编号 | 09 |
| 测试人员 |  | 测试时间 | 2020.4.20 |
| 测试项：管理员登录智库系统后，点击工作联络管理，设置联络员许映阳与专家钟南山，用联络员  账号许映阳登录后，是否能够选择钟南山专家，并编写联络信息 | | | |
| 输入：   1. 打开智库系统网站，管理员登录系统后 2. 找到工作联络管理，点击工作联络设置，选择联络员许映阳，专家钟南山 3. 用 “许映阳”账号登录，找到工作联络模块，选择联络专家， 4. 编写联络记录 | | | |

续表 8-10

|  |
| --- |
| 预期结果：许映阳账号登录后，能够选择钟南山专家，并编写联络信息其他均错误 |
| 测试结论： 正确 |

* + 1. 测试用例维护

1. 及时删除过时的测试用例

在之前的测试用例检测出系统缺陷以后，可能存在系统缺陷被修复了，但是原有的测试用例并没有及时的删除，导致重复测试，加重了工作量，因此，在测试工程中，应该及时的删除已经过时的测试用例，做好测试文档的维护工作。

1. 及时删除冗余的测试用例

在测试系统的过程里，可能存在不同的测试人员重复测试已经出现的问题，并且都发布在测试文档中，这将导致测试用例出现冗余，因此，需要我们在系统测试之前，做好系统测试沟通工作，点明每个测试人员的工作量，避免重复冗余的测试用例出现。

1. 增加新的测试用例

在测试的过程中，某些测试工作检出出问题之后，需要更新需求，也可能是在发布新的版本之后，原有的测试用例没有覆盖新的功能点，这时就需要增加新的测试用例。

1. 改进测试用例

随着开发过程工作的推进，测试工作暴露出来的问题显现出来，及时地修复测试出来的系统问题，测试用例需要不断改进，因为某些问题随着系统功能发生变化，以前的测试用例显得不适用，这样就会影响测试工作的效率，故测试用例随着工作的进行需要不断改进，使之变得能够真正地测试出系统暴露的问题。

# 总结与展望

### 设计工作总结

通过此次毕业设计，使我把大学四年学习过的专业知识又重新温习了一遍，获益匪浅，深刻地理解了软件工程的理论方法和实践技术的结合对软件工程的重要性，科学流程对编码工作的益处。在本次毕业设计中，我运用了Spring+SpringMVC+Mybatis 框架技术，Maven 项目对象模型以及 shrio 安全框架技术。以前在校期间，也学过这些技术，以为对这些技术方法很熟悉，虽然不理解代码底层是如何运作的，但是也能通过书上的样例可以编写出一些简单的程序。

但是，通过此次的毕业设计，是我对 SSM 框架有了深刻的理解，知道了从前台数据传输到后台的控制层，再由控制层传输到服务层，再到数据访问层，通过数据访问层中的操作，转入到 mybatis 框架中的 mapper,对数据库进行操作。这一详细的数据流程过 程。还是我深刻地理解了虽然这些技术讲起来不是很难，但是其中的过程却是坎坷艰难， 一面要处理好前台界面的设计，一面又要处理前台数据格式，还要处理后台数据转换过程。Shrio 安全框架在企业级开发中，是很常见的技术，但是由于此前没有学习过，本 次设计还是出现了不少问题，最终完成了对用户密码的加密工作。在整个过程中，虽然时不时就出现 bug,但是在处理 bug 的过程中，是我对工作实践的过程有了自己的一套方 法，对编程技术有了很大的提升。还有就是代码优化过程，虽然重复的代码还是很多， 但是还是尽量做到了系统的稳定性和响应能力达标。

就整个系统而言，虽然不是很完美，一些界面不是非常美观，一些操作过程有点繁琐，但是整个项目还是基本符合前期的期望，通过此次的动手操作经历，使我明白了一些工作看似简单，但是实际动手起来还是需要认真对待，把虚心好学的心态融入进去， 才能在以后的学习和工作中立于不败之地。

### 未来工作展望

通过此次的毕业设计，是我明白了在未来的工作中，光有技术，光有理论是远远不够的，还要对事物进行发掘，研究出其中的需求。在未来开发系统中，一定要按照软件工程的理论来实行，详尽的写出每个需求文档，技术文档，测试文档等等，因为这样会大大的提高工作效率，还要保持虚心好学的心态，不懂的问题要向老师和前辈们虚心请教，加强团队合作的能力，互相帮助，达到整个项目安全快速的开发。在编码过程中， 要严格遵守编码规范，详尽写出注释，便于系统的维护。

进入学习是我这一生的财富，从进入大学的第一天起，我曾经迷茫过， 总是认为自己的能力不足，对于软件这一专业觉得很深奥，很难学，但是从我和我室友及同学们第一次参加学院举办的程序竞赛之后，我觉得我还是可以做些东西出来的，只要自己努力就行，当下网络学习资源丰富，从中学习到不少的新技术，新想法，新观点。我也很珍惜和同学们一起的学习和生活时光，这是我人生的一笔财富。我会倍加珍惜各

位老师和同学们在我大学时光对我的帮助。拥有本次的程序设计的经验，在未来的编程工作和学习中，我会更加的从容和客观地面对各种技术问题，努力的学习各种更新的新技术、新方法。

我的毕业设计在我不断的修改下,终于在解决了各种各样的bug 后成功完美的运行了一次,但是当我把我的成果发送给指导老师给他看过之后,他却看出了不一样的地方。在那些我没能注意到小角落,就像是选择时间的时候选项的第一个时间没有和现在的时间对应这样的小事。如果是练习的话这样的小细节自然不是什么大问题,但是如果作为以使用为前提的毕业设计,这就远远不够,在接受了老师的建议后,我又继续对自己毕业设计的，最终完成了本次的毕业设计。

