# 目 录

1. [**绪 论 1**](#_bookmark0)

### [研究的背景及意义 1](#_bookmark1)

### [选题的背景 1](#_bookmark1)

### [国内外研究现状 1](#_bookmark1)

### [研究的意义 1](#_bookmark1)

### [系统目标 2](#_bookmark2)

1. [**需求分析 3**](#_bookmark4)

### [业务需求 3](#_bookmark3)

### [主要业务流程 3](#_bookmark3)

### [功能需求 8](#_bookmark5)

### [角色分析 8](#_bookmark5)

### [业务功能 8](#_bookmark5)

### [非功能需求 19](#_bookmark6)

### [环境需求 19](#_bookmark6)

### [性能需求 19](#_bookmark6)

### [安全需求 20](#_bookmark7)

1. [**总体设计 21**](#_bookmark9)

### [系统设计的原则 21](#_bookmark8)

### [系统体系结构设计 21](#_bookmark8)

### [系统功能模块设计 22](#_bookmark10)

1. [**数据库设计 24**](#_bookmark11)

### [概念结构设计 24](#_bookmark12)

### [设计思路 24](#_bookmark12)

### [4.1.2 E-R 图 24](#_bookmark12)

### [4.1.3 主要实体属性图 25](#_bookmark13)

### [逻辑结构设计 27](#_bookmark14)

### [设计思路 27](#_bookmark14)

### [逻辑模型 27](#_bookmark14)

### [物理结构设计 28](#_bookmark15)

### [存取方式 28](#_bookmark15)

### [存储结构 28](#_bookmark15)

1. [**界面设计 32**](#_bookmark17)

### [界面关系图 32](#_bookmark16)

### [界面设计成果 32](#_bookmark16)

### [主界面 32](#_bookmark16)

### [子界面 34](#_bookmark18)

1. [**详细设计 40**](#_bookmark20)
   1. [系统主要功能模块介绍 40](#_bookmark19)
   2. [个人信息管理模块设计 40](#_bookmark19)
      1. [功能结构设计 40](#_bookmark19)
      2. [类图设计 41](#_bookmark21)
      3. [顺序图设计 42](#_bookmark22)
      4. [核心处理流程设计 42](#_bookmark22)
   3. [评分标准管理模块设计 43](#_bookmark23)
      1. [功能结构设计 43](#_bookmark23)
      2. [类图设计 44](#_bookmark24)
      3. [顺序图设计 45](#_bookmark25)
      4. [核心处理流程设计 46](#_bookmark26)
   4. [作品主题管理模块设计 46](#_bookmark26)
      1. [功能结构设计 46](#_bookmark26)
      2. [类图设计 47](#_bookmark27)
      3. [顺序图设计 48](#_bookmark28)
      4. [核心处理流程设计 49](#_bookmark29)
   5. [作品材料管理模块设计 49](#_bookmark29)
      1. [功能结构设计 49](#_bookmark29)
      2. [类图设计 50](#_bookmark30)
      3. [顺序图设计 51](#_bookmark31)
      4. [核心处理流程设计 52](#_bookmark32)
   6. [作品信息管理模块设计 53](#_bookmark33)
      1. [功能结构设计 53](#_bookmark33)
      2. [类图设计 53](#_bookmark33)
      3. [顺序图设计 55](#_bookmark34)
      4. [核心处理流程设计 56](#_bookmark35)
   7. [点赞和下载模块设计 56](#_bookmark35)
      1. [功能结构设计 56](#_bookmark35)
      2. [类图设计 57](#_bookmark36)
      3. [顺序图设计 58](#_bookmark37)
      4. [核心处理流程设计 59](#_bookmark38)
   8. [作品评分管理模块设计 59](#_bookmark38)
      1. [功能结构设计 59](#_bookmark38)
      2. [类图设计 60](#_bookmark39)
      3. [顺序图设计 61](#_bookmark40)
      4. [核心处理流程设计 62](#_bookmark41)
   9. [团队信息管理模块设计 62](#_bookmark41)
      1. [功能结构设计 62](#_bookmark41)
      2. [类图设计 63](#_bookmark42)
      3. [顺序图设计 64](#_bookmark43)
      4. [核心处理流程设计 65](#_bookmark44)

[7 编码 66](#_bookmark46)

* 1. [代码实现与核心算法 66](#_bookmark45)
  2. [代码优化分析 70](#_bookmark47)

[8 测试 73](#_bookmark48)

* 1. [测试方案设计 73](#_bookmark49)
     1. [测试策略 73](#_bookmark49)
     2. [测试进度安排 73](#_bookmark49)
     3. [测试资源 73](#_bookmark49)
     4. [关键测试点 74](#_bookmark50)
  2. [测试用例构建 74](#_bookmark50)
     1. [测试用例编写约定 74](#_bookmark50)
     2. [测试用例设计 75](#_bookmark51)
     3. [关键测试用例 76](#_bookmark52)
     4. [测试用例维护 78](#_bookmark53)

1. [总结与展望 79](#_bookmark55)
   1. [设计工作总结 79](#_bookmark54)
   2. [未来工作展望 79](#_bookmark54)

[谢 辞 80](#_bookmark56)

[参考文献 81](#_bookmark57)

[附录 A 外文翻译—原文部分 83](#_bookmark58)

[附录 B 外文翻译—译文部分 90](#_bookmark59)

[附录 C 软件使用说明书 95](#_bookmark60)

[附录 D 主要源代码 102](#_bookmark61)

# 1 绪 论

## 研究的背景及意义

### 选题的背景

随着互联网日新月异的发展，越来越多的事物通过互联网进行管理与展示，这其中就包括高校中企业实训的项目管理与展示，而以往的作品都是由学生或者老师自己管理，这样其实会耗费他们的大量精力以及财力，同时还无法实现实时监控，难以保证数据的准确性与及时性。并且其他人很少能系统全面观看到一些优秀的作品，这样就会缺少交流，容易造成学生没有树立榜样的目标，丧失学习的积极性，而且教师通过人工管理作品效率低下，因此一套合理的作品展示系统至关重要。

### 根据以上形势，管理与展示平台将可以获得极大的发展机会，将有越来越多的作品管理和展示系统面世。

### 国内外研究现状

### 自 2005 年以来，随着互联网大潮的变革趋势日益增强，作品管理类网站逐渐得到发展。网民可以自由选择适合其兴趣爱好的作品管理信息进行自主观看。本网站充分利用网络的便利性，可靠性强，成本低等优点逐步取代传统的报纸和杂志的方式，不仅减少了错误和工作量，而且大大提高了用户获取作品管理信息的效率。

Behance 是 Adobe 旗下的一个非常著名的设计社区网站，它于 2006 年在美国创立，多年来一直保持高质量高活跃度，创意设计人士可以在网站上发布并展示自己的作品，也可以查看别人的高质量作品，在这个社区里互动交流是非常活跃的，设计师们可以相互关注、相互私信、相互评论。Behance 的管理团队每天都会从各种领域中的顶级组合探索出新作品。这些领域包括设计、时尚、插图、工业设计、建筑、摄影、美术、广告、排版、动画、声效以及更多。领先的创意公司可以通过 Behance 发现人才，数百万的访客也可以使用Behance 跟踪最新和最杰出的创意人才。

### 站酷网是 2006 年 8 月在北京创立的综合性设计社区网站，我国有很多年轻的设计师、

设计艺术家、艺校师生总计逾 200 万在此网站上注册了账号，根据最新数据统计显示，该

### 网站每日更新有超过 7000 项原创设计作品，4 年来总共更新设计作品超过 500 万项。“站酷网”为设计师之间的交流与提升提供了一个理想的平台，并且设计师们的原创设计作品能得到更加广泛的推广与传播，这在国际上也增加了我们中国设计的知名度与影响力。“站酷网”一直致力于搭建企业与设计师之间的学习与沟通的桥梁，让优秀的人才有一个良好的环境来发挥他的才能，让企业也能更加简单地找到并网罗优秀人才，通过这种方式促进社会的进步。在 2018 年 2 月，美国 Shutterstock 公司宣布投资站酷网。

### 研究的意义

### 本文设计并实现了作品展示系统，通过此系统可以管理学生在项目开发时产生的作品以及材料，极大解放了学生本身与教师来人工管理作品和材料，同时可以做到实

时监控，学生一上传作品材料等信息教师就可以迅速收到，也保证了数据的准确性，而且系统前台可看到优秀的作品，可能其中的部分作品会对不同的人产生灵感，指导他人进行开发优秀的作品，从而提高学习的主动性。本系统分为前台和后台，前台展示优秀作品并提供阅读者进行浏览，点赞与下载，后台管理作品数据，从作品的主题开始，到作品的创建、团队组建资料上传、作品评价等。通过使用本系统，实现对实训过程中的作品的统一管理，提高相关人员的工作效率，实现系统的简单的操作处理更多的事务。

## 系统目标

### 在作品展示系统中，将用户分为评委、教师、开发团队、管理员等四个不同的身份，而每个不同的身份在系统中扮演不同的角色，对应每个模块的操作也不同。

1. 管理员能对主题进行增删改，以及状态管理，管理员可以设置评分标准并可对评分标准进行增删改查操作。

### 教师可审核上传的材料，对于不同开发团队所上传的材料根据其提交的质量选择通过或者拒绝。

1. 开发团队可以上传自己的作品以及材料，并拥有对其修改、删除的权限，开发团队可以在团队信息模块中对团队成员的信息进行增删改查。

### 评委可以对不同开发团队上传的项目使用某个标准进行评分，最后由系统自动统计出平均分。

1. 所有人员都可以在前台进行作品的查看、点赞、下载以及在后台进行个人信息的修改。
   1. **业务需求**

# 需求分析

### 个人信息管理模块，所有人员可以在登录系统后查看个人信息、修改个人信息、修改密码。

1. 评分标准管理模块，管理员可以在登录系统后查看评分标准、修改评分标准、删除评分标准、新增评分标准。

### 作品主题管理模块，管理员可以在登录系统后查看作品主题、修改作品主题、删除作品主题、新增作品主题。

1. 作品材料管理模块，开发团队可以在登录系统后查看作品材料、修改作品材料、删除作品材料、新增作品材料、上传作品材料，教师可以审核作品材料。

### 作品信息管理模块，开发团队可以在登录系统后查看作品信息、修改作品信息、删除作品信息、新增作品信息、上传作品、上传作品缩略图。

1. 作品评分管理模块，评委可以在登录系统后查看作品信息、选择评分标准为作品进行评分。

### 团队信息管理模块，开发团队可以在登录系统后查看团队信息、修改团队人员信息、删除团队人员信息、新增团队人员信息。

1. 点赞和下载模块，所有人员可以在登录系统后查看首页、直接浏览作品信息、分类浏览作品信息、进入详情页点赞作品、进入详情页下载作品。

### 主要业务流程

* + - 1. 个人信息管理活动图如图 2-1 所示，所有人员可以在登录系统后查看个人信息、修改个人信息、修改密码。

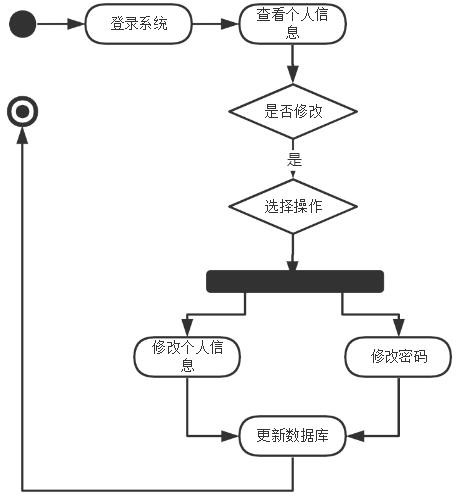


图 2-1 个人信息管理活动图

### 评分标准管理活动图如图 2-2 所示，管理员可以在登录系统后查看评分标准、修改评分标准、删除评分标准、新增评分标准。

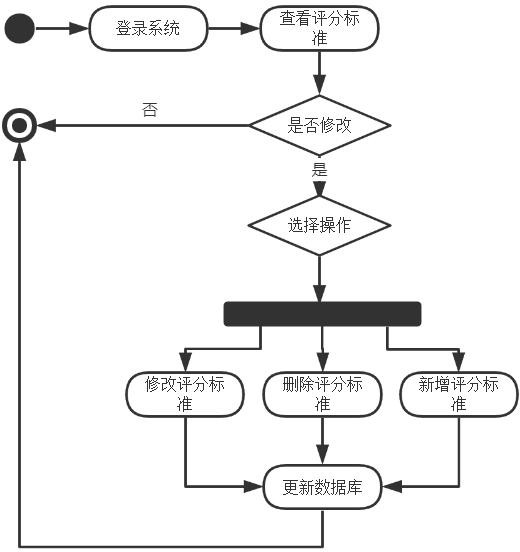


图 2-2 评分标准管理活动图

### 作品主题管理活动图如图 2-3 所示，管理员可以在登录系统后查看作品主题、修改作品主题、删除作品主题、新增作品主题。

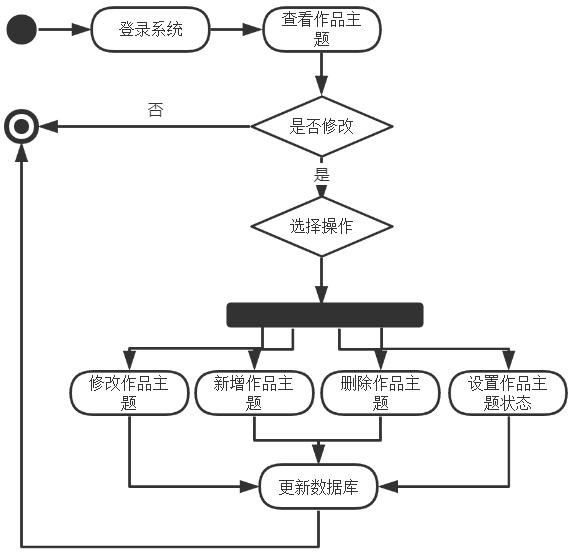


图 2-3 主题管理活动图

### 作品材料管理活动图如图 2-4 所示，开发团队可以在登录系统后查看作品材料、修改作品材料、删除作品材料、新增作品材料、上传作品材料，教师可以审核作品材料。

图 2-4 作品材料管理活动图

### 作品信息管理活动图如图 2-5 所示，开发团队可以在登录系统后查看作品信息、修改作品信息、删除作品信息、新增作品信息、上传作品、上传作品缩略图。

图 2-5 作品信息管理活动图

### 作品评分活动图如图 2-6 所示，评委可以在登录系统后查看作品信息、选择评分标准为作品进行评分。

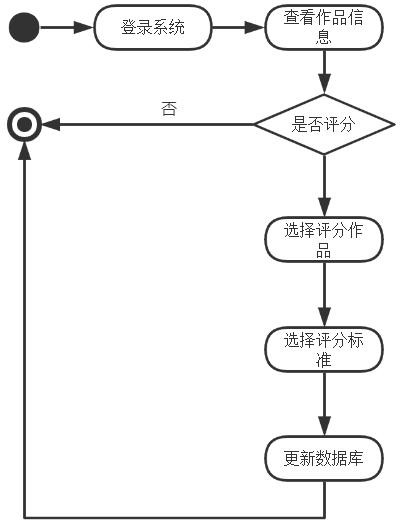


图 2-6 作品评分活动图

### 团队信息管理活动图如图 2-7 所示，开发团队可以在登录系统后查看团队信息、修改团队人员信息、删除团队人员信息、新增团队人员信息。

图 2-7 团队信息管理活动图

### （八）点赞和下载活动图如图 2-8 所示，所有人员可以在登录系统后查看首页、直接浏览作品信息、分类浏览作品信息、进入详情页点赞作品、进入详情页下载作品。

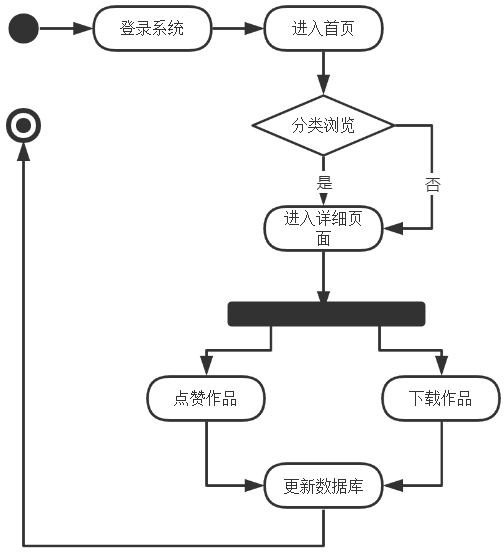


图 2-8 点赞和下载活动图

* 1. **功能需求**

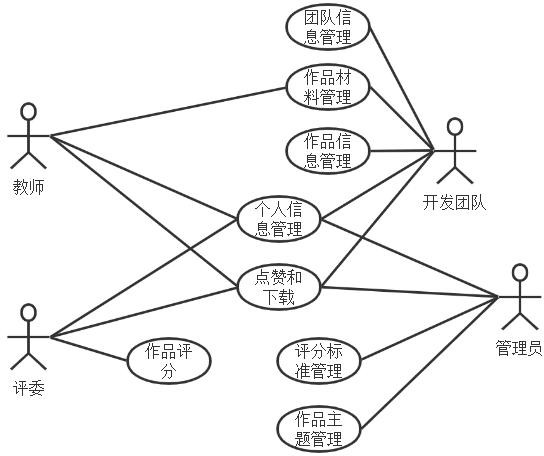
### 角色分析

以下从作品展示系统的实际需求分析，系统涉及到以下角色如表 2-1 所示。

表 2-1 系统角色表

|  |  |
| --- | --- |
| 角色 | 职责或功能 |
| 管理员 | 主要负责管理评分标准,作品主题,并且可以点赞和下载作品 |
| 评委 | 主要为作品在使用不同标准下给出评分,并且可以点赞和下载作品 |
| 开发团队 | 主要管理自己所开发的项目作品以及需要的材料,并且可以点赞和下载作  品 |
| 教师 | 主要审核开发团队所提交的材料是否合格,并且可以点赞和下载作品 |