本次的毕业设计仍然存在某些问题，例如应该增加系统登录的手机验证码功能，增加考勤定位功能等，由于突如其来的疫情，打乱了学习和工作的节奏，在一边准备研究生复试一边写毕业设计的过程中，我深刻地懂得了自己的潜力和耐力，在以后工作中， 我都能像这段时间一样，能够珍惜学习时间，感受到生命的短暂，应该努力的学习，努力的融入社会工作，把每次的工作都努力的完成好，为我国社会建设贡献自己的一份力量。

# 谢 辞

# 参考文献

[1]刘义忠,张伟.基于 SSM 框架的后台管理系统设计与实现[J].软件导刊,2019，18(2):68-71. [2]周国华.基于 SSM 框架学习平台的设计与实现[J].信息与电脑,2017,0(24):138-139.

[3]贺雪梅.Web 应用开发中的 SSM 框架设计［J］. 电子世界,2019,0(1):206-206. [4]张晶.智库专家信息管理系统的设计与实现[J] 2017，6.

[5]许云鹏. 智库管理系统的设计与实现[J].《信息技术时代·上旬刊》 2019 年第 01 期. [6]柳青松.[JAVA 语言的开发平台及 J2EE 编程技术](http://www.cnki.com.cn/Article/CJFDTOTAL-DZRU201717048.htm)[J];电子技术与软件工程;2017 年 17 期.

[7]辽宁省财政科学研究所，辽宁省财政学会.中国智库发展的现状与特点[J]. 2015 年 5 月 25 日. [8]张煜东.吴乐南;王水花;;[专家系统发展综述](http://www.cnki.com.cn/Article/CJFDTOTAL-JSGG201019013.htm)[J];计算机工程与应用;2010 年 19 期.

1. 李刚. 轻量级 Java EE 企业应用实战（第 5 版）[M]——Struts 2+Spring 5+Hibernate 5/JPA 2 整合开发.电子工业出版社.
2. 薛华成.管理信息系统[M].清华大学出版社,1996.7.2-5.
3. 曹阳.软件工程 Web 技术开发平台[J].电子技术与软件工程,2019,0(5):34-34.
4. 朱丹丹.[浅析 JAVA 语言的开发平台及 J2EE 编程技术](http://www.cnki.com.cn/Article/CJFDTOTAL-HBYD201511072.htm)[J];信息通信;2015 年 11 期. [13]刘颖.软件工程技术在系统软件开发中的运用[J].电子技术与软件工程,2019,0(5):31-31. [14]萨师煊,王珊.数据库系统概论[M].高等教育出版社,2000.7:21-347.
5. 张亚东.酒店企业销售管理信息系统的设计与实现[J].管理信息系统,2000.9:45249.
6. 彼得·戈特沙尔克,陈源.[世界信息管理系统关键课题研究](http://www.cnki.com.cn/Article/CJFDTOTAL-GWSH200006025.htm)[J];国外社会科学;2000 年06 期. [17]徐盼.[我国社会智库有效性的评价和提升策略研究](http://cdmd.cnki.com.cn/Article/CDMD-10358-1017295596.htm)[J];中国科学技术大学;2017 年.
7. 熊立勇.[国际比较视域下中国特色新型智库建设问题研究](http://cdmd.cnki.com.cn/Article/CDMD-10358-1017295550.htm)[J];中国科学技术大学;2017 年.
8. 张海藩, 吕云翔. 《软件工程(第 4 版）》［M］. 北京：人民邮电出版社，2013. 39-42.
9. 杨开振, 周吉文. 《Java EE E 联网轻量级框架整合开发》［M］. 北京：电子工业出版社， 2017.224-255.
10. 王雨竹. 《MySQL 入门经典》［M］. 机械工业出版社,2014.158-232.
11. 周志明. 《深入理解Java 虚拟机 JVM 高级特性与最佳实践(第二版)》［M］. 北京：机械工业出版社，2013. 139-217.
12. 杨开振.《深入浅出 MyBatis 技术原理与实战》［M］. 北京：电子工业出版社，2016.133-188. [24]舒红平.Web 数据库编程-Java［M］.西安电子科技大学出版社，2005：97-143.

[25]姚素红. 基于 Java 的Web 应用系统开发模式[J]. 江苏工程职业技术学院学报,2018,18(01):6-9. [26]Abdelhak Mesbah,Jean-Louis Lanet,Mohamed Mezghiche. Reverse engineering a Java Card memory management algorithm[J]. Computers & Security,2017,66.

[27]Bruce Eckel. 《thinking in java》[M]. Prentice Hall PTR, 2013.400-526. [28][Ogihara](http://search.dangdang.com/?key2=Ogihara&medium=01&category_path=01.00.00.00.00.00), [Mitsunori](http://search.dangdang.com/?key2=Mitsunori&medium=01&category_path=01.00.00.00.00.00).《Fundamentals of JAVA Programming》[M] Springer. 2019.8.

1. Mark Allen Weiss. 《Data structures and algorithms in java》[M]. Cengage Learning Asia, 2014.142-226.
2. Rasoolzadegan A, Barforoush A. Reliable yet flexible software through formal model transformation (rule definition)[J]. Knowledge & Information Systems, 2014.79-126.