### 业务功能

以下根据需求分析调研的结论，给出了作品展示系统的总体用例图，包含个人信息管理、评分标准管理、作品主题管理、作品材料管理、作品信息管理、团队信息管理、作品评分和点赞和下载等用例，系统总体用例图如图 2-9 所示。

### 个人信息管理

图 2-9 系统总体用例图

### 个人信息管理功能包括个人信息修改以及密码修改，其中个人信息修改为所有用户修改姓名、年龄、性别、电话号码等信息，修改密码为所有用户修改个人登录密码，个人信息管理用例图如图 2-10 所示。

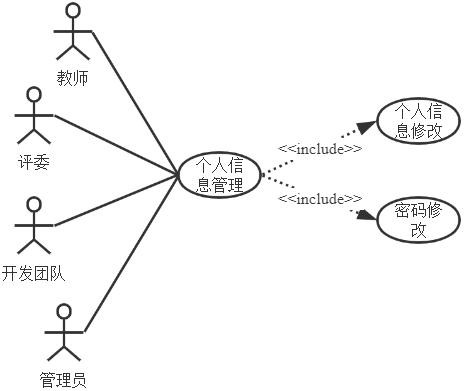


图 2-10 个人信息管理用例图

### 个人信息修改用例描述如表 2-2 所示。

表 2-2 个人信息修改用例描述

|  |  |
| --- | --- |
| 描述项 | 个人信息修改说明 |
| 用例名称 | 个人信息修改 |
| 标识符\* | YL01 |
| 用例描述 | 描述了使用本系统所有角色进行个人信息修改的过程 |
| 参与者表 | 所有角色 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 用户已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 用户在个人信息模块，进入个人信息页面； 2. 点击修改，修改相应的信息； 3. 修改后点击确定，修改成功； |
| 可选操作流 | 用户修改相应信息后，不想完成本次修改，点击取消按钮。 |

### 密码修改用例描述如表 2-3 所示。

表 2-3 密码修改用例描述

|  |  |
| --- | --- |
| 描述项 | 密码修改说明 |
| 用例名称 | 密码修改 |

续表 2-3

|  |  |
| --- | --- |
| 标识符\* | YL02 |
| 用例描述 | 描述了使用本系统所有角色进行密码修改的过程 |
| 参与者表 | 所有角色 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 用户已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 用户在个人信息模块，进入个人信息页面； 2. 点击修改，修改相应的信息； 3. 修改后点击确定，修改成功； |
| 可选操作流 | 用户修改密码后，不想完成本次修改，点击取消按钮。 |

### 评分标准管理

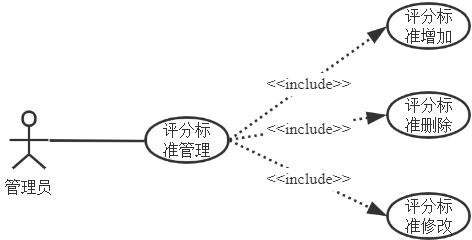
评分标准管理功能包括评分标准修改、评分标准增加、评分标准删除，评分标准包括名称、说明、备注、要求等内容，管理员可以在登录系统后查看评分标准、修改评分标准、删除评分标准、新增评分标准，评分标准管理用例图如图 2-11 所示。

图 2-11 评分标准管理用例图

### 评分标准增加用例描述如表 2-4 所示。

表 2-4 评分标准增加用例描述

|  |  |
| --- | --- |
| 描述项 | 评分标准增加说明 |
| 用例名称 | 评分标准增加 |
| 标识符\* | YL03 |
| 用例描述 | 描述了管理员使用本系统进行评分标准增加的过程 |
| 参与者表 | 管理员 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 管理员已登录系统 |

续表 2-4

|  |  |
| --- | --- |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 管理员在评分标准管理模块，进入评分标准管理页面； 2. 点击新增，新增相应的信息； 3. 新增后点击确定，新增成功； |
| 可选操作流 | 用户新增信息后，不想完成本次新增，点击取消按钮。 |

### 评分标准删除用例描述如表 2-5 所示。

表 2-5 评分标准删除用例描述

|  |  |
| --- | --- |
| 描述项 | 评分标准删除说明 |
| 用例名称 | 评分标准删除 |
| 标识符\* | YL04 |
| 用例描述 | 描述了管理员使用本系统进行评分标准删除的过程 |
| 参与者表 | 管理员 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 管理员已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 管理员在评分标准管理模块，进入评分标准管理页面； 2. 点击删除，删除相应的信息； 3. 删除后点击确定，删除成功； |
| 可选操作流 | 用户删除信息后，不想完成本次删除，点击取消按钮。 |

### 作品主题管理

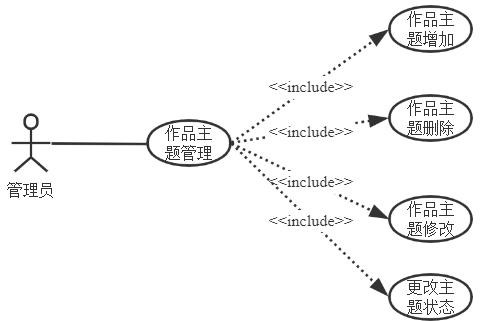
作品主题管理功能包括作品主题修改、作品主题增加、作品主题删除、更改主题状态， 作品主题包括名称、说明、备注、要求等内容，管理员可以在登录系统后查看作品主题、修改作品主题、删除作品主题、新增作品主题、更改主题状态，作品主题管理用例图如图2-12 所示。

图 2-12 作品主题管理用例图

### 作品主题修改用例描述如表 2-6 所示。

表 2-6 作品主题修改用例描述

|  |  |
| --- | --- |
| 描述项 | 作品主题修改说明 |
| 用例名称 | 作品主题修改 |
| 标识符\* | YL05 |
| 用例描述 | 描述了管理员使用本系统进行作品主题修改的过程 |
| 参与者表 | 管理员 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 管理员已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 管理员在作品主题管理模块，进入作品主题管理页面； 2. 点击修改，修改相应的信息； 3. 修改后点击确定，修改成功； |
| 可选操作流 | 管理员修改相应信息后，不想完成本次修改，点击取消按钮。 |

### 更改主题状态用例描述如表 2-7 所示。

表 2-7 更改主题状态用例描述

|  |  |
| --- | --- |
| 描述项 | 更改主题状态说明 |
| 用例名称 | 更改主题状态 |
| 标识符\* | YL06 |
| 用例描述 | 描述了管理员使用本系统进行更改主题状态的过程 |
| 参与者表 | 管理员 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 管理员已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 管理员在作品主题管理模块，进入作品主题管理页面； 2. 点击可用或禁用按钮，更改相应的状态； |
| 可选操作流 | 无 |

### 作品材料管理

作品材料管理功能包括作品材料修改、作品材料增加、作品材料删除、上传缩略图、上传材料，作品材料包括名称、出处、作者、所属作品、所属团队、材料文件、审核教师等内容，开发团队可以在登录系统后查看作品材料、修改作品材料、删除作品材料、新增作品材料、上传材料，教师可以在登录系统后查看作品材料、评审材料，作品材料管理用例图如图 2-13 所示。

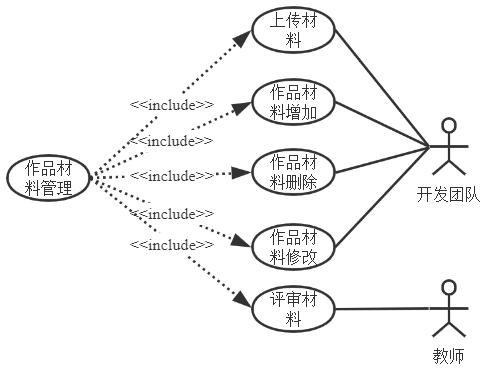


图 2-13 作品材料管理用例图

### 上传材料用例描述如表 2-8 所示。

表 2-8 上传材料用例描述

|  |  |
| --- | --- |
| 描述项 | 上传材料说明 |
| 用例名称 | 上传材料 |
| 标识符\* | YL07 |
| 用例描述 | 描述了开发团队使用本系统进行上传材料的过程 |
| 参与者表 | 开发团队 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 开发团队已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 开发团队在作品材料管理模块，进入作品材料管理页面； 2. 点击上传，上传相应的材料； 3. 上传后点击确定，上传成功； |
| 可选操作流 | 开发团队上传材料后，不想完成本次上传，点击取消按钮。 |

### 作品材料评审用例描述如表 2-9 所示。

表 2-9 作品材料评审用例描述

|  |  |
| --- | --- |
| 描述项 | 作品材料评审说明 |
| 用例名称 | 材料评审 |
| 标识符\* | YL08 |
| 用例描述 | 描述了教师使用本系统进行作品材料评审的过程 |

续表 2-9

|  |  |
| --- | --- |
| 参与者表 | 教师 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 教师已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 教师在作品材料管理模块，进入作品材料管理页面； 2. 点击通过或拒绝，审核相应的材料； 3. 审核成功刷新页面； |
| 可选操作流 | 无 |

### 作品信息管理

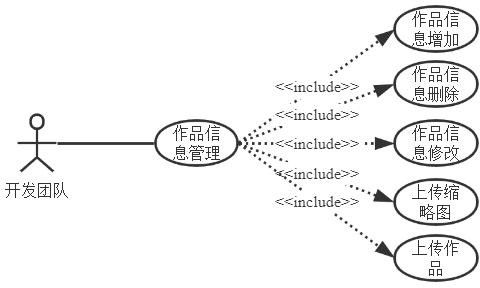
作品信息管理功能包括作品信息修改、作品信息增加、作品信息删除、上传缩略图、上传作品，作品信息包括名称、文件、简介、缩略图、备注、点赞数、说明、作品主题、团队、下载数、总分等内容，开发团队可以在登录系统后查看作品信息、修改作品信息、删除作品信息、新增作品信息、上传缩略图、上传作品，作品信息管理用例图如图 2-14 所示。

图 2-14 作品信息管理用例图

### 上传缩略图用例描述如表 2-10 所示。

表 2-10 上传缩略图用例描述

|  |  |
| --- | --- |
| 描述项 | 上传缩略图说明 |
| 用例名称 | 上传缩略图 |
| 标识符\* | YL09 |
| 用例描述 | 描述了开发团队使用本系统进行上传缩略图的过程 |
| 参与者表 | 开发团队 |
| 优先级 | 1 |

续表 2-10

|  |  |
| --- | --- |
| 状态\* | 进行中 |
| 前置条件 | 开发团队已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 开发团队在作品信息管理模块，进入作品信息管理页面； 2. 点击上传，上传相应的缩略图； 3. 上传后点击确定，上传成功； |
| 可选操作流 | 开发团队上传缩略图后，不想完成本次上传，点击取消按钮。 |

### 上传作品用例描述如表 2-11 所示。

表 2-11 上传作品用例描述

|  |  |
| --- | --- |
| 描述项 | 上传作品说明 |
| 用例名称 | 上传作品 |
| 标识符\* | YL10 |
| 用例描述 | 描述了开发团队使用本系统进行上传作品的过程 |
| 参与者表 | 开发团队 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 开发团队已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 开发团队在作品信息管理模块，进入作品信息管理页面； 2. 点击上传，上传相应的作品； 3. 上传后点击确定，上传成功； |
| 可选操作流 | 开发团队上传作品后，不想完成本次上传，点击取消按钮。 |

### 点赞和下载

点赞与下载功能包括作品点赞、作品下载，本系统所有用户都可以在登录系统后对作品进行点赞与作品下载，点赞与下载用例图如图 2-15 所示。

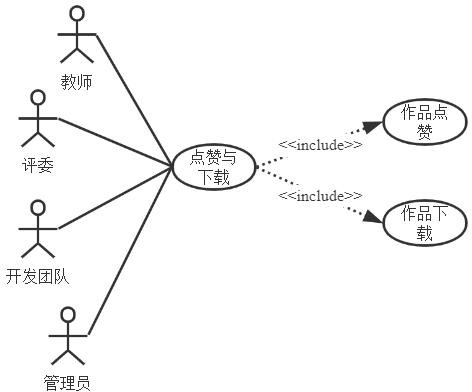


图 2-15 点赞与下载用例图

### 作品点赞用例描述如表 2-12 所示。

表 2-12 作品点赞用例描述

|  |  |
| --- | --- |
| 描述项 | 作品点赞说明 |
| 用例名称 | 作品点赞 |
| 标识符\* | YL11 |
| 用例描述 | 描述了使用本系统所有角色进行作品点赞的过程 |
| 参与者表 | 所有角色 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 用户已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 用户进入作品列表主页，再进入作品详情页面； 2. 点击点赞，点赞相应的作品； 3. 点赞后弹出提示，点赞成功； |
| 可选操作流 | 无 |

### 作品下载用例描述如表 2-13 所示。

表 2-13 作品下载用例描述

|  |  |
| --- | --- |
| 描述项 | 作品下载说明 |
| 用例名称 | 作品下载 |
| 标识符\* | YL12 |
| 用例描述 | 描述了使用本系统所有角色进行作品下载的过程 |
| 参与者表 | 所有角色 |

续表 2-13

|  |  |
| --- | --- |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 用户已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 用户进入作品列表主页，再进入作品详情页面； 2. 点击下载，下载相应的作品； 3. 下载后弹出提示，下载成功； |
| 可选操作流 | 无 |

### 作品评分管理

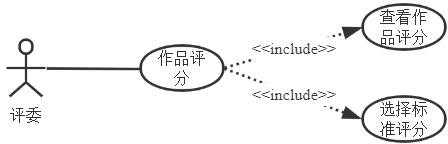
作品评分管理功能包括查看作品评分、选择标准评分，评委可以在登录系统后查看作品信息及评分、选择评分标准对作品评分，作品评分管理用例图如图 2-16 所示。