## 附录 A 外文翻译—原文部分

The progress of abstraction

All programming languages provide abstractions. It can be argued that the complexity of the problems you’re able to solve is directly related to the kind and quality of abstraction. By“kind” I mean, “What is it that you are abstracting?” Assembly language is a small abstraction of the underlying machine. Many so-called “imperative” languages that followed (such as FORTRAN, BASIC, and C) were abstractions of assembly language. These languages are big improvements over assembly language, but their primary abstraction still requires you to think in terms of the structure of the computer rather than the structure of the problem you are trying to solve. The programmer must establish the association between the machine model (in the “solution space,” which is the place where you’re implementing that solution,such as a computer) and the model of the problem that is actually being solved (in the16 Thinking in Java Bruce Eckel“problem space,” which is the place where the problem exists, such as a business). The effort required to perform this mapping, and the fact that it is extrinsic to the programming language, produces programs that are difficult to write and expensive to maintain, and as a side effect created the entire “programming methods” industry. The alternative to modeling the machine is to model the problem you’re trying to solve.

Earlylanguages such as LISP and APL chose particular views of the world (“All problems are ultimately lists” or “All problems are algorithmic,” respectively). Prolog casts all problems into chains of decisions. Languages have been created for constraint-based programming and for programming exclusively by manipulating graphical symbols. (The latter proved to be too restrictive.) Each of these approaches may be a good solution to the particular class of problem they’re designed to solve, but when you step outside of that domain they become awkward.

The object-oriented approach goes a step further by providing tools for the programmer

to

represent elements in the problem space. This representation is general enough that the

programmer is not constrained to any particular type of problem. We refer to the elements in the problem space and their representations in the solution space as “objects.” (You will also need other objects that don’t have problem-space analogs.) The idea is that the program is allowed to adapt itself to the lingo of the problem by adding new types of objects, so when you read the code describing the solution, you’re reading words that also express the problem. This is a more flexible and powerful language abstraction than what we’ve had before.

* 1. Thus, OOP allows you to describe the problem in terms of the problem, rather than in terms of the computer where the solution will run. There’s still a connection back to the computer: Each object looks quite a bit like a little computer—it has a state, and it has

operations that you can ask it to perform. However, this doesn’t seem like such a bad analogy to objects in the real world—they all have characteristics and behaviors.

Alan Kay summarized five basic characteristics of Smalltalk, the first successful object-oriented language and one of the languages upon which Java is based. These characteristics

represent a pure approach to object-oriented programming:

1. Everything is an object. Think of an object as a fancy variable; it stores data,but you can “make requests” to that object, asking it to perform operations on itself. In theory, you can take any conceptual component in the problem you’re trying to solve(dogs, buildings, services, etc.) and represent it as an object in your program.
2. A program is a bunch of objects telling each other what to do by sending messages. To make a request of an object, you “send a message” to that object. More concretely, you can think of a message as a request to call a method that belongs to a particular object.
3. Each object has its own memory made up of other objects. Put another way, you create a new kind of object by making a package containing existing objects. Thus, you can build complexity into a program while hiding it behind the simplicity of objects.
4. Every object has a type. Using the parlance, each object is an instance of a class, in which “class” is synonymous with “type.” The most important distinguishing characteristic of a class is “What messages can you send to it?”
5. All objects of a particular type can receive the same messages. This is actually a loaded statement, as you will see later. Because an object of type “circle” is also an object of type “shape,” a circle is guaranteed to accept shape messages. This means you can write code that talks to shapes and automatically handle anything that fits the description of a shape. This substitutability is one of the powerful concepts in OOP.

Booch offers an even more succinct description of an object:

An object has state, behavior and identity.

This means that an object can have internal data (which gives it state), methods (to produce behavior), and each object can be uniquely distinguished from every other object—to put this in a concrete sense, each object has a unique address in memory.

* 1. An object has an interface

Aristotle was probably the first to begin a careful study of the concept of type; he spoke of“the class of fishes and the class of birds.” The idea that all objects, while being unique, are also part of a class of objects that have characteristics and behaviors in common was used directly in the first object-oriented language, Simula-67, with its fundamental keyword class that introduces a new type into a program.

Simula, as its name implies, was created for developing simulations such as the classic “bank teller problem.” In this, you have numerous tellers, customers, accounts, transactions, and units of money—a lot of “objects.” Objects that are identical except for their state during a program’s execution are grouped together into “classes of objects,” and that’s where the keyword class came from. Creating abstract data types (classes) is a fundamental concept in object-oriented programming. Abstract data types work almost exactly like built-in types:

You can create variables of a type (called objects or instances in object-oriented parlance) and manipulate those variables (called sending messages or requests; you send a message

and the object figures out what to do with it). The members (elements) of each class share some commonality: Every account has a balance, every teller can accept a deposit, etc. At the same time, each member has its own state: Each account has a different balance, each teller has a name. Thus, the tellers, customers, accounts, transactions, etc., can each be represented with a unique entity in the computer program. This entity is the object, and each object belongs to a particular class that defines its characteristics and behaviors.

So, although what we really do in object-oriented programming is create new data types,virtually all object-oriented programming languages use the “class” keyword. When you see the word “type” think “class” and vice versa.

* 1. Since a class describes a set of objects that have identical characteristics (data elements) and behaviors (functionality), a class is really a data type because a floating point number, for example, also has a set of characteristics and behaviors. The difference is that a programmer defines a class to fit a problem rather than being forced to use an existing data type that was designed to represent a unit of storage in a machine. You extend the programming language by adding new data types specific to your needs. The programming system welcomes the new classes and gives them all the care and type checking that it gives to built-in types.