图 2-16 作品评分管理用例图

### 选择标准评分用例描述如表 2-14 所示。

表 2-14 选择标准评分用例描述

|  |  |
| --- | --- |
| 描述项 | 选择标准评分说明 |
| 用例名称 | 选择标准评分 |
| 标识符\* | YL13 |
| 用例描述 | 描述了评委使用本系统进行选择标准评分的过程 |
| 参与者表 | 评委 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 评委已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 评委在作品信息模块，再进入作品信息管理页面； 2. 点击评分，选择标准对相应的作品评分； 3. 评分后点击确定，评分成功； |
| 可选操作流 | 评委评分相应作品时，不想完成本次评分，点击取消按钮。 |

### 团队信息管理

团队信息管理功能包括团队人员修改、团队人员增加、团队人员删除，团队人员包括

### 姓名、专业、工作、年龄等内容，开发团队可以在登录系统后查看团队人员、修改团队人员、删除团队人员、新增团队人员，团队信息管理用例图如图 2-17 所示。

图 2-17 团队信息管理用例图

### 团队人员增加用例描述如表 2-15 所示。

表 2-15 团队人员增加用例描述

|  |  |
| --- | --- |
| 描述项 | 团队人员增加说明 |
| 用例名称 | 团队人员增加 |
| 标识符\* | YL14 |
| 用例描述 | 描述了开发团队使用本系统进行团队人员增加的过程 |
| 参与者表 | 开发团队 |
| 优先级 | 1 |
| 状态\* | 进行中 |
| 前置条件 | 开发团队已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 开发团队在团队信息管理模块，进入团队信息管理页面； 2. 点击新增，新增相应的信息； 3. 新增后点击确定，新增成功； |
| 可选操作流 | 开发团队新增信息后，不想完成本次新增，点击取消按钮。 |

### 团队人员删除用例描述如表 2-16 所示。

表 2-16 团队人员删除用例描述

|  |  |
| --- | --- |
| 描述项 | 团队人员删除说明 |
| 用例名称 | 团队人员删除 |
| 标识符\* | YL15 |
| 用例描述 | 描述了开发团队使用本系统进行团队人员删除的过程 |
| 参与者表 | 开发团队 |
| 优先级 | 1 |

续表 2-16

|  |  |
| --- | --- |
| 状态\* | 进行中 |
| 前置条件 | 开发团队已登录系统 |
| 后置条件 | 系统给出操作成功提示 |
| 基本操作流 | 1. 开发团队在团队信息管理模块，进入团队信息管理页面； 2. 点击删除，删除相应的信息； 3. 删除后点击确定，删除成功； |
| 可选操作流 | 开发团队删除信息后，不想完成本次删除，点击取消按钮。 |

## 非功能需求

### 环境需求

开发环境需求如表 2-17 所示。

表 2-17 开发环境需求表

|  |  |
| --- | --- |
| 要求 | 内容 |
| 操作系统 | Windows 10 1903 及以上版本 |
| JDK | 9.0.1 及其以上版本 |
| 开发工具 | IDEA2019 及以上版本 |
| 浏览器 | IE、Microsoft Edge、Chrome、Firefox |
| CPU | Intel Core i5-4210M 处理器及以上配置 |
| 内存 | 4GB(2G\*2) DDR3L-1333 及以上配置 |

### 服务器环境需求如表 2-18 所示。

表 2-18 服务器环境需求表

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

### 客户端运行环境需求如表 2-19 所示。

表 2-19 客户端运行环境需求表

|  |  |
| --- | --- |
| 要求 | 内容 |
| CPU | Intel Core i5-4210M 处理器及以上配置 |
| 内存 | 4GB(2G\*2) DDR3L-1333 及以上配置 |
| 硬盘 | 256G SSD 及以上配置 |
| 显卡 | NVIDIA GT940M 2GB 及以上配置 |

### 性能需求

1. 速度(响应时间)

### 为给用户呈现良好的交互性，作品展示系统执行每一个请求从开始到数据返回的总体响应时间不超过 3 秒。

1. 吞吐量

### 在数据库方面，本系统中数据库在 5 秒内执行任意 SQL 语句不低于 10 条。

1. 用户并发数

### 本系统需要保证在 10 个用户同时同一个操作时，系统能够容纳这些请求同时发生， 并且正常快速响应不出错。

### 安全需求

### 系统保密性

在系统中，对于每一个特定的功能，只有特定的用户能够进行修改等操作，还需要防止信息的非法泄露。

### 可用性和抗毁性

在系统中，如果出现单点失败，则此时备份机制和容错机制应当介入，从而保证系统不出错。

### 系统完整性

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

* 1. **系统设计的原则**

# 总体设计

### 系统设计总的原则是保证系统设计目标的实现，就是回答”概括地说，系统应当如何实现”这个问题，站在全局的角度上，用更低的成本，从比较抽象的层次上分析出最适合系统方案和软件的结构，最后高质量地去实现它。在系统设计中，应遵循以下原则。

* 1. 明确性原则

### 首先最重要的是系统的目标须是明确的，系统的边界应当清晰，不可以自行放大或者缩小系统的需求。

* 1. 实用性原则

### 实用性原则是指开发者在开发本系统时要以技术合理使用，功能合适为原则。

* 1. 可行性原则

### 可行性原则是指在综合考量人力、物力、财力等多方面因素来决定方案是否可行，如果方案可行，即表明方案是科学的，如果综合各方面因素发现方案不合理，那就说明方案是不可行的。

* 1. 扩展性原则

可扩展性是指软件扩展系统的能力，在编码阶段，设计良好的一套代码可以让更多的功能加入到合适的位置，这样做可以应对未来开发与维护中避免代码被过度工程化开发。

* 1. **系统体系结构设计**

### 本系统采用的系统体系结构是 B/S 结构。B/S 结构，也称为浏览器/服务器结构，是目前应用程序的发展方向。B/S 结构是在网络技术的不断发展过程中对 C/S 架构进行的一种改良，为了和传统的 C/S 结构模式进行区分，称之为 B/S 结构模式。在 B/S 结构下，通过网络浏览器来进入程序的主界面，系统中极少的事务请求在网络浏览器前台处理，大部分的事务请求在服务器端进行处理，这样就出现了一种特殊的三层结构。这样可以使得用户电脑的负荷极大降低（因此被称为瘦客户端），降低了系统维护与升级的成本，也减轻了客户的综合成本。为提高代码的重用性和程序的规范性，本系统使用 SSM 框架进行开发， 即 SpringMVC、Spring 与 MyBatis 三个框架。

它们在三层架构中所处的位置是不同的，即它们在三层架构中的功能各不相同，各司其职。

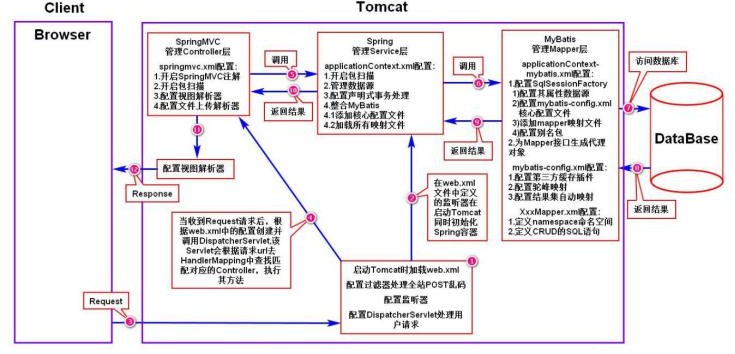
### SpringMVC：它作为视图层的实现者，完成接收用户请求的功能。SpringMVC 的

Controller 作为整个应用的控制器，完成用户的转发请求并对用户进行响应。

### MyBatis：它作为数据访问对象层的实现者，完成对数据库的新增、删除、修改、查询等操作。

Spring：它以一个应用管家的身份存在。整个应用中所有 Bean 的生命周期行为，都由 Spring 来进行管理。

系统架构图如图 3-1 所示。



## 系统功能模块设计

图 3-1 系统架构图

### 本系统的系统总包图如图 3-2 所示。

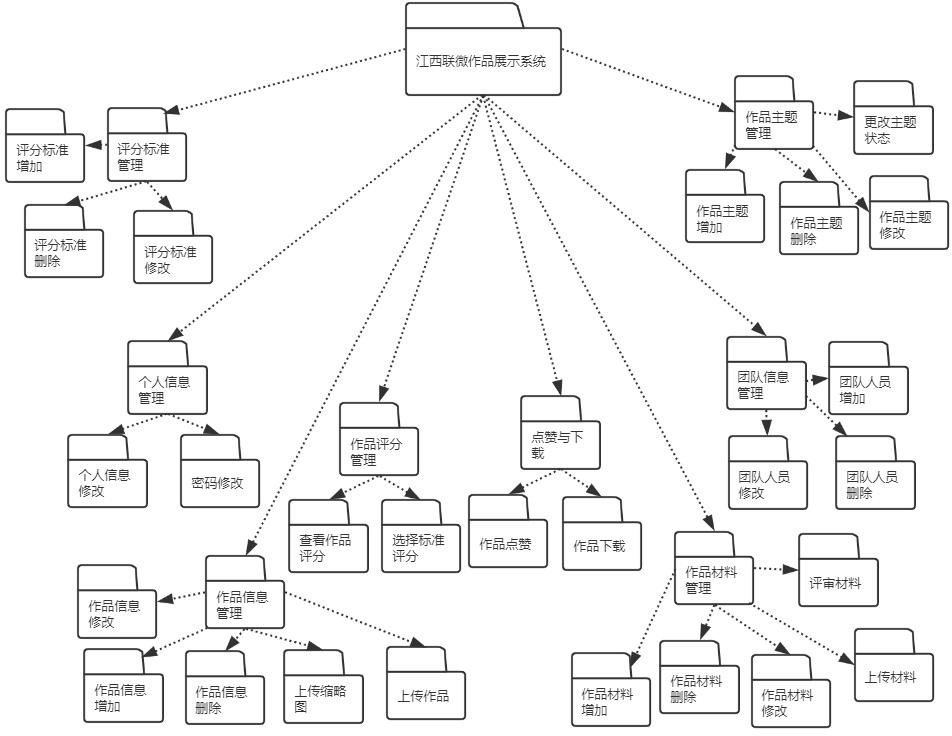


图 3-2 系统总包图

### 如图 3-2 所示，作品展示系统一共分为八个模块：个人信息管理模块、评分标准管理模块、作品主题管理模块、作品评分管理模块、作品信息管理模块、作品材料管理模块、点赞与下载模块、团队信息管理模块。其中个人信息管理模块包含个人信息修改以及密码修改，评分标准管理模块分为评分标准修改、评分标准增加、评分标准删除，作品主题管理模块分为作品主题修改、作品主题增加、作品主题删除、更改主题状态，作品评分管理模块分为查看作品评分、选择标准评分，作品信息管理模块分为作品信息修改、作品信息增加、作品信息删除、上传缩略图、上传作品，作品材料管理模块分为材料修改、作品材料增加、作品材料删除、上传缩略图、上传材料，点赞与下载模块分为作品点赞、作品下载。

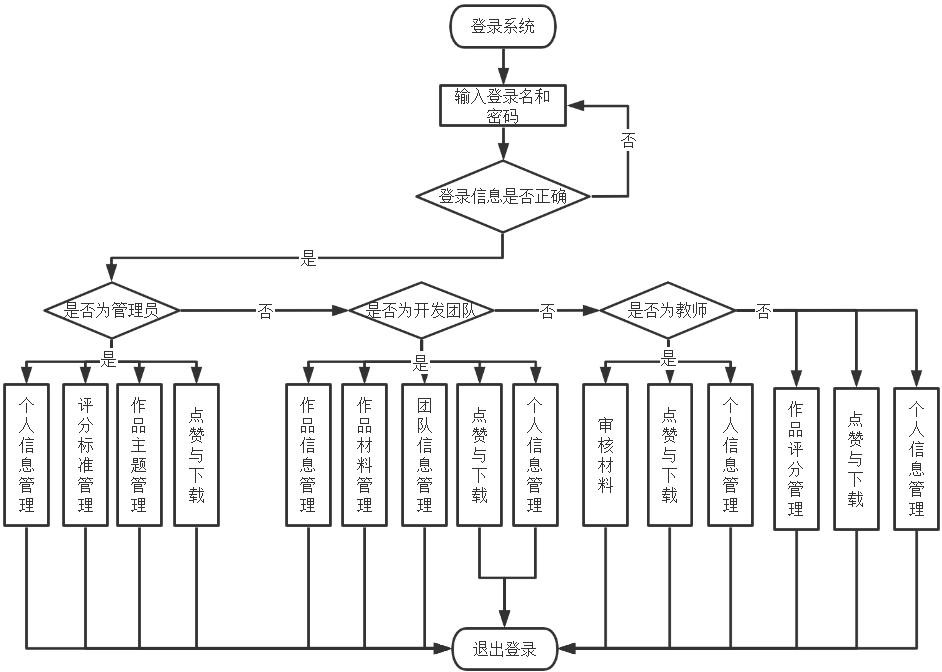
本系统的系统总体流程图如图 3-3 所示。

图 3-3 系统总体流程图

# 数据库设计

## 概念结构设计

### 设计思路

作品展示系统的实体包括管理员、教师、评委、开发团队、团队人员、作品、作品主题、评分标准、作品材料等。从一开始管理员发布若干个主题，制定若干个评分标准，然后开发团队选择一个主题进行相应的开发，开发团队在开发的同时会产生相应的材料，比如论文等相关文档，而这些论文由教师进行评审，一旦材料不合格可以给予不通过，最后开发出来的作品由评委根据评分标准来进行打分。

### 由以上分析可知，在“发布”关系中，管理员实体与主题实体是 1:N 的关系。在“制定” 关系中，管理员实体与评分标准实体之间是 1:N 的关系。在“审核”关系中，教师实体与作品材料实体之间是 1:N 的关系。在“管理”关系中，开发团队实体与团队人员实体之间是 1:N 的关系。在“上传”关系中，开发团队与作品之间是 1:N 的关系。在“选择”关系中，作品主题与作品之间是 1:N 的关系。在“使用”关系中，评分标准与作品之间是 M:N 的关系。在“评审”关系中，评委与作品之间是 M:N 的关系。