The object-oriented approach is not limited to building simulations. Whether or not you agree that any program is a simulation of the system you’re designing, the use of OOP techniques can easily reduce a large set of problems to a simple solution. Once a class is established, you can make as many objects of that class as you like, and then manipulate those objects as if they are the elements that exist in the problem you are trying to solve. Indeed, one of the challenges of object-oriented programming is to create a one-to-one mapping between the elements in the problem space and objects in the solution space.

But how do you get an object to do useful work for you? There needs to be a way to make a request of the object so that it will do something, such as complete a transaction, draw something on the screen, or turn on a switch. And each object can satisfy only certain requests. The requests you can make of an object are defined by its interface, and the type is what

determines the interface. A simple example might be a representation of a light bulb: The interface determines the requests that you can make for a particular object. However,there must be code somewhere to satisfy that request. This, along with the hidden data, comprises the implementation. From a procedural programming standpoint, it’s not that complicated. A type has a method associated with each possible request, and when you make a particular request to an object, that method is called. This process is usually summarized by saying that you “send a message” (make a request) to an object, and the object figures out what to do with that message (it executes code).

Here, the name of the type/class is Light, the name of this particular Light object is lt, and the requests that you can make of a Light object are to turn it on, turn it off, make it

brighter, or make it dimmer. You create a Light object by defining a “reference” (lt) for that object and calling new to request a new object of that type. To send a message to the object, you state the name of the object and connect it to the message request with a period (dot).

From the standpoint of the user of a predefined class, that’s pretty much all there is to programming with objects.

The preceding diagram follows the format of the Unified Modeling Language (UML). Each class is represented by a box, with the type name in the top portion of the box, any data members that you care to describe in the middle portion of the box, and the methods (the functions that belong to this object, which receive any messages you send to that object) in the bottom portion of the box. Often, only the name of the class and the public methods are shown in UML design diagrams, so the middle portion is not shown, as in this case. If you’re interested only in the class name, then the bottom portion doesn’t need to be shown, either.

## 附录 B 外文翻译—译文部分

所有编程语言都提供抽象（abstraction）机制。可以认为，你所能够解决的问题的复杂性直接取决于抽象的类型和质量。我所谓的“类型”是指“你所抽象的是什么？” 汇编语言是对底层机器的小型抽象。接着出现的许多所谓“命令式（Imperative）”语言

（诸如 FORTRAN、BASIC、C 等）都是对汇编语言的抽象。这些语言在汇编语言之上有了大幅的改进，但是它们所作的主要抽象仍要求你在解决问题时要基于计算机的结构，而不是基于你试图要解决的问题的结构来考量。

程序员必须建立在机器模型（Machine Model）（位于你对问题建模所在的解空间

（Solution Space）内，例如计算机）和实际待解问题模型（Problem Model）（位于问题所在的问题空间（Problem Space）内）之间的关联。建立这种映射（Mapping）是费力的，而且它不属于编程语言的内在性质，这使得程序难以编写，并且维护代价高昂。由此，产生了完整的“编程方法（Programming Method）”产业。

另一种对机器建模的方式就是对待解决问题建模。早期的编程语言，诸如 LISP 和APL 都选择世界的某种特定视图（分别对应于“所有问题最终都是列表（List）”或者“所有问题都是算法形式的（algorithmic）”）。PROLOG 则将所有问题都转换成为决策链

（Chain of decisions）。此外还产生了基于约束条件（constraint-based）编程的语言和专门通过对图形符号操作来实现编程的语言（后者被证明限制性过强）。这些方式对于它们被设计时所瞄准要解决的特定类型的问题都是不错的解决方案，但是一旦超出其特定领域，它们就力不从心了。

面向对象方式（Object-oriented approach）通过向程序员提供用来表示在问题空间中的元素的工具而更进一步。这种表示方式具有足够的概括性，使得程序员不会受限于任何特定类型的问题。我们将问题空间中的元素及其在解空间中的表示成为“对象

（Object）”。（你还需要一些无法类比为问题空间元素的对象）。这种思想的实质是：程序可以通过添加新类型的对象使自身适用于某个特定问题。因此，当你在阅读描述解决方案的代码的同时，也是在阅读问题的表述。相比以前我们所拥有的所有语言，这是一种更灵活和更强有力的语言抽象。

所以，OOP 允许以问题的形式来描述问题，而不是以执行解决方案的计算机的形式来描述问题。但是它仍然与计算机有联系：每个对象看起来都有点像一台微型计算机—

—它具有状态，并且能够执行你赋予它的各种操作。如果要在现实世界中对对象作类比， 那么说它们都具有特性（Characteristic）和行为（Behavior）似乎不错。

Alan Kay 曾经总结了第一个成功的面向对象语言，同时也是 Java 赖为根基的语言之一的 Smalltalk 的五个基本特性，这些特性表现了一种纯粹的面向对象程序设计方式：

1. 万物皆为对象。将对象视为奇特的变量，它可以存储数据，除此之外，你还可以要求它在自身上执行操作。理论上讲，你可以抽取待解问题的任何概念化构件（狗、建筑物、服务等），将其表示为程序中的对象。
2. 程序是对象的集合，它们彼此通过发送消息来调用对方。要想产生一个对对象的请求，就必须对该对象发送一条消息。更具体地说，你可以把消息想象为对某个特定对象的方法的调用请求。
3. 每个对象都拥有由其它对象所构成的存储。你可以通过创建包含现有对象集合的包的方式来创建新类型的对象。因此，你可以在程序中构建复杂的体系，同时将其复杂性通过对象的质朴性得以屏蔽。
4. 每个对象都拥有其类型（Type）。按照通用的说法，“每个对象都是某个类（Class）的一个实例（Instance）”，其中“类”就是“类型”的同义词。每个类中最重要的区别于其它类的特性就是“你可以发送什么消息给它？”
5. 某一特定类型的所有对象都可以接收(Receive)同样的消息。这是一句意味深长的表述，你在稍后便会看到。因为“圆形（circle）”类型的对象同时也是“几何形（shape）”类型的对象，所以一个“圆形”对象必定能够接受（accept）发送给“几何形”对象的消息。这意味着你可以编写与“几何形”交互并自动处理所有与几何形性质相关的事物的的代码。这种“可替代性（substitutability）”是 OOP 中最强有力的概念之一。