### E-R 图

### 作品展示系统的总体 E-R 图如下图 4-1 所示。

图 4-1 系统总体 E-R 图

### 主要实体属性图

* + - 1. 用户的实体属性主要包括用户编号、用户登录名、用户密码、用户姓名、用户电话、用户年龄、用户类型、创建时间，用户的实体图如图 4-2 所示。

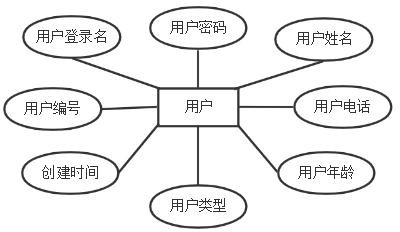


图 4-2 用户 E-R 图

### 团队人员的实体属性主要包括人员编号、人员姓名、团队编号、人员专业、人员工作、人员年龄、创建时间，团队人员的实体图如图 4-3 所示。

图 4-3 团队人员 E-R 图

### 评分标准的实体属性主要包括标准编号、标准名称、标准备注、标准要求、标准说明、创建时间，评分标准的实体图如图 4-4 所示。

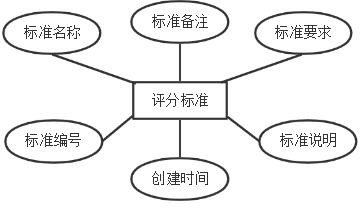


图 4-4 评分标准 E-R 图

### 作品主题的实体属性主要包括主题编号、主题名称、主题备注、主题要求、主题说明、主题类型、创建时间，作品主题的实体图如图 4-5 所示。

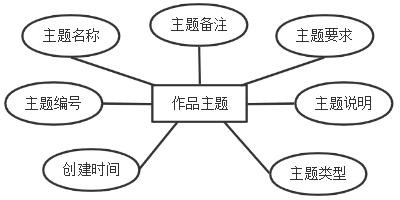


图 4-5 作品主题 E-R 图

### 作品信息的实体属性主要包括作品编号、作品名称、作品文件、作品简介、缩略图地址、作品文件名、作品备注、点赞数、作品说明、缩略图名称、主题编号、团队编号、作品总分、下载数、创建时间，作品信息的实体图如图 4-6 所示。

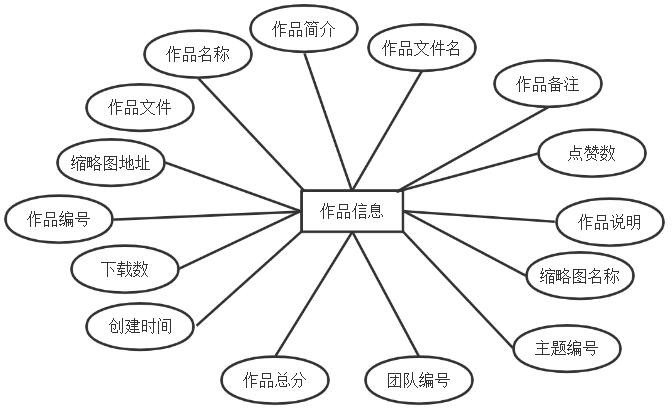


图 4-6 作品信息 E-R 图

### 作品材料的实体属性主要包括材料编号、材料名称、作品出处、作品作者、作品编号、团队编号、材料文件、教师编号、创建时间，作品材料的实体图如图 4-7 所示。

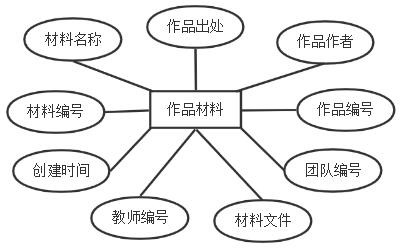


图 4-7 作品材料 E-R 图

### 作品评分的实体属性主要包括评分编号、作品编号、主题编号、评委编号、标准编号、团队编号、评分、创建时间，作品评分的实体图如图 4-8 所示。

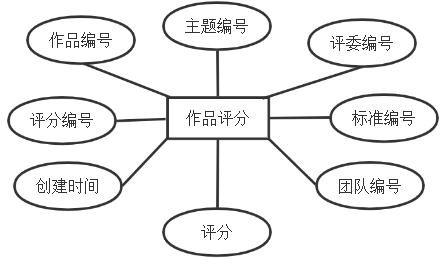


图 4-8 作品评分 E-R 图

## 逻辑结构设计

### 设计思路

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

### 逻辑模型

由关系 E-R 图得到的关系模型如下所示：

### 用户 (用户编号、用户登录名、用户密码、用户姓名、用户电话、用户年龄、用户类型、创建时间)

* + - 1. 团队人员 (人员编号、人员姓名、团队编号、人员专业、人员工作、人员年龄、创建时间)

### 评分标准 (标准编号、标准名称、标准备注、标准要求、标准说明、创建时间)

* + - 1. 作品主题 (主题编号、主题名称、主题备注、主题要求、主题说明、主题类型、创建时间)

### 作品信息 (作品编号、作品名称、作品文件、作品简介、缩略图地址、作品文件名、作品备注、点赞数、作品说明、缩略图名称、主题编号、团队编号、作品总分、下载数、创建时间)

* + - 1. 作品材料 (材料编号、材料名称、作品出处、作品作者、作品编号、团队编号、材料文件、教师编号、创建时间)

### 作品评分 (评分编号、作品编号、主题编号、评委编号、标准编号、团队编号、评分、创建时间)

根据以上主要关系模型设计了总共 9 张表来处理本系统模块功能，其中 7 张数据表为

### 关键数据表，数据总表如表 4-1 所示。

表 4-1 系统涉及数据表

|  |  |
| --- | --- |
| 表名 | 功能说明 |
| user | 用户表 |
| zhuti | 主题表 |
| group | 团队人员表 |
| cailiao | 作品材料表 |
| zuopin | 作品信息表 |
| biaozhun | 评分标准表 |
| pingfen | 评分表 |
| admin | 管理员表 |
| tuandui | 开发团队表 |

## 物理结构设计

### 存取方式

存取方式是快速存取数据库中的数据的技术，数据库管理系统一般都提供多种存取方法。主要是三种分别为 B+树索引存取、Hash 索引存取、聚簇存取。

### 存储结构

本系统采用 MySQL 数据库进行存储。B+树索引存取是按照顺序存储的，所以他返回的数据是顺序的。Hash 索引存取在用 Hash 值找到后还需要再通过指针才能找到数据，因此它返回的结果是随机的。

### 确定数据的存放在磁盘中位置以及数据的存储结构需要充分考虑存取效率、空间利用率和维护成本等多个方面的因素。

1. 用户表

### 用户表中字段主包括用户编号、用户登录名、用户密码、用户姓名、用户电话、用户年龄、用户类型、创建时间，具体表结构设计如表 4-2 所示：

表 4-2 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键否 | 备注 |
| id | int | 11 | 主键 | 用户编号 |
| login\_name | varchar | 255 | 否 | 用户登录名 |
| password | varchar | 255 | 否 | 用户密码 |
| name | varchar | 255 | 否 | 用户姓名 |
| phone | varchar | 255 | 否 | 用户电话 |
| age | int | 11 | 否 | 用户年龄 |
| type | int | 11 | 否 | 用户类型 |
| create\_time | datetime |  | 否 | 创建时间 |

### 团队人员表

团队人员表中字段主包括人员编号、人员姓名、团队编号、人员专业、人员工作、人员年龄、创建时间，具体表结构设计如表 4-3 所示：

表 4-3 团队人员表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键否 | 备注 |
| id | int | 11 | 主键 | 人员编号 |
| name | varchar | 255 | 否 | 人员姓名 |
| group\_id | int | 11 | 否 | 团队编号 |
| major | varchar | 255 | 否 | 人员专业 |
| work | varchar | 255 | 否 | 人员工作 |
| age | int | 11 | 否 | 人员年龄 |
| create\_time | datetime |  | 否 | 创建时间 |

### 评分标准表

评分标准表中字段主包括标准编号、标准名称、标准备注、标准要求、标准说明、创建时间，具体表结构设计如表 4-4 所示：

表 4-4 评分标准表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键否 | 备注 |
| id | int | 11 | 主键 | 标准编号 |
| name | varchar | 255 | 否 | 标准名称 |
| beizhu | varchar | 255 | 否 | 标准备注 |
| yaoqiu | varchar | 255 | 否 | 标准要求 |
| shuoming | varchar | 255 | 否 | 标准说明 |
| create\_time | datetime |  | 否 | 创建时间 |

### 作品主题表

作品主题表中字段主包括主题编号、主题名称、主题备注、主题要求、主题说明、主题类型、创建时间，具体表结构设计如表 4-5 所示：

表 4-5 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键否 | 备注 |
| id | int | 11 | 主键 | 主题编号 |
| name | varchar | 255 | 否 | 主题名称 |

续表 4-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| beizhu | varchar | 255 | 否 | 主题备注 |
| yaoqiu | varchar | 255 | 否 | 主题要求 |
| shuoming | varchar | 255 | 否 | 主题说明 |
| leixing | int | 11 | 否 | 主题类型 |
| create\_time | datetime |  | 否 | 创建时间 |

### 作品信息表

作品信息表中字段主包括作品编号、作品名称、作品文件、作品简介、缩略图地址、作品文件名、作品备注、点赞数、作品说明、缩略图名称、主题编号、团队编号、作品总分、下载数、创建时间，具体表结构设计如表 4-6 所示：

表 4-6 作品信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键否 | 备注 |
| id | int | 11 | 主键 | 作品编号 |
| name | varchar | 255 | 否 | 作品名称 |
| doc\_src | varchar | 255 | 否 | 作品文件 |
| introduce | varchar | 2550 | 否 | 作品简介 |
| pic\_addr | varchar | 255 | 否 | 缩略图地址 |
| doc\_name | varchar | 255 | 否 | 作品文件名 |
| beizhu | varchar | 255 | 否 | 作品备注 |
| create\_time | datetime |  | 否 | 创建时间 |
| dz\_num | int | 11 | 否 | 点赞数 |
| shuoming | varchar | 255 | 否 | 作品说明 |
| pic\_name | varchar | 255 | 否 | 缩略图名称 |
| zhuti\_id | int | 11 | 否 | 主题编号 |
| group\_id | int | 11 | 否 | 团队编号 |
| sum\_grade | double |  | 否 | 作品总分 |
| download\_num | int | 11 | 否 | 下载数 |

### 作品材料表

作品材料表中字段主包括材料编号、材料名称、作品出处、作品作者、作品编号、团队编号、材料文件、教师编号、创建时间，具体表结构设计如表 4-7 所示：

表 4-7 作品材料表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键否 | 备注 |
| id | int | 11 | 主键 | 材料编号 |
| name | varchar | 255 | 否 | 材料名称 |
| chuchu | varchar | 255 | 否 | 作品出处 |
| author | varchar | 255 | 否 | 作品作者 |
| zuoping\_id | int | 11 | 否 | 作品编号 |
| group\_id | int | 11 | 否 | 团队编号 |
| cailiao\_src | varchar | 255 | 否 | 材料文件 |
| create\_time | datetime |  | 否 | 创建时间 |
| teacher\_id | int | 11 | 否 | 教师编号 |

### 作品评分表

作品评分表中字段主包括评分编号、作品编号、主题编号、评委编号、标准编号、团队编号、评分、创建时间，具体表结构设计如表 4-8 所示：

表 4-8 作品评分表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键否 | 备注 |
| id | int | 11 | 主键 | 评分编号 |
| zuoping\_id | int | 11 | 否 | 作品编号 |
| zhuti\_id | int | 11 | 否 | 主题编号 |
| pingwei\_id | int | 11 | 否 | 评委编号 |
| biaozhun\_id | int | 11 | 否 | 标准编号 |
| group\_id | int | 11 | 否 | 团队编号 |
| grade | double |  | 否 | 评分 |
| create\_time | datetime |  | 否 | 创建时间 |

* 1. **界面关系图**

# 界面设计

### 作品展示系统从浏览器输入指定 URL 即可进入网站首页，首页一开始可根据按钮到作品展示页，通过选择可分类展示，列表页点击每一项可进入详情页，可以在首页点击登录进入登录页，如果是第一次登录则需要注册，注册成功进入登录页， 登录成功可进入后台主页，后台主页所有的角色能访问的总共有 7 个不同的页面，分别为个人信息页、作品主题页、作品信息页、作品材料页、评分标准页、作品评分页、团队信息页，但是不同角色会访问到不同个数的页面。界面关系图如图 5-1 所示。

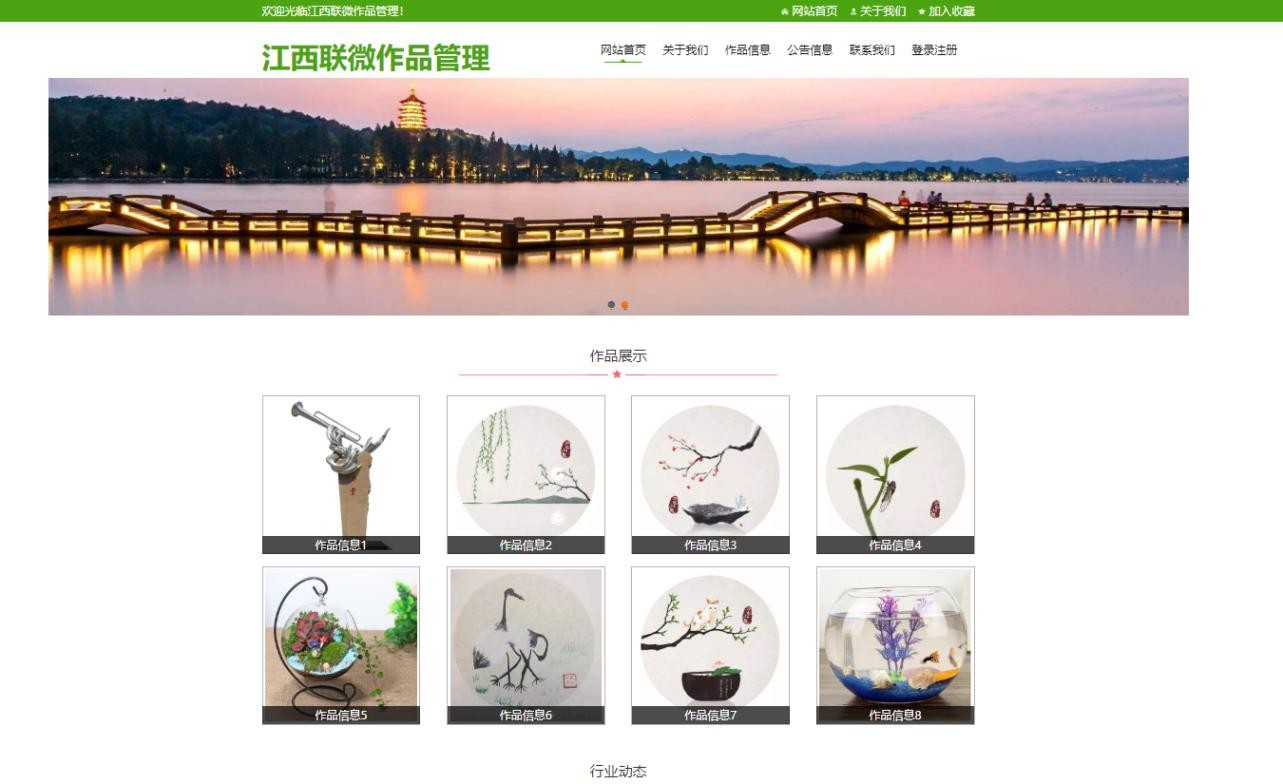
图 5-1 界面关系图

* 1. **界面设计成果**

### 主界面

1. 用户主界面

### 本系统通过输入指定 URL 即可进入网站主界面，网站主界面有作品的展示，可以选择进行分类展示，每一个作品都可以点击进行查看作品详情，所以用户还可以在这里选择登录或者注册， 主界面分三部分， 分别为头部(header)、尾部(footer)、内容主体(mainbox)，如图 5-2 所示。



1. 管理员主界面

图 5-2 网站主界面图

### 管理员登录后即可进入管理员主界面，可以展开每个列表框进入不同的管理页面，左部为可展开的导航菜单，右部为功能操作页，这两个部分属于两个盒子，外面是一个大盒子嵌套，如图 5-3 所示。

1. 非管理员主界面

图 5-3 管理员主界面图

### 非管理员登录后即可进入非管理员主界面，其他角色的界面差不多，不同之处在于他们可以操作的功能不同，如图 5-4 所示。

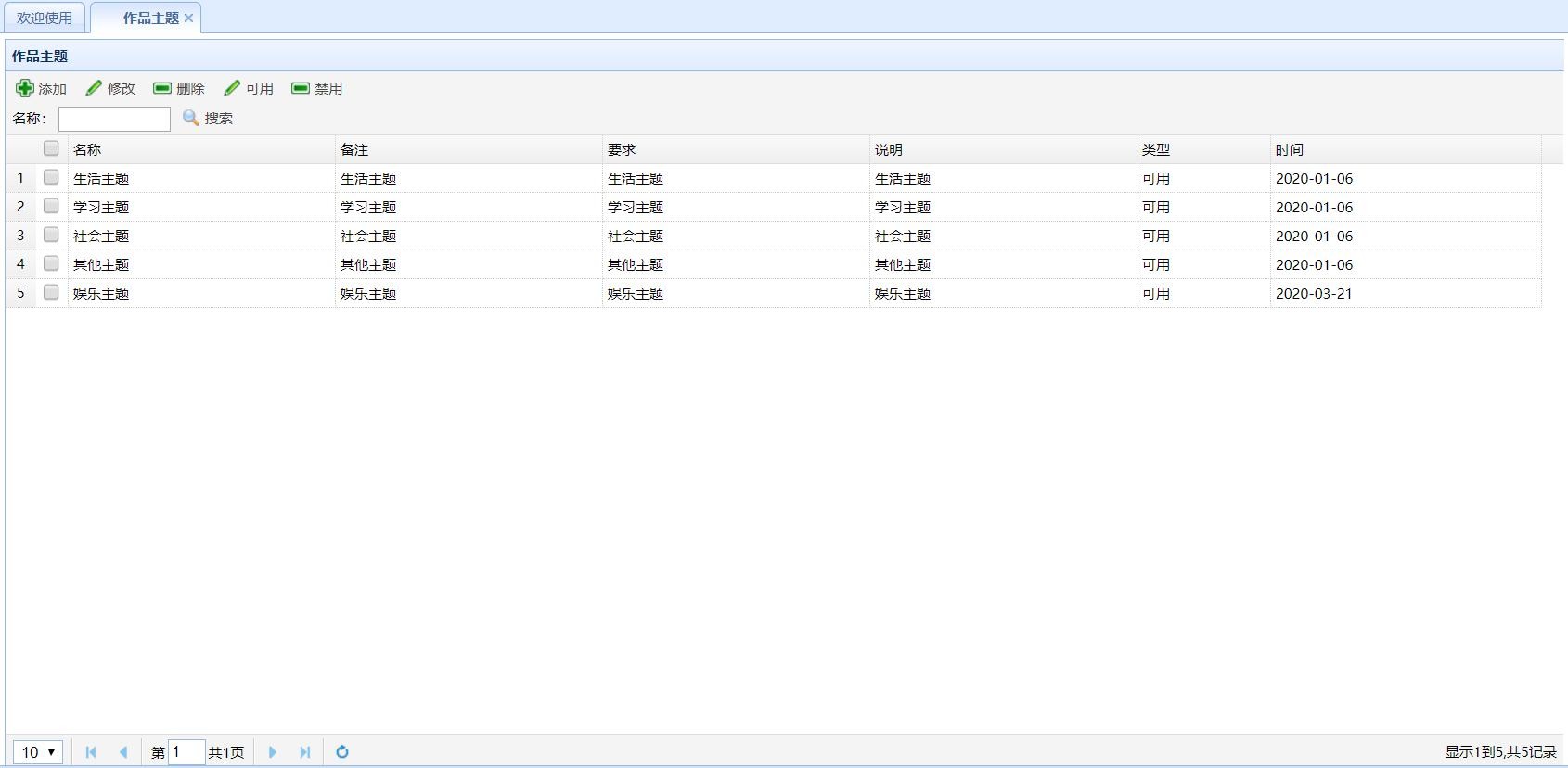


图 5-4 非管理员主界面图

### 子界面

1. 作品主题界面

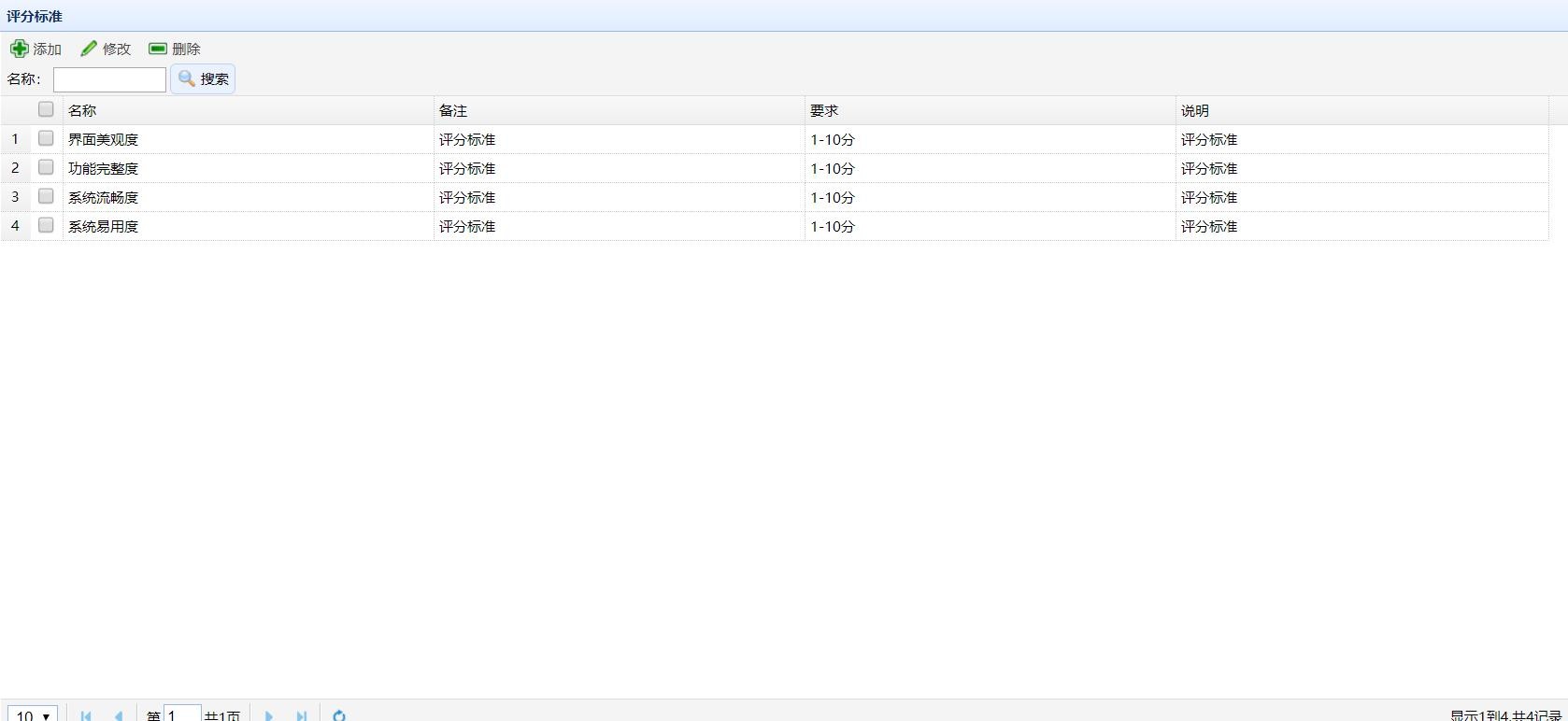
### 本系统登录即进入作品主题界面，在这里可以点击添加按钮、修改按钮、删除按钮等进行相应的操作，点击添加和修改按钮后会弹出操作框，如图 5-5 所示。



1. 评分标准界面

图 5-5 作品主题界面图

### 本系统登录即进入评分标准界面，可以点击添加按钮、修改按钮、删除按钮等进行相应的的操作，如图 5-6 所示。



1. 个人信息界面

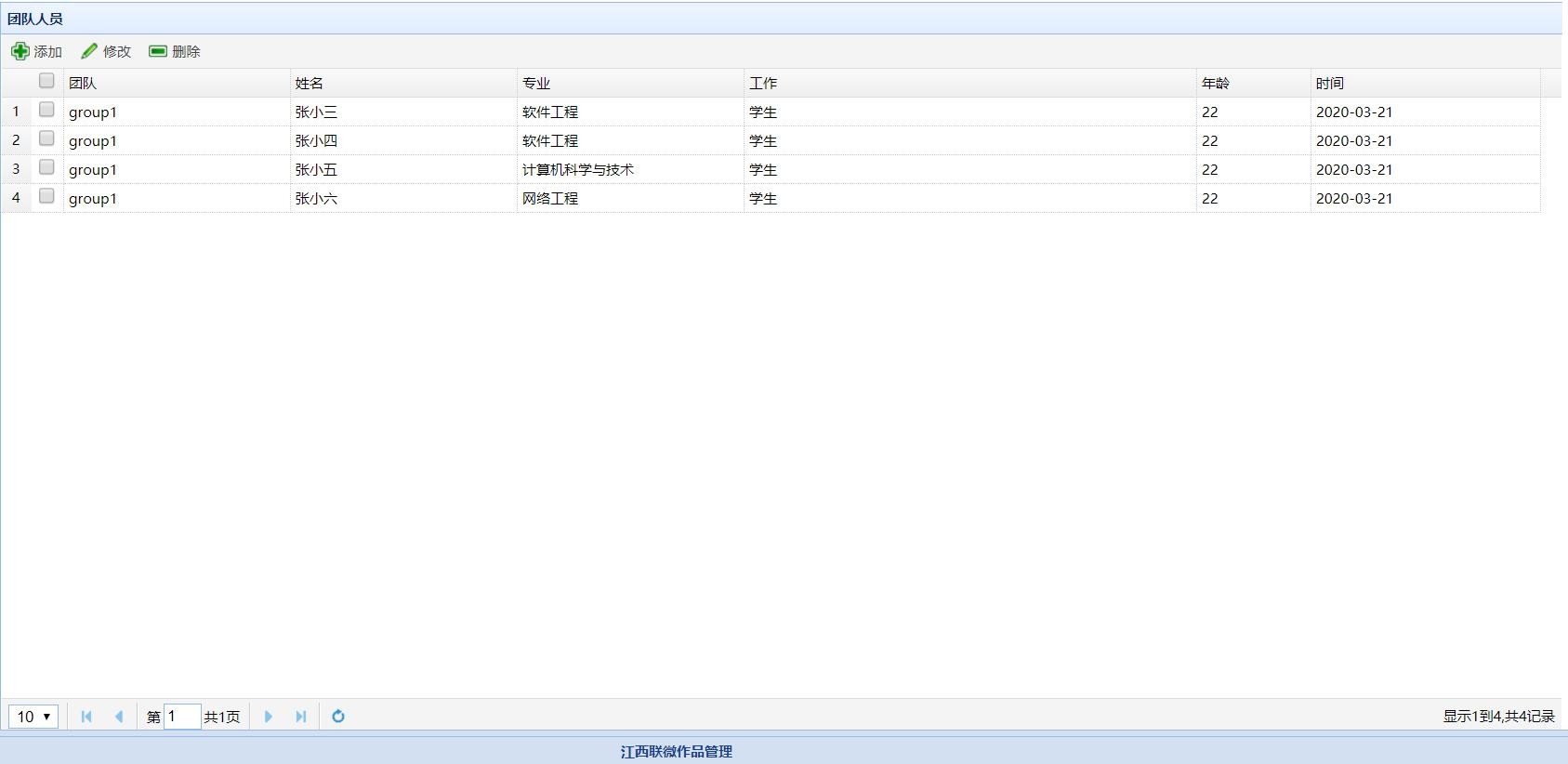
图 5-6 评分标准界面图

### 本系统登录即进入个人信息界面，在这里可以点击修改按钮进行修改个人信息操作， 如图 5-7 所示。

1. 团队信息界面

图 5-7 个人信息界面图

### 本系统登录即进入团队信息界面，可以点击添加按钮、修改按钮、删除按钮等进行相应的的操作，如图 5-8 所示。



1. 作品信息界面

图 5-8 团队信息界面图

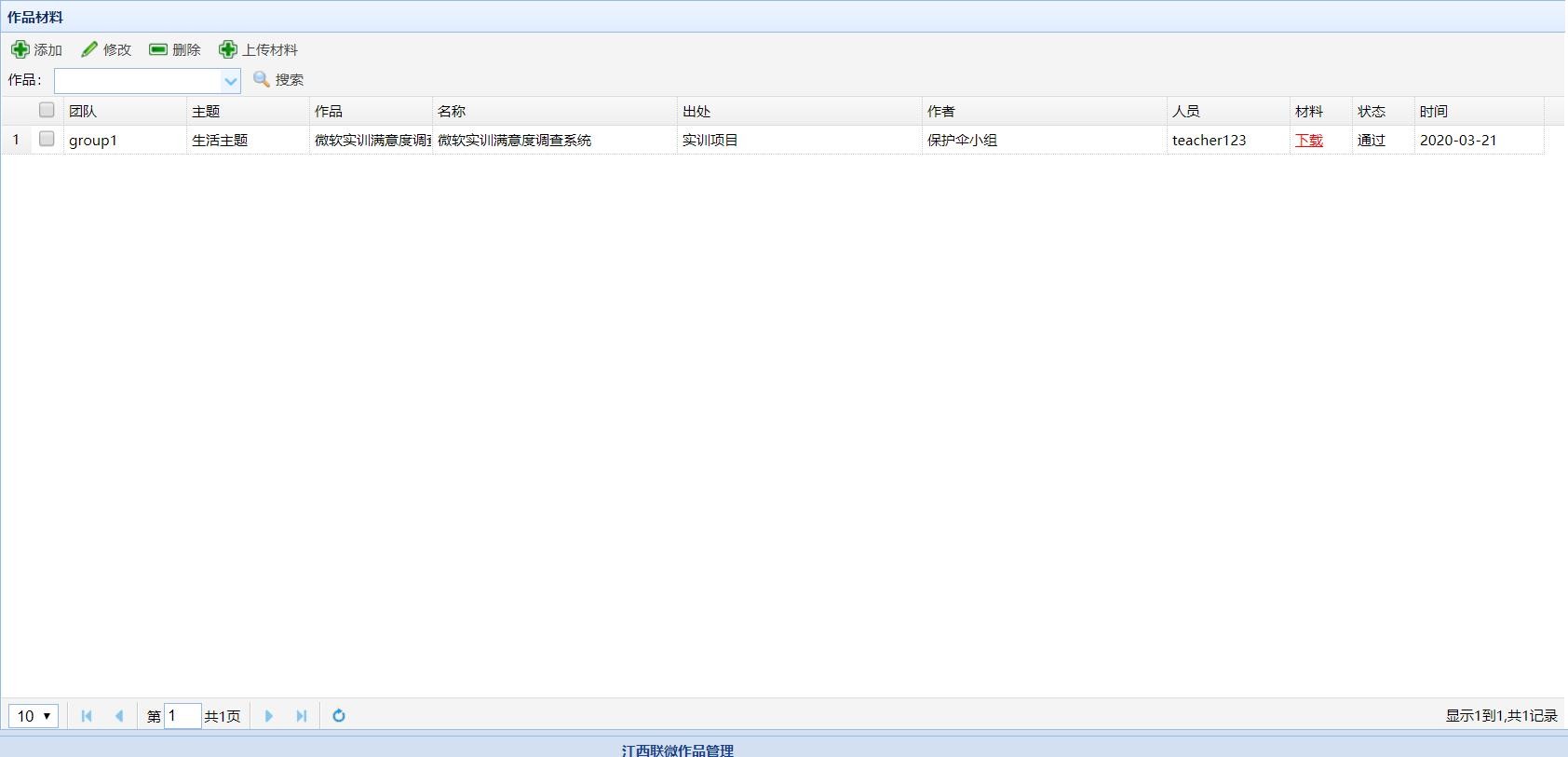
### 本系统登录即进入作品信息界面，在这里可以点击相应的按钮进行相应的的操作，如图 5-9 所示。