Booch 提出了一个对对象的更加简洁的描述：对象拥有状态（ State ）、行为(Behaviour)和标识(Identity)。这意味着每一个对象都可以拥有内部数据（它们给出了该对象的状态）和方法（它们产生行为），并且每一个对象都可以唯一地与其他对象相区分开，具体说来，就是每一个对象在内存中都有一个唯一的地址。

亚里士多德大概是第一个深入研究类型（Type）的哲学家，他曾提出过鱼类和鸟类

（the class of fishes and the class of birds）这样的概念。所有的对象都是唯一的，但同时也是具有相同的特性和行为的对象所归属的类的一部分，这种思想被直接应用于第一个面向对象语言 Simula-67，它在程序中使用基本关键词 class 来引入新的类型。

Simula，就像其名字一样，是为了开发诸如经典的“银行出纳员问题（Bank teller problem）”这样的仿真程序而创建的。在银行出纳员问题中，有出纳员、客户、账户、交易和货币单位等许多“对象”。在程序执行期间具有不同的状态而其他方面都相似的对象会被分组到对象的类中，这就是关键词 class 的由来。创建抽象数据类型（类）是面向对象程序设计的基本概念之一。

抽象数据类型的运行方式与内置（built-in）类型几乎完全一致：你可以创建某一类型的变量（按照面向对象的说法，程其为对象或实例），然后操作这些变量（称其为发送消息或请求；你发送消息，对象就能够知道需要做什么）。每个类的成员（member）或元素（element）都共享相同的性质：每个账户都有结余金额，每个出纳都可以处理存款请求等。同时，每个成员都有其自身的状态：每个账户都有不同的结余金额，每个出纳

都有自己的名称。因此，出纳、客户、账户、交易等都可以在计算机程序中被表示成为唯一的实体（entity）。这些实体就是对象，每一个对象都属于定义了特性和行为的某个特定的类。

所以，尽管我们在面向对象程序设计中实际所作的是创建新的数据类型，但事实上所有的面向对象程序设计语言都使用 Class 关键词来表示数据类型。当你看到类型

（Type）一词时，请将其作为类（Class）来考虑，反之亦然。

既然类被描述成了具有相同特性（数据元素）和行为（功能）的对象集合，那么一个类就确实是一个数据类型，就像所有浮点型数字具有相同的特性和行为集合一样。二者的差异在于，程序员通过定义类来适应问题，而不再被强制只能使用现有的被设计用来表示在机器中的存储单元的数据类型。你可以根据需求，通过添加新的数据类型来扩展编程语言。编程系统欣然接受新的类，并且给予它们与内置类型相同的管护和类型检查（Type-checking）。

面向对象方法并不是仅局限于构件仿真程序。无论你是否同意任何程序都是你所设计的系统的一个仿真的观念，面向对象技术确实可以将大量的问题降解为一个简单的解决方案。一旦类被建立，你想要创建该类的多少个对象，就可以创建多少个了，然后去操作它们，就像它们是存在于你的待解问题中的元素一样。事实上，面向对象程序设计的挑战之一，就是在问题空间的元素和解空间的对象之间创建一对一的映射。

但是，你怎样才能获得对你有用的对象呢？必须有某种方式产生对对象的请求，使对象完成诸如完成一笔交易、在屏幕上画图、打开开关之类的任务。每个对象都只能满足某些请求，这些请求由对象的接口（Interface）所定义，决定接口的便是类型（Type）。以电灯泡为例来做一个简单的比喻：接口定义了你能够对某一特定对象发出的请求。但是，在程序中必须有满足这些请求的代码。这些代码与隐藏的数据一起构成了实现

（implementation）。从过程型编程的观点来看，这并不太复杂。在类型中，每一个可能的请求都有一个方法与之相关联，当你向对象发送请求时，与之相关联的方法就会被调用。此过程通常被总结为：你向某个对象发送消息（产生请求），这个对象便知道此消息的目的，然后执行对应的程序代码。

上例中，类型/类的名称是 Light，特定的 Light 对象的名称是 lt，你可以向 Light 对象发出的请求是：打开它、关闭它、将它调亮、将它调暗。你以这种方式创建了一个Light 对象：定义这个对象的“引用（reference）”（lt），然后调用 new 方法来创建该类型的新对象。为了向对象发送消息，你需要声明对象的名称，并以圆点符号连接一个消息请求。从预定义类的用户观点来看，这些差不多就是用对象来进行设计的全部。

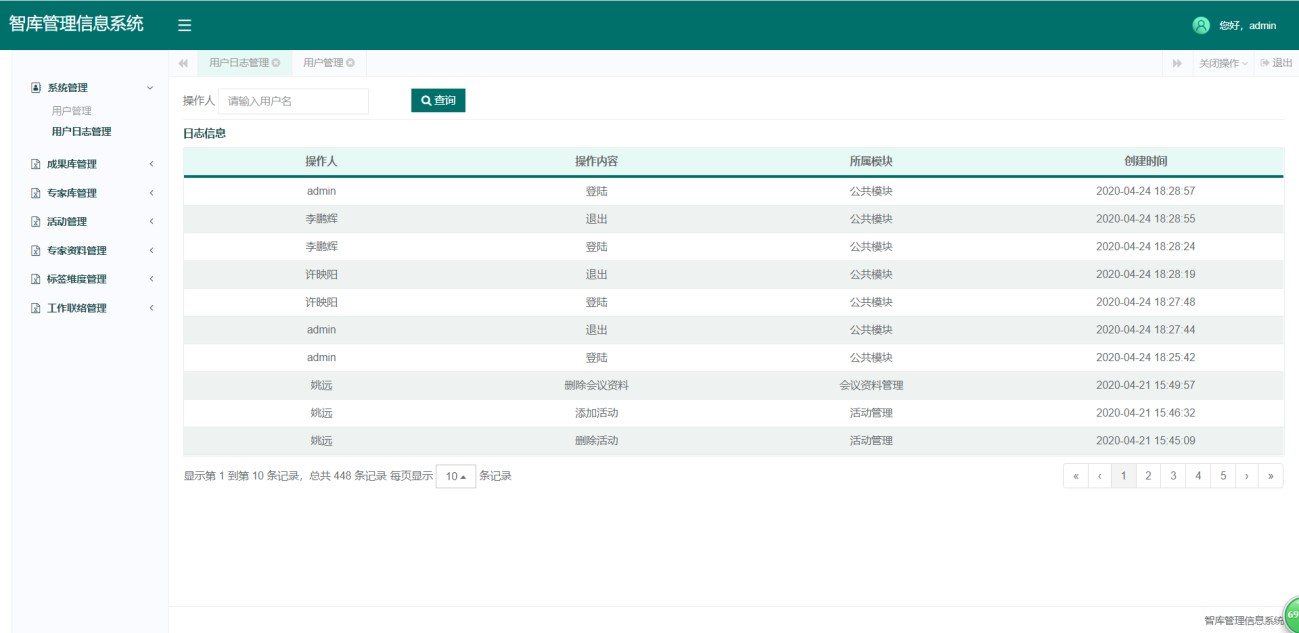
前面的图是 UML(Unified Modelling Language)形式的图，每个类都用一个方框表示，类名在方框的顶部，你所关心的任何数据成员（data member）都描述在方框的中间部分，方法（隶属于此对象的，用来接收你发给此对象的消息的函数）在方框的底部。通常，只有类名和公共方法（Public Method）被示于 UML 设计图中，因此，方框的中部并不绘出。如果你只对类型感兴趣，那么方框的底部甚至也不需要被绘出。