1. 作品材料界面

图 5-9 作品信息界面图

### 本系统登录即进入作品材料界面，可以点击相应的按钮进行相应的的操作，如图 5-10

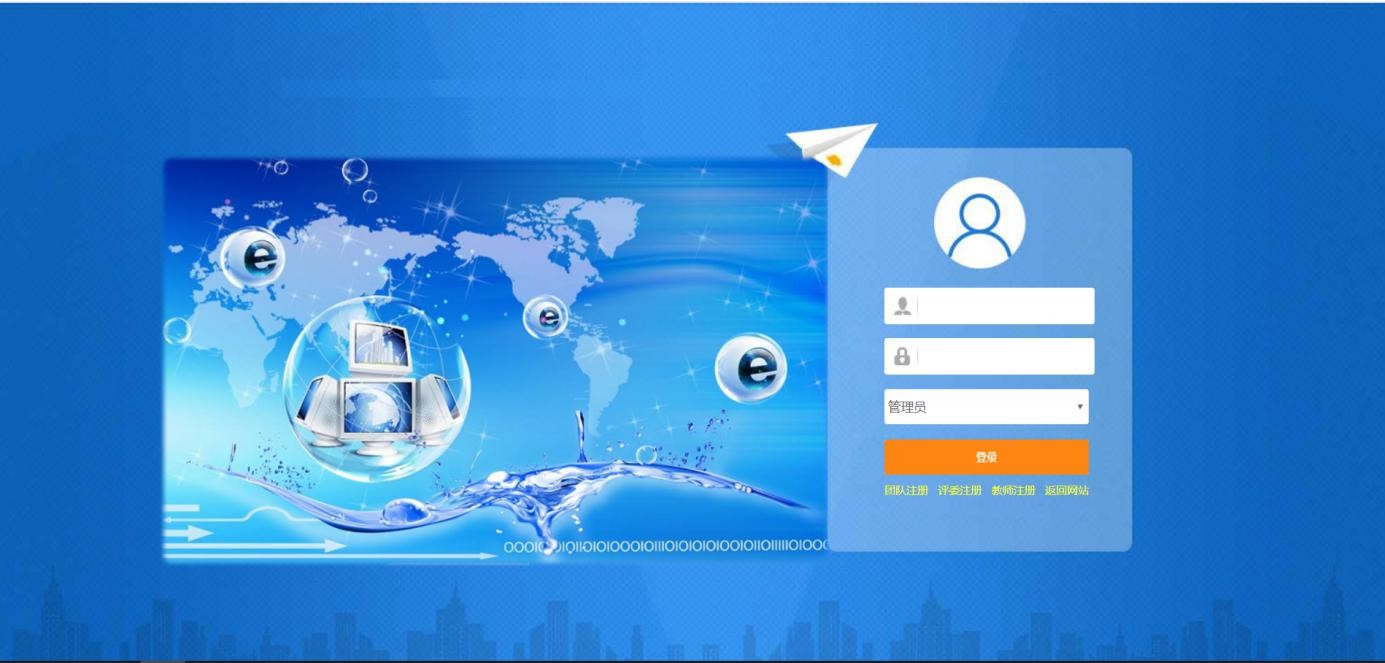
所示。



### 登录界面

图 5-10 作品材料界面图

### 本系统首页选择可进入登录界面，输入用户登录帐号和密码，选择登录角色后点击登录，如图 5-11 所示。



1. 注册界面

图 5-11 登录界面图

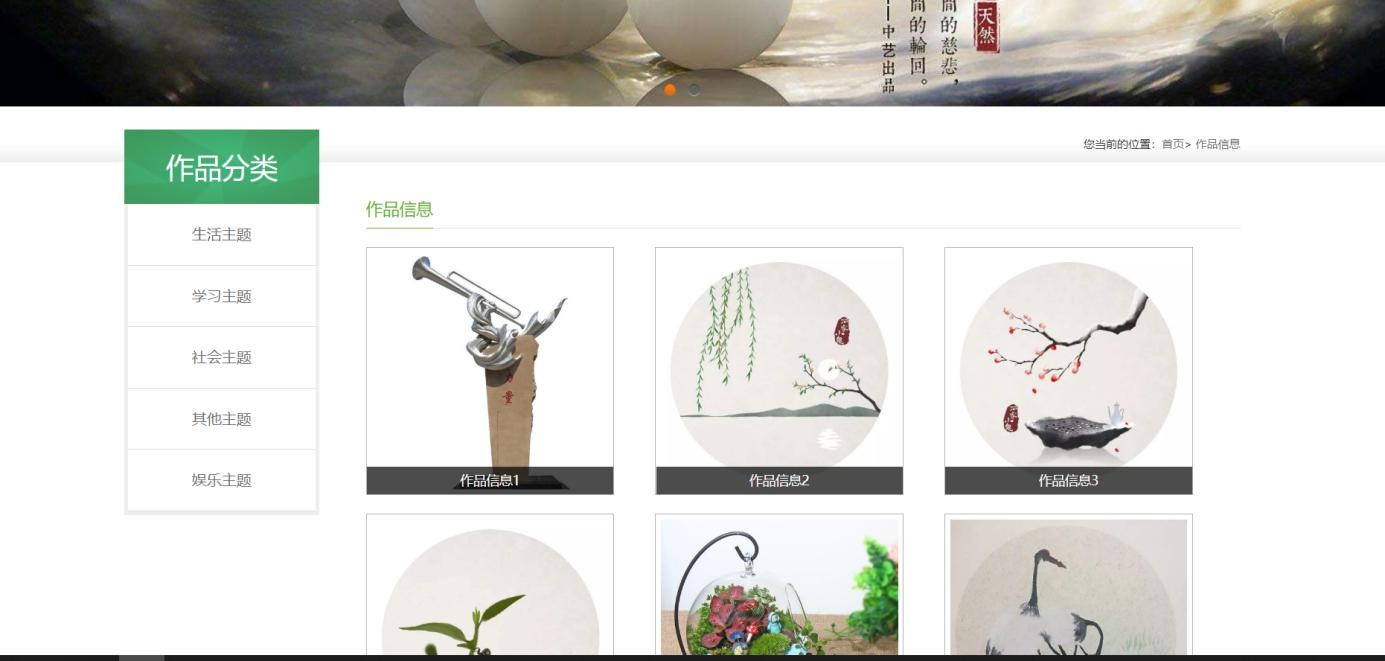
### 本系统登录界面可选择进入注册界面，输入基本信息即可注册，然后再跳转到登陆界面，如图 5-12 所示。



1. 分类展示界面

图 5-12 注册界面图

### 本系统登录即进入分类展示界面，如图 5-13 所示。



1. 详情界面

图 5-13 分类展示界面图

### 本系统通过展示界面可选择进入详情界面，可以点击点赞按钮和下载按钮，如图 5-14

所示。



图 5-14 详情界面图

* 1. **系统主要功能模块介绍**

# 详细设计

### 作品展示平台一共分为八个模块：个人信息管理模块、评分标准管理模块、作品主题管理模块、作品评分管理模块、作品信息管理模块、作品材料管理模块、点赞与下载模块、团队信息管理模块。其中个人信息管理模块包含个人信息修改以及密码修改， 评分标准管理模块分为评分标准修改、评分标准增加、评分标准删除，作品主题管理模块分为作品主题修改、作品主题增加、作品主题删除、更改主题状态，作品评分管理模块分为查看作品评分、选择标准评分，作品信息管理模块分为作品信息修改、作品信息增加、作品信息删除、上传缩略图、上传作品，作品材料管理模块分为材料修改、作品材料增加、作品材料删除、上传缩略图、上传材料，点赞与下载模块分为作品点赞、作品下载。系统的模块结构图如图 6-1 所示。

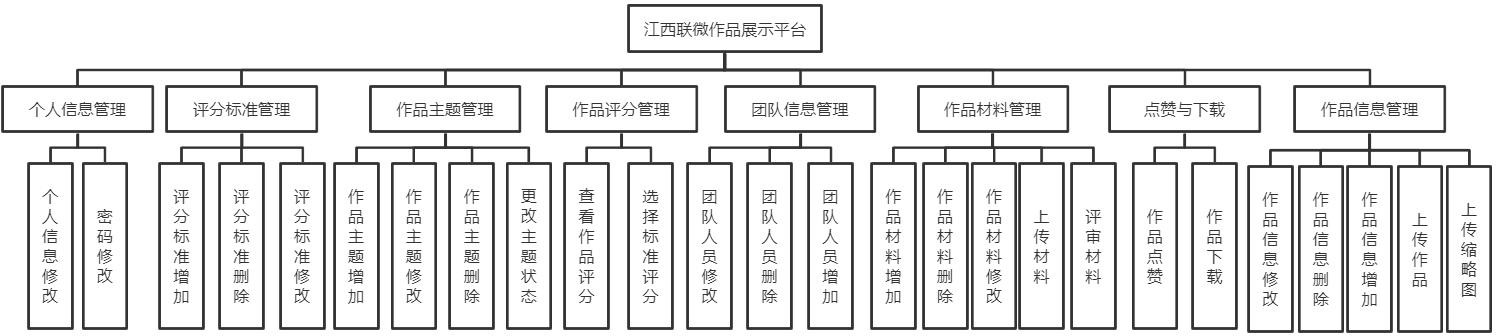


图 6-1 系统模块结构图

* 1. **个人信息管理模块设计**

### 功能结构设计

个人信息管理包图如图 6-2 所示。

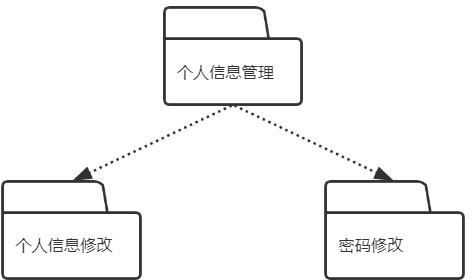


图 6-2 个人信息管理包图

### 个人信息管理功能包括个人信息修改以及密码修改，其中个人信息修改为所有用户修改姓名、年龄、性别、电话号码等信息，修改密码为所有用户修改个人登录密码。

### 类图设计

### 个人信息管理类图如图 6-3 所示。

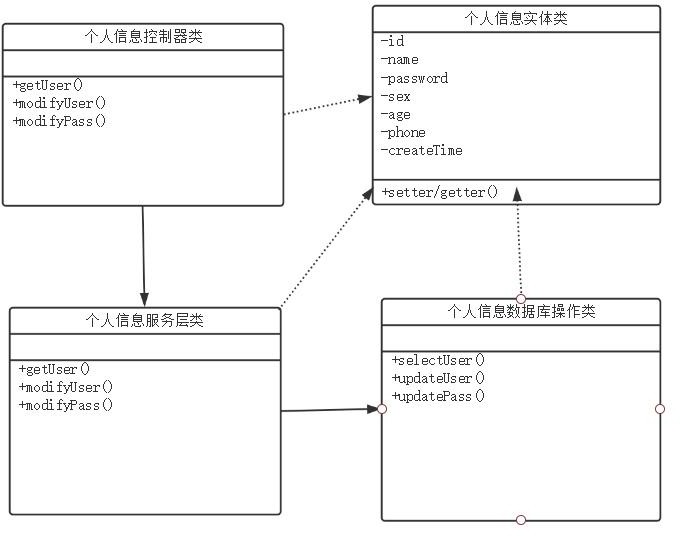


图 6-3 个人信息管理类图

### 在个人信息管理类图中，主要涉及到四个类的关联，个人信息控制器类关联个人信息服务层类，同时关联个人信息实体类。个人信息服务层类关联个人信息数据库操作类，同时关联个人信息实体类。个人信息管理涉及到的各个类的描述如下所示，其中个人信息控制器类描述如表 6-1 所示。

表 6-1 个人信息控制器类描述

|  |  |  |
| --- | --- | --- |
| 个人信息控制器类 | | |
| 类的描述：个人信息控制器类 | | |
| 操作描述 | GetUser () | 获取个人信息 |
| ModifyUser() | 修改个人信息 |
| ModifyPass () | 修改密码 |

### 个人信息服务层类描述如表 6-2 所示。

表 6-2 个人信息服务层类描述

|  |  |  |
| --- | --- | --- |
| 个人信息服务层类 | | |
| 类的描述：个人信息服务层类 | | |
| 操作描述 | GetUser () | 获取个人信息 |
| ModifyUser() | 修改个人信息 |
| ModifyPass () | 修改密码 |

### 个人信息数据库操作类描述如表 6-3 所示。

表 6-3 个人信息数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 个人信息数据库操作类 | | |
| 类的描述：个人信息数据库操作类 | | |
| 操作描述 | GetUser () | 获取个人信息 |
| ModifyUser() | 修改个人信息 |
| ModifyPass () | 修改密码 |

### 顺序图设计

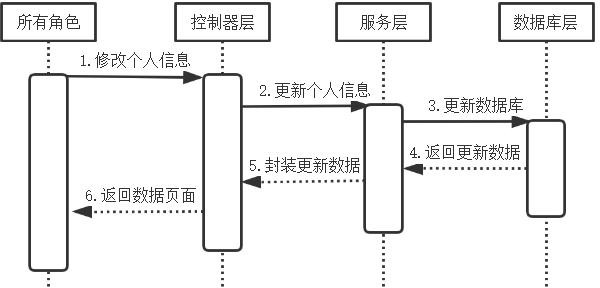
个人信息管理顺序图如图 6-4 所示。

图 6-4 个人信息管理顺序图

### 在个人信息顺序图中，所有角色点击修改个人信息，此时前台请求通过控制器层进行处理，然后修改的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

* + 1. 核心处理流程设计

### 个人信息管理活动图如图 6-5 所示。

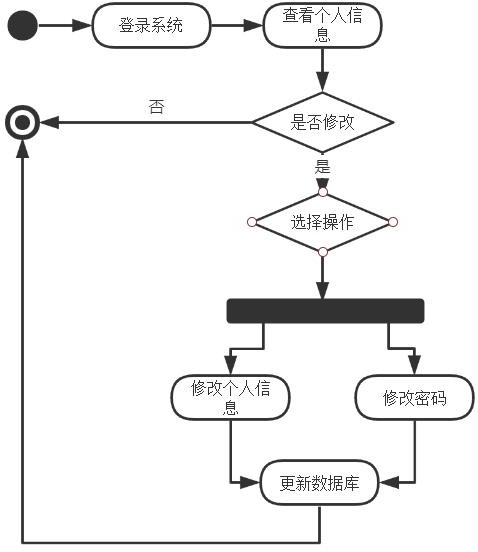


图 6-5 个人信息管理活动图

在个人信息管理活动图中，所有人员可以在登录系统后查看个人信息、修改个人信息、修改密码。

## 评分标准管理模块设计

### 功能结构设计

评分标准管理包图如图 6-6 所示。

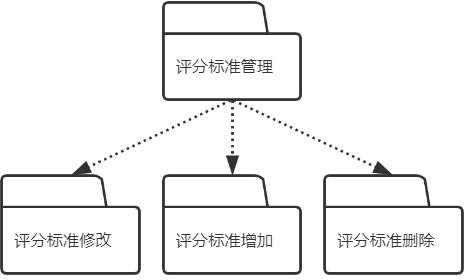


图 6-6 评分标准管理包图

### 评分标准管理功能包括评分标准修改、评分标准增加、评分标准删除，评分标准包括名称、说明、备注、要求等内容，管理员可以在登录系统后查看评分标准、修改评分标准、

删除评分标准、新增评分标准。

### 类图设计

评分标准管理类图如图 6-7 所示。

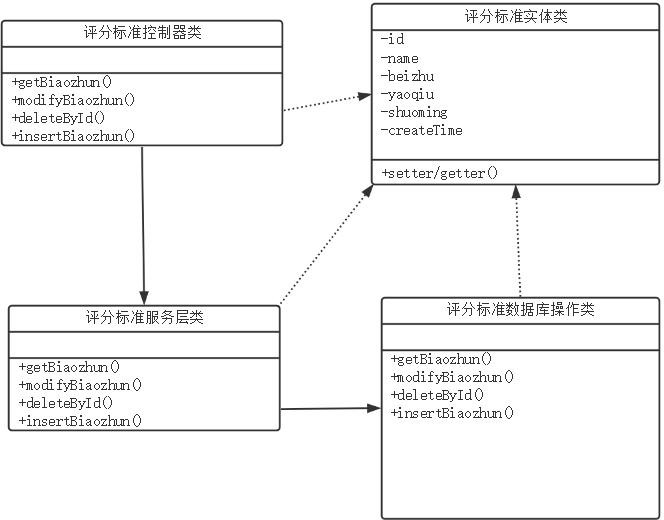


图 6-7 评分标准管理类图

### 在评分标准管理类图中，主要涉及到四个类的关联，评分标准控制器类关联评分标准服务层类，同时关联评分标准实体类。评分标准服务层类关联评分标准数据库操作类，同时关联评分标准实体类。评分标准管理涉及到的各个类的描述如下所示，其中评分标准控制器类描述如表 6-4 所示。

表 6-4 评分标准控制器类描述

|  |  |  |
| --- | --- | --- |
| 评分标准控制器类 | | |
| 类的描述：评分标准控制器类 | | |
| 操作描述 | GetBiaozhun () | 获取评分标准 |
| ModifyBiaozhun() | 修改评分标准 |
| DeleteById () | 删除评分标准 |
| InsertBiaozhun() | 新增评分标准 |

### 评分标准服务层类描述如表 6-5 所示。

表 6-5 评分标准服务层类描述

|  |  |  |
| --- | --- | --- |
| 评分标准服务层类 | | |
| 类的描述：评分标准服务层类 | | |
| 操作描述 | GetBiaozhun () | 获取评分标准 |
| ModifyBiaozhun() | 修改评分标准 |
| DeleteById () | 删除评分标准 |
| InsertBiaozhun() | 新增评分标准 |