## 附录 C 软件使用说明书

本项目是教育厅机关内部项目，因此，用户必须为机关内部人员，由管理员注册之后，用户方可登录本系统，登录之后根据角色分类，拥有不同的系统功能。

管理员角色：

登录之后，拥有系统的一切权限，包括系统管理，成果库，专家库，专家资料，活动，工作联络，标签维度功能模块，点击左框的具体功能进行操作。如下图所示：



联络员角色：

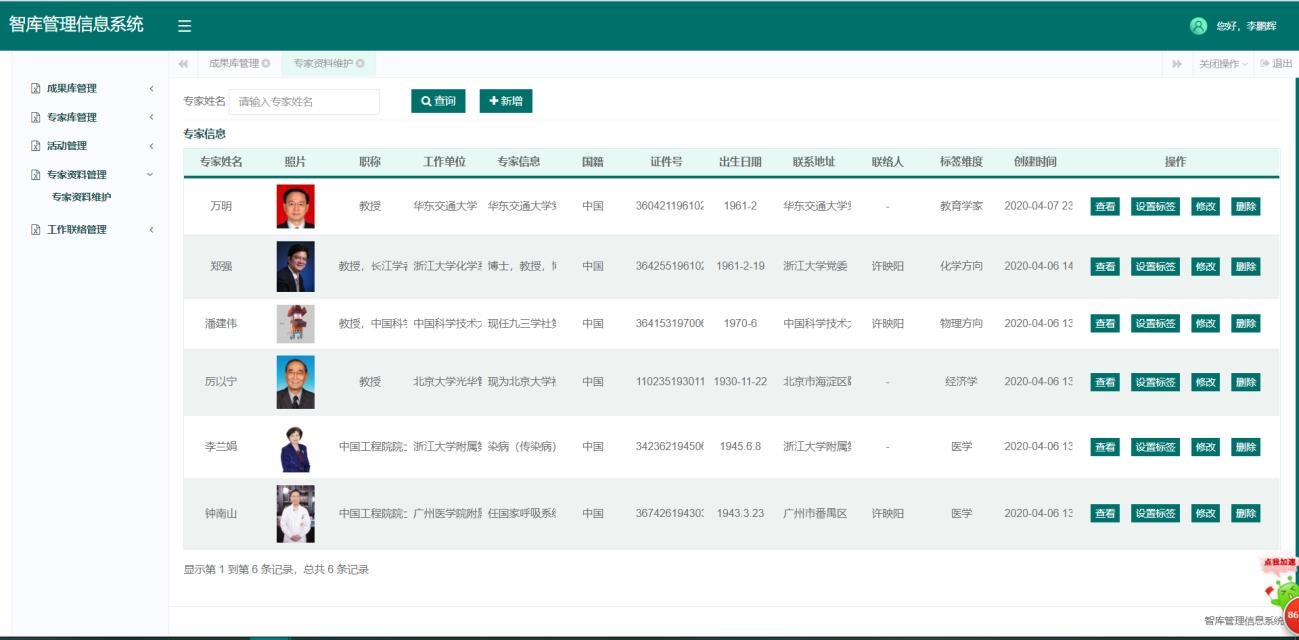
管理员角色后台管理页面

登录之后，对工作联络记录进行操作。如下图所示：

领导角色：

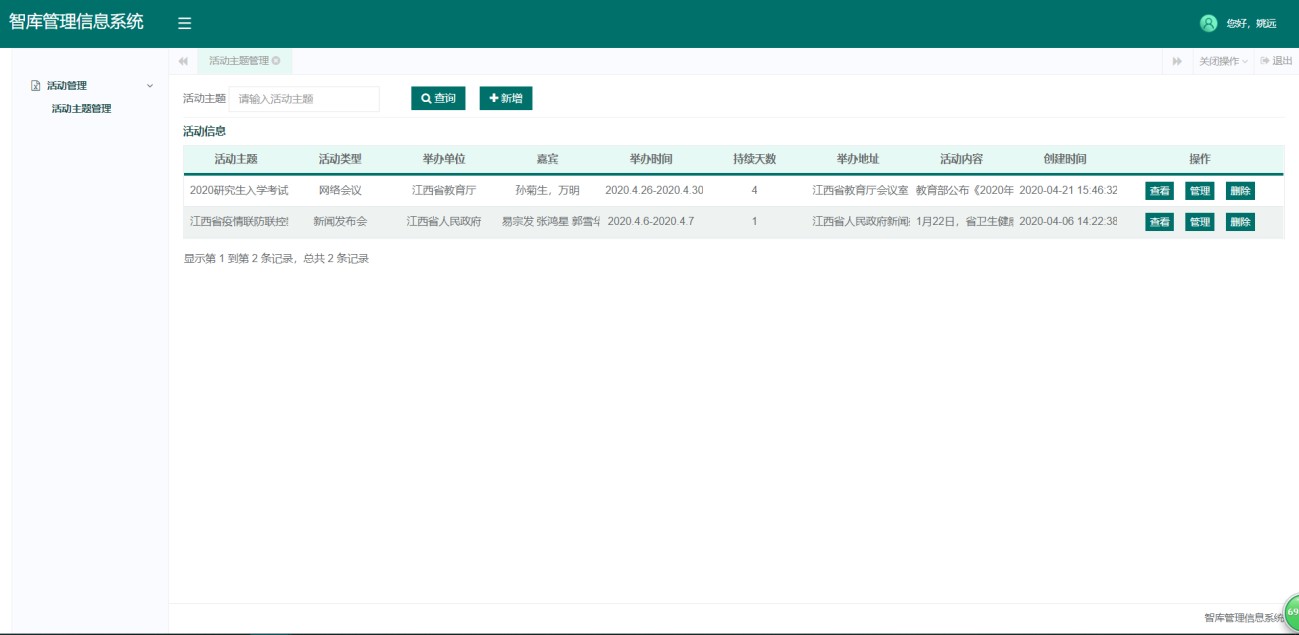
联络员角色后台管理页面

登录之后，点击左框，相应地对成果库，专家库，活动管理，工作联络，专家资料功能进行操作，如下图所示：



秘书角色：

领导角色后台管理页面

登录之后，点击左框，活动主题进行操作，如下图所示：

秘书角色后台管理页面

## 附录 D 主要源代码

|  |
| --- |
| @Controller @RequestMapping("/account")  **public class** AccountController **extends** BaseController {  **private static final** Logger ***logger*** = LoggerFactory.*getLogger*(AccountController.**class**);  @Autowired  **private** UserService userService;  @Autowired  **private** LogService logService;  @Autowired  **private** AchievementsService achievementsService;  @Autowired  **private** ActivityService activityService;  @Autowired  **private** ExpertService expertService;  @Autowired  **private** LiaisonRecordService liaisonRecordService;  @Autowired  **private** IdeaService ideaService;  @Autowired  **private** SummaryService summaryService;  @RequestMapping("/login")  **public** String login() {  **return** "login"; |

|  |
| --- |
| }  @RequestMapping("/main")  **public** String main() {  **return** "main";  }  @RequestMapping("/user")  **public** String user() {  **return** "user";  }  @RequestMapping("/expert")  **public** String expert() {  **return** "expert";  }  @RequestMapping("/tag")  **public** String tag() {  **return** "tag";  }  @RequestMapping("/log")  **public** String log() {  **return** "log";  }  @RequestMapping("/expert\_group")  **public** String expert\_group() {  **return** "expert\_group";  }  @RequestMapping("/expert\_group\_detail")  **public** String expert\_group\_detail(String groupId,Model model) { model.addAttribute("groupId", groupId);  **return** "expert\_group\_detail"; |

|  |
| --- |
| }  @RequestMapping("/liaison")  **public** String liaison() {  **return** "liaison";  }  @RequestMapping("/liaison\_record")  **public** String liaison\_record() {  **return** "liaison\_record";  }  @RequestMapping("/expert\_detail")  **public** String expert\_detail(String expertId,Model model) { Expert e=expertService.info(expertId); model.addAttribute("address", e.getAddress()); model.addAttribute("birth", e.getBirth()); model.addAttribute("createtime", e.getCreatetime()); model.addAttribute("id", e.getId()); model.addAttribute("idNumber", e.getIdnumber()); model.addAttribute("information", e.getInformation()); model.addAttribute("liaisonId", e.getLiaisonId()); model.addAttribute("liaisonName", e.getLiaisonName()); model.addAttribute("name", e.getName()); model.addAttribute("nation", e.getNation()); model.addAttribute("orgnization", e.getOrganization()); model.addAttribute("tagId", e.getTagId()); model.addAttribute("tagName", e.getTagName()); model.addAttribute("workTitle", e.getWorkTitle()); model.addAttribute("filePath", e.getFilePath());  **return** "expert\_detail";  }  @RequestMapping("/liaison\_record\_detail")  **public** String liaison\_record\_detail(String recordId,Model model) { |

|  |
| --- |
| LiaisonRecord e=liaisonRecordService.info(recordId); model.addAttribute("expertName", e.getExpertName()); model.addAttribute("liaisonName", e.getLiaisonName()); model.addAttribute("method", e.getMethod()); model.addAttribute("title", e.getTitle()); model.addAttribute("content", e.getContent()); model.addAttribute("createtime", e.getCreatetime());  **return** "liaison\_record\_detail";  }  @RequestMapping("/idea\_detail")  **public** String idea\_detail(String ideaId,Model model) { Idea e=ideaService.info(ideaId);  model.addAttribute("expertName", e.getExpertName()); model.addAttribute("information", e.getInformation()); model.addAttribute("createtime", e.getCreatetime());  **return** "idea\_detail";  }  @RequestMapping("/summary\_detail")  **public** String summary\_detail(String summaryId,Model model) { Summary e=summaryService.info(summaryId); model.addAttribute("stage", e.getStage()); model.addAttribute("information", e.getInformation()); model.addAttribute("createtime", e.getCreatetime()); model.addAttribute("title", e.getTitle());  **return** "summary\_detail";  }  @RequestMapping("/achievements")  **public** String achievements() {  **return** "achievements";  }  @RequestMapping("/achievements\_detail") |

|  |
| --- |
| **public** String achievements\_detail(String achievementsId,Model model) { Achievements a=achievementsService.info(achievementsId); model.addAttribute("title", a.getTitle()); model.addAttribute("information", a.getInformation()); model.addAttribute("expertName", a.getExpertName()); model.addAttribute("createtime", a.getCreatetime());  **return** "achievements\_detail";  }  @RequestMapping("/activity")  **public** String activity() {  **return** "activity";  }  @RequestMapping("/activity\_detail")  **public** String activity\_detail(String activityId,String isManage,Model model) { Activity a=activityService.info(activityId);  model.addAttribute("theme", a.getTheme()); model.addAttribute("information", a.getInformation()); model.addAttribute("createtime", a.getCreatetime()); model.addAttribute("activityId", activityId); model.addAttribute("isManage", isManage);  **return** "activity\_detail";  }  @RequestMapping("/userInfo")  **public** String userInfo(Model model) { User user =ShiroUtils.*getUser*();  model.addAttribute("username", user.getUsername()); model.addAttribute("mobile", user.getMobile()); model.addAttribute("email", user.getEmail()); model.addAttribute("id", user.getId());  **return** "userInfo";  }  @GetMapping({ "/index" }) |

|  |
| --- |
| String index(Model model,String redirect) { User u = ShiroUtils.*getUser*();  User user = **new** User();  user =userService.selectUserById(u.getId()); model.addAttribute("username", user.getUsername()); model.addAttribute("type", u.getType());  **return** "index";  }  @SuppressWarnings("rawtypes") @ResponseBody  @RequestMapping(value = "/doLogin", method = RequestMethod.***POST***)  **public** AjaxJson doLogin(User user) {  **if** (user == **null**) {  **return new** AjaxJson(**false**, "参数错误");  }  **if** (StringUtils.*isBlank*(user.getUsername())) {  **return new** AjaxJson(**false**, "用户名不能为空");  }  **if** (StringUtils.*isBlank*(user.getPassword())) {  **return new** AjaxJson(**false**, "密码不能为空");  }  **try** {  String password = MD5.*MD5Encode*(user.getUsername() + user.getPassword()); UsernamePasswordToken token = **new** UsernamePasswordToken(user.getUsername(),  password);  Subject subject = SecurityUtils.*getSubject*(); subject.login(token);  Log log=**new** Log(); log.setUserId(getUser().getId());  log.setContent("登陆"); |

|  |
| --- |
| log.setModular("公共模块"); logService.insert(log);  **return new** AjaxJson(**true**, "success");  } **catch** (Exception e){  ***logger***.error(e.getMessage(), e);  **return new** AjaxJson(**false**, e.getMessage());  }  }  /\*\*   * 退出 * **@param** session * **@return**   \*/ @RequestMapping("/logout")  **public** String logout(HttpSession session) {  **try** {  Log log=**new** Log(); log.setUserId(getUser().getId());  log.setContent("退出"); log.setModular("公共模块"); logService.insert(log);  } **catch** (Exception e) { e.printStackTrace();  }  ShiroUtils.*logout*();  **return** "redirect:/account/login";  } |

|  |
| --- |
| @SuppressWarnings("rawtypes") @ResponseBody  @RequestMapping(value = "/changePwd", method = RequestMethod.***POST***)  **public** AjaxJson changePwd(String password) {  **if** (StringUtils.*isBlank*(password)) {  **return new** AjaxJson(**false**, "参数错误");  }  **try** {  User user = getUser(); user.setPassword(MD5.*MD5Encode*(user.getUsername() + password)); userService.update(user);  Log log=**new** Log(); log.setUserId(getUser().getId());  log.setContent("修改密码"); log.setModular("公共模块"); logService.insert(log);  } **catch** (Exception e) {  ***logger***.error(e.getMessage(), e);  **return new** AjaxJson(**false**, "修改失败");  }  **return new** AjaxJson(**true**, "修改成功");  }  } |