### 评分标准数据库操作类描述如表 6-6 所示。

表 6-6 评分标准数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 评分标准数据库操作类 | | |
| 类的描述：评分标准数据库操作类 | | |
| 操作描述 | GetBiaozhun () | 获取评分标准 |
| ModifyBiaozhun() | 修改评分标准 |
| DeleteById () | 删除评分标准 |
| InsertBiaozhun() | 新增评分标准 |

### 顺序图设计

评分标准管理顺序图如图 6-8 所示。

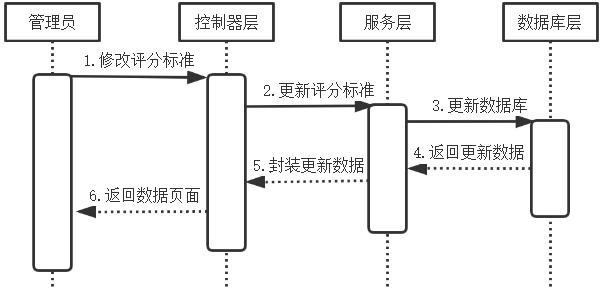


图 6-8 评分标准管理顺序图

### 在评分标准管理顺序图中，管理员点击修改评分标准，此时前台请求通过控制器层进行处理，然后修改的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

### 核心处理流程设计

### 评分标准管理活动图如图 6-9 所示。

图 6-9 评分标准管理活动图

在评分标准管理修改活动图中，管理员可以在登录系统后查看评分标准、修改评分标准、删除评分标准、新增评分标准。

## 作品主题管理模块设计

### 功能结构设计

作品主题管理包图如图 6-10 所示。

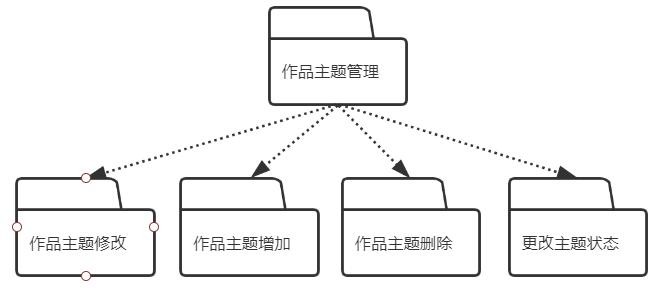


图 6-10 作品主题管理包图

### 作品主题管理功能包括作品主题修改、作品主题增加、作品主题删除、更改主题状态， 作品主题包括名称、说明、备注、要求等内容，管理员可以在登录系统后查看作品主题、

修改作品主题、删除作品主题、新增作品主题、更改主题状态。

### 类图设计

作品主题管理类图如图 6-11 所示。

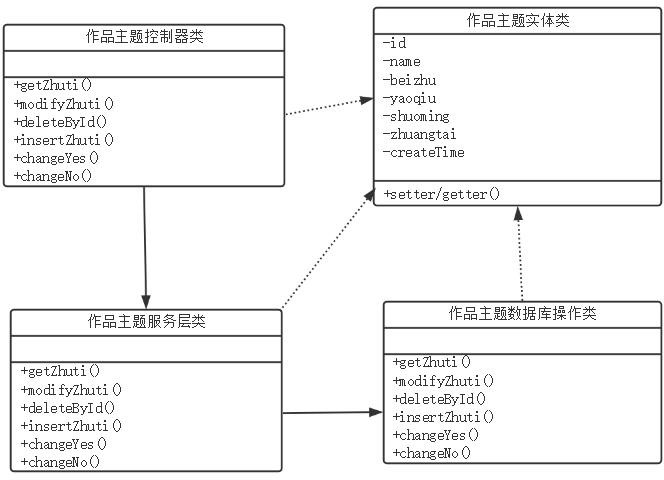


图 6-11 作品主题管理类图

### 在作品主题管理类图中，主要涉及到四个类的关联，作品主题控制器类关联作品主题服务层类，同时关联作品主题实体类。作品主题服务层类关联作品主题数据库操作类，同时关联作品主题实体类。作品主题管理涉及到的各个类的描述如下所示，其中作品主题控制器类描述如表 6-7 所示。

表 6-7 作品主题控制器类描述

|  |  |  |
| --- | --- | --- |
| 作品主题控制器类 | | |
| 类的描述：作品主题控制器类 | | |
| 操作描述 | GetZhuti () | 获取作品主题 |
| ModifyZhuti() | 修改作品主题 |
| DeleteById () | 删除作品主题 |
| InsertZhuti() | 新增作品主题 |
| ChangeYes() | 更改主题状态为可用 |
| ChangeNo() | 更改主题状态为禁用 |

### 作品主题服务层类描述如表 6-8 所示。

表 6-8 作品主题服务层类描述

|  |  |  |
| --- | --- | --- |
| 作品主题服务层类 | | |
| 类的描述：作品主题服务层类 | | |
| 操作描述 | GetZhuti () | 获取作品主题 |
| ModifyZhuti() | 修改作品主题 |
| DeleteById () | 删除作品主题 |
| InsertZhuti() | 新增作品主题 |
| ChangeYes() | 更改主题状态为可用 |
| ChangeNo() | 更改主题状态为禁用 |

### 作品主题数据库操作类描述如表 6-9 所示。

表 6-9 作品主题数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 作品主题数据库操作类 | | |
| 类的描述：作品主题数据库操作类 | | |
| 操作描述 | GetZhuti () | 获取作品主题 |
| ModifyZhuti() | 修改作品主题 |
| DeleteById () | 删除作品主题 |
| InsertZhuti() | 新增作品主题 |
| ChangeYes() | 更改主题状态为可用 |
| ChangeNo() | 更改主题状态为禁用 |

### 顺序图设计

作品主题管理顺序图如图 6-12 所示。

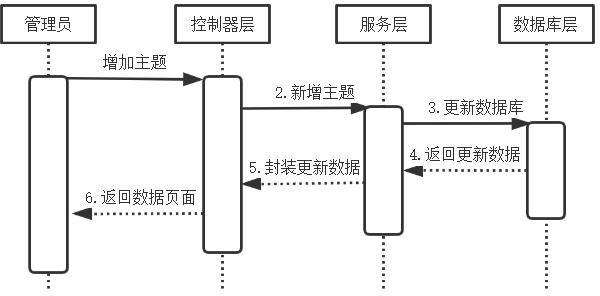


图 6-12 作品主题管理顺序图

### 在作品主题管理顺序图中，管理员点击增加作品主题，此时前台请求通过控制器层进行处理，然后修改的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

### 核心处理流程设计

### 作品主题管理活动图如图 6-13 所示。

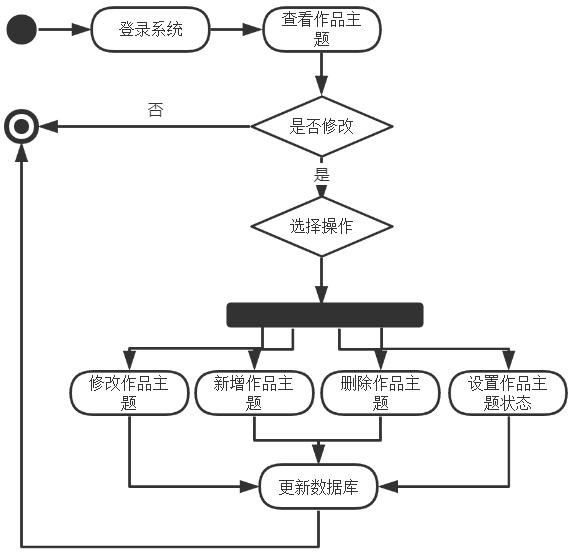


图 6-13 作品主题管理活动图

在作品主题管理活动图中，管理员可以在登录系统后查看作品主题、修改作品主题、删除作品主题、新增作品主题。

## 作品材料管理模块设计

### 功能结构设计

作品材料管理包图如图 6-14 所示。

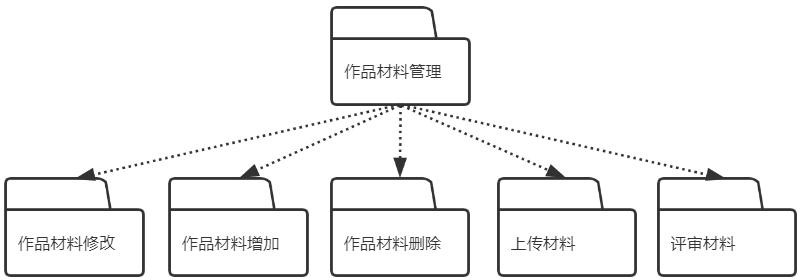


图 6-14 作品材料管理包图

### 作品材料管理功能包括作品材料修改、作品材料增加、作品材料删除、上传缩略图、上传材料，作品材料包括名称、出处、作者、所属作品、所属团队、材料文件、审核教师等内容，开发团队可以在登录系统后查看作品材料、修改作品材料、删除作品材料、新增作品材料、上传材料，教师可以在登录系统后查看作品材料、评审材料。

### 类图设计

### 作品材料管理类图如图 6-15 所示。

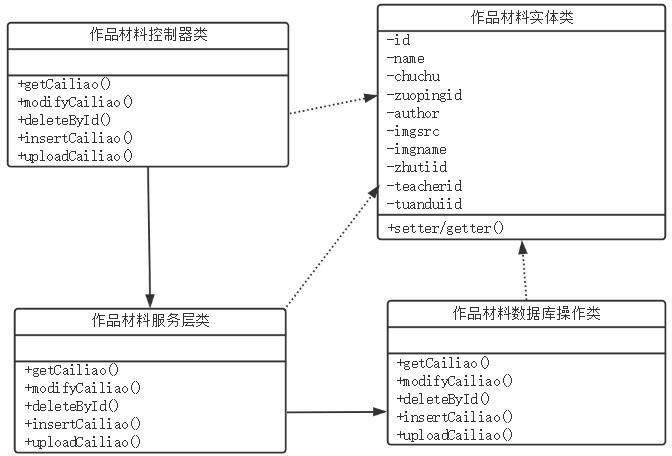


图 6-15 作品材料管理类图

### 在作品材料管理类图中，主要涉及到四个类的关联，作品材料控制器类关联作品材料服务层类，同时关联作品材料实体类。作品材料服务层类关联作品材料数据库操作类，同时关联作品材料实体类。作品材料管理涉及到的各个类的描述如下所示，其中作品材料控制器类描述如表 6-10 所示。

表 6-10 作品材料控制器类描述

|  |  |  |
| --- | --- | --- |
| 作品材料控制器类 | | |
| 类的描述：作品材料控制器类 | | |
| 操作描述 | GetCailiao () | 获取作品材料 |
| ModifyCailiao() | 修改作品材料 |
| DeleteById () | 删除作品材料 |
| InsertCailiao() | 新增作文材料 |
| UploadCailiao() | 上传作品材料 |

### 作品材料服务层类描述如表 6-11 所示。

表 6-11 作品材料服务层类描述

|  |  |  |
| --- | --- | --- |
| 作品材料服务层类 | | |
| 类的描述：作品材料服务层类 | | |
| 操作描述 | GetCailiao () | 获取作品材料 |
| ModifyCailiao() | 修改作品材料 |
| DeleteById () | 删除作品材料 |
| InsertCailiao() | 新增作文材料 |
| UploadCailiao() | 上传作品材料 |

### 作品材料数据库操作类描述如表 6-12 所示。

表 6-12 作品材料数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 作品材料数据库操作类 | | |
| 类的描述：作品材料数据库操作类 | | |
| 操作描述 | GetCailiao () | 获取作品材料 |
| ModifyCailiao() | 修改作品材料 |
| DeleteById () | 删除作品材料 |
| InsertCailiao() | 新增作文材料 |
|  | UploadCailiao() | 上传作品材料 |

### 顺序图设计

作品材料管理顺序图如图 6-16 所示。

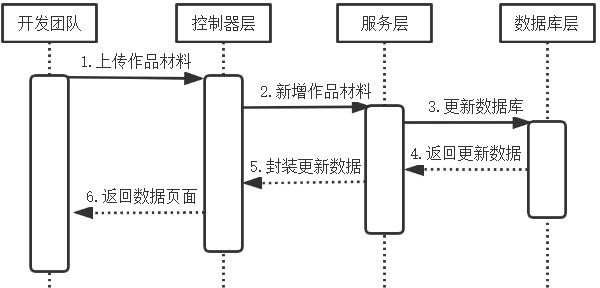


图 6-16 作品材料管理顺序图

### 在作品材料管理顺序图中，开发团队点击上传作品材料，此时前台请求通过控制器层进行处理，然后新增的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

* + 1. 核心处理流程设计

### 作品材料管理活动图如图 6-17 所示。

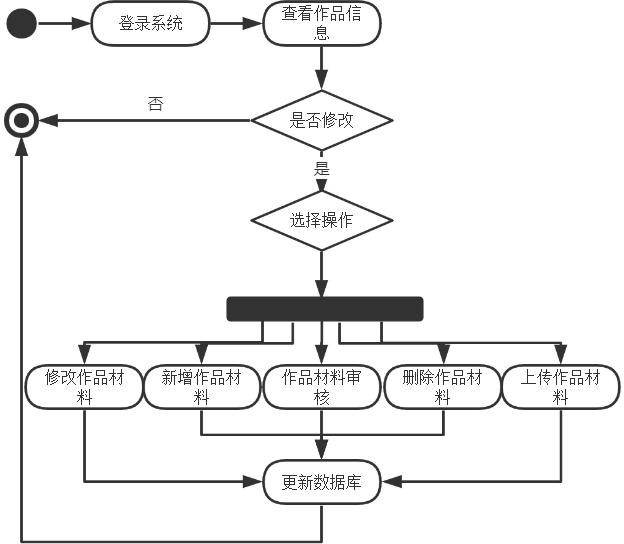


图 6-17 作品材料管理活动图

在作品材料管理活动图中，开发团队可以在登录系统后查看作品材料、修改作品材料、删除作品材料、新增作品材料、上传作品材料，教师可以审核作品材料。

## 作品信息管理模块设计

### 功能结构设计

作品信息管理包图如图 6-18 所示。

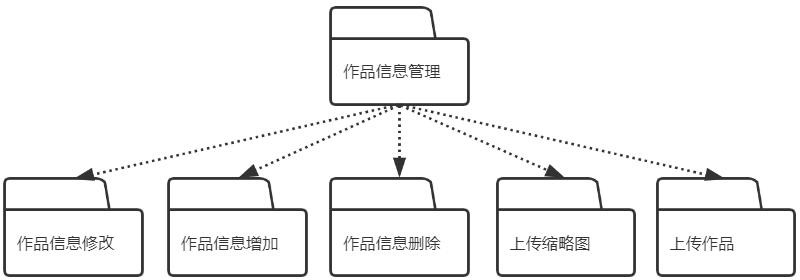


图 6-18 作品信息管理包图

### 作品信息管理功能包括作品信息修改、作品信息增加、作品信息删除、上传缩略图、上传作品，作品信息包括名称、文件、简介、缩略图、备注、点赞数、说明、作品主题、团队、下载数、总分等内容，开发团队可以在登录系统后查看作品信息、修改作品信息、删除作品信息、新增作品信息、上传缩略图、上传作品。

### 类图设计

### 作品信息管理类图如图 6-19 所示。

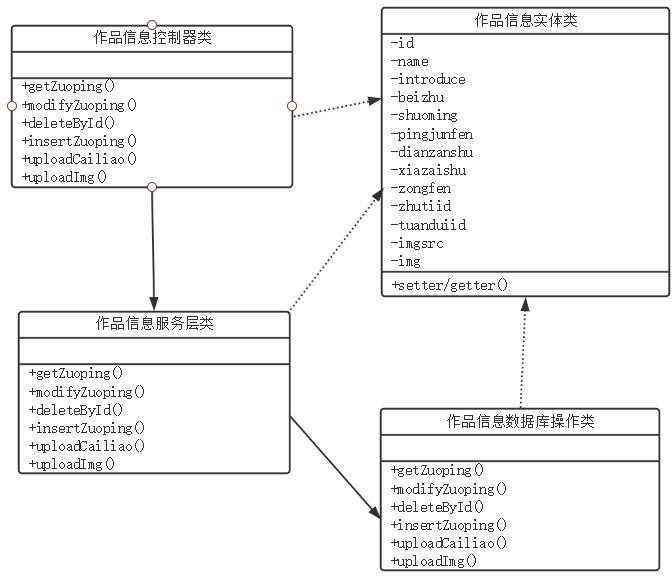


图 6-19 作品信息管理类图

### 在作品信息管理类图中，主要涉及到四个类的关联，作品信息控制器类关联作品信息服务层类，同时关联作品信息实体类。作品信息服务层类关联作品信息数据库操作类，同时关联作品信息实体类。作品信息管理涉及到的各个类的描述如下所示，其中作品信息控制器类描述如表 6-13 所示。

表 6-13 作品信息控制器类描述

|  |  |  |
| --- | --- | --- |
| 作品信息控制器类 | | |
| 类的描述：作品信息控制器类 | | |
| 操作描述 | GetZuoping () | 获取作品信息 |
| ModifyZuoping() | 修改作品信息 |
| InsertZuoping () | 新增作品信息 |
| UploadCailiao() | 上传作品 |
| UploadImg() | 上传缩略图 |

### 作品信息服务层类描述如表 6-14 所示。

表 6-14 作品信息服务层类描述

|  |  |  |
| --- | --- | --- |
| 作品信息服务层类 | | |
| 类的描述：作品信息服务层类 | | |
| 操作描述 | GetZuoping () | 获取作品信息 |
| ModifyZuoping() | 修改作品信息 |
| InsertZuoping () | 新增作品信息 |
| UploadCailiao() | 上传作品 |
| UploadImg() | 上传缩略图 |

### 作品信息数据库操作类描述如表 6-15 所示。

表 6-15 作品信息数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 作品信息数据库操作类 | | |
| 类的描述：作品信息数据库操作类 | | |
| 操作描述 | GetZuoping () | 获取作品信息 |
| ModifyZuoping() | 修改作品信息 |
| InsertZuoping () | 新增作品信息 |
| UploadCailiao() | 上传作品 |
| UploadImg() | 上传缩略图 |

### 顺序图设计

作品信息管理顺序图如图 6-20 所示。

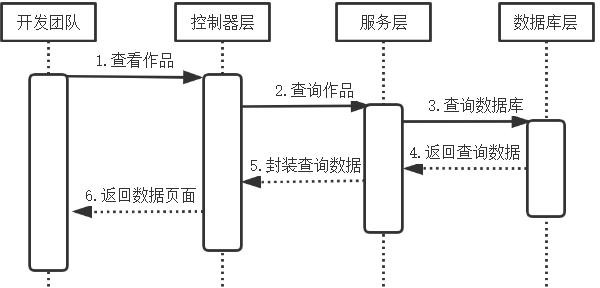


图 6-20 作品信息管理顺序图

### 在作品信息管理顺序图中，开发团队查看作品，此时前台请求通过控制器层进行处理，

然后查询的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

### 核心处理流程设计

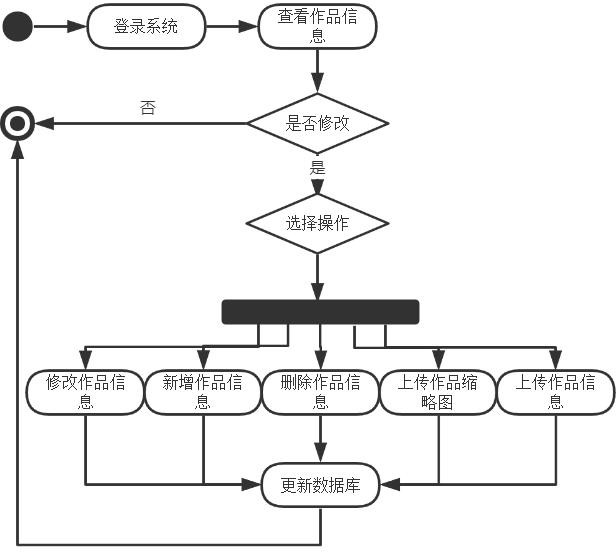
作品信息管理活动图如图 6-21 所示。

图 6-21 作品信息管理活动图

在作品信息管理活动图中，开发团队可以在登录系统后查看作品信息、修改作品信息、删除作品信息、新增作品信息、上传作品、上传作品缩略图。

## 点赞和下载模块设计

### 功能结构设计

点赞和下载包图如图 6-22 所示。

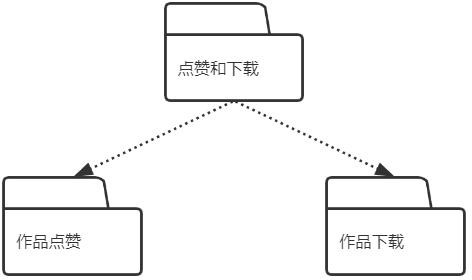


图 6-22 点赞和下载包图

### 点赞与下载功能包括作品点赞、作品下载，本系统所有用户都可以在登录系统后对作品进行点赞与作品下载。

* + 1. 类图设计

### 点赞与下载类图如图 6-23 所示。

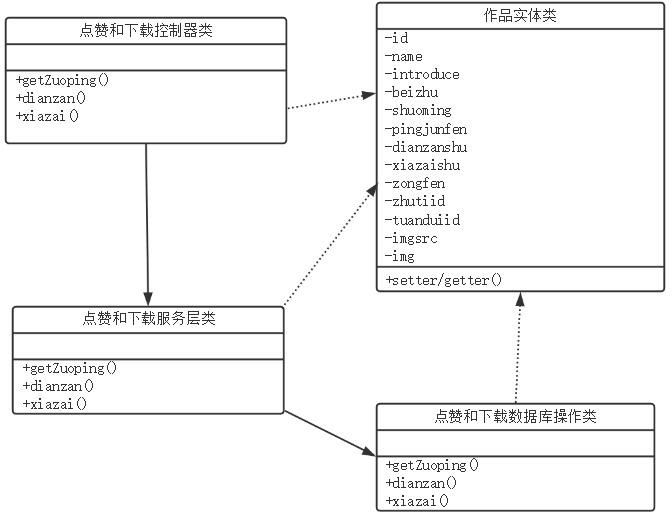


图 6-23 点赞和下载类图

### 在点赞与下载类图中，主要涉及到四个类的关联，点赞与下载控制器类关联点赞与下载服务层类，同时关联作品实体类。点赞与下载服务层类关联点赞与下载数据库操作类， 同时关联作品信息实体类。点赞与下载涉及到的各个类的描述如下所示，其中点赞与下载控制器类描述如表 6-16 所示。

表 6-16 点赞和下载控制器类描述

|  |  |  |
| --- | --- | --- |
| 点赞和下载控制器类 | | |
| 类的描述：点赞和下载控制器类 | | |
| 操作描述 | GetZuoping() | 获取作品 |
| Dianzan() | 点赞 |
| Xiazai () | 下载 |

### 点赞与下载服务层类描述如表 6-17 所示。

表 6-17 点赞和下载服务层类描述

|  |  |  |
| --- | --- | --- |
| 点赞和下载服务层类 | | |
| 类的描述：点赞和下载服务层类 | | |
| 操作描述 | GetZuoping() | 获取作品 |
| Dianzan() | 点赞 |
| Xiazai () | 下载 |

### 点赞与下载数据库操作类描述如表 6-18 所示。

表 6-18 点赞和下载数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 点赞和下载数据库操作类 | | |
| 类的描述：点赞和下载数据库操作类 | | |
| 操作描述 | GetZuoping() | 获取作品 |
| Dianzan() | 点赞 |
| Xiazai () | 下载 |

### 顺序图设计

点赞和下载顺序图如图 6-24 所示。

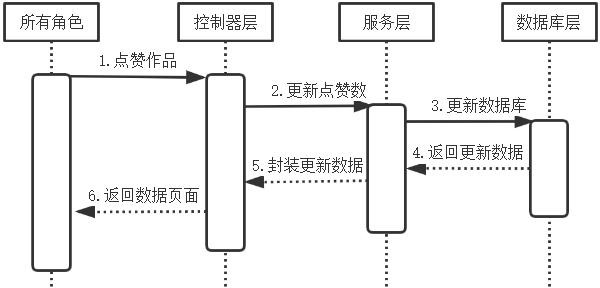


图 6-24 点赞和下载顺序图

### 在点赞和下载顺序图中，所有角色可以点赞作品，此时前台请求通过控制器层进行处理，然后更新的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

* + 1. 核心处理流程设计

### 点赞和下载活动图如图 6-25 所示。

图 6-25 点赞和下载活动图

在点赞和下载活动图中，所有人员可以在登录系统后查看首页、直接浏览作品信息、分类浏览作品信息、进入详情页点赞作品、进入详情页下载作品。

## 作品评分管理模块设计

### 功能结构设计

作品评分管理包图如图 6-26 所示。

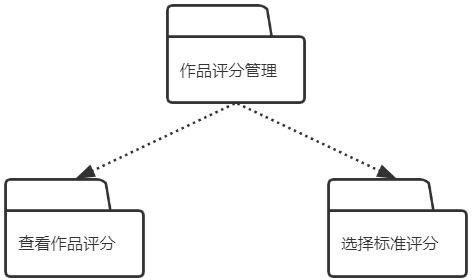


图 6-26 作品评分管理包图

### 作品评分管理功能包括查看作品评分、选择标准评分，评委可以在登录系统后查看作品信息及评分、选择评分标准对作品评分。

* + 1. 类图设计

### 作品评分管理类图如图 6-27 所示。

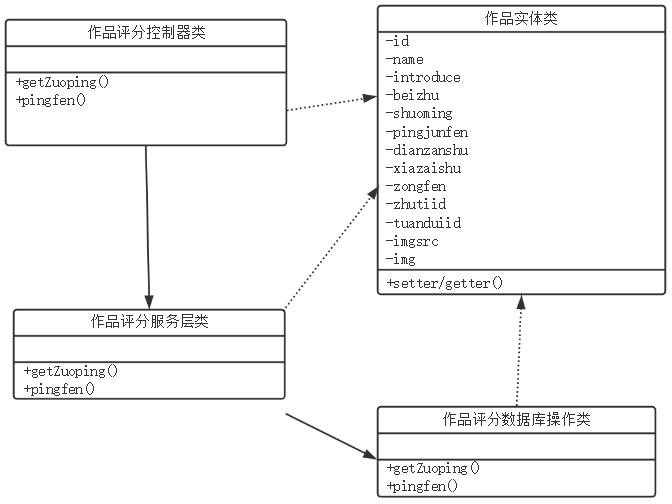


图 6-27 作品评分管理类图

### 在作品评分管理类图中，主要涉及到四个类的关联，作品评分控制器类关联作品评分服务层类，同时关联作品评分实体类。作品评分服务层类关联作品评分数据库操作类，同时关联作品评分实体类。作品评分管理涉及到的各个类的描述如下所示，其中作品评分控制器类描述如表 6-19 所示。

表 6-19 作品评分控制器类描述

|  |  |  |
| --- | --- | --- |
| 作品评分控制器类 | | |
| 类的描述：作品评分控制器类 | | |
| 操作描  述 | GetZuoping () | 获取作品 |
| Pingfen() | 评分 |

### 作品评分服务层类描述如表 6-20 所示。

表 6-20 作品评分服务层类描述

|  |  |  |
| --- | --- | --- |
| 作品评分服务层类 | | |
| 类的描述：作品评分服务层类 | | |
| 操作描  述 | GetZuoping () | 获取作品 |
| Pingfen() | 评分 |

### 作品评分数据库操作类描述如表 6-21 所示。

表 6-21 作品评分数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 作品评分数据库操作类 | | |
| 类的描述：作品评分数据库操作类 | | |
| 操作描  述 | GetZuoping () | 获取作品 |
| Pingfen() | 评分 |

### 顺序图设计

作品评分管理顺序图如图 6-28 所示。

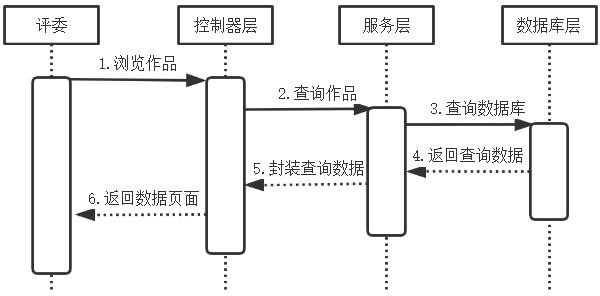


图 6-28 作品评分管理顺序图

### 在作品评分管理顺序图中，评委浏览作品，此时前台请求通过控制器层进行处理，然后查询的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

* + 1. 核心处理流程设计

### 作品评分活动图如图 6-29 所示。

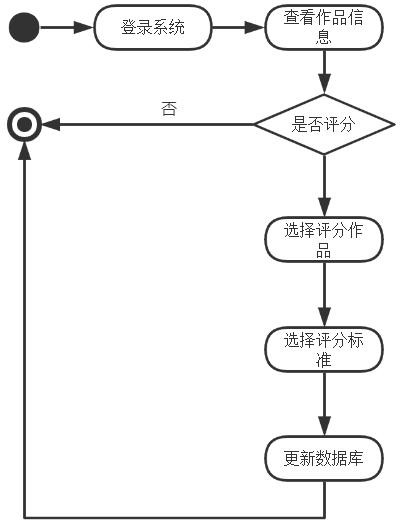


图 6-29 作品评分活动图

在作品评分活动图中，评委可以在登录系统后查看作品信息、选择评分标准为作品进行评分。

## 团队信息管理模块设计

### 功能结构设计

团队信息管理包图如图 6-30 所示。

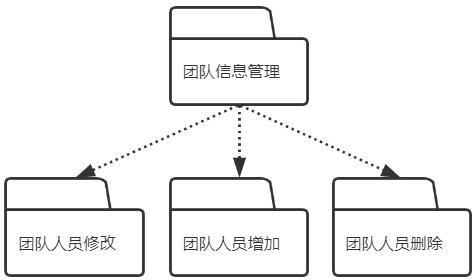


图 6-30 团队信息管理包图

### 团队信息管理功能包括团队人员修改、团队人员增加、团队人员删除，团队人员包括姓名、专业、工作、年龄等内容，开发团队可以在登录系统后查看团队人员、修改团队人员、删除团队人员、新增团队人员。

* + 1. 类图设计

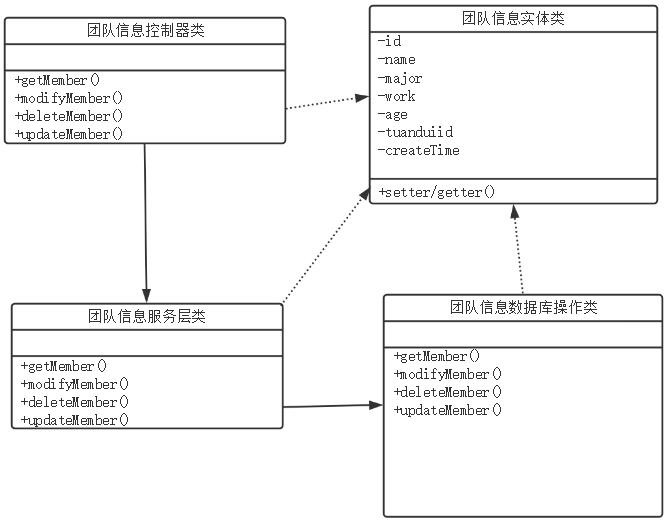


图 6-31 团队信息管理类图

### 在团队信息管理类图中，主要涉及到四个类的关联，团队信息控制器类关联团队信息服务层类，同时关联团队信息实体类。团队信息服务层类关联团队信息数据库操作类，同时关联团队信息实体类。团队信息管理涉及到的各个类的描述如下所示，其中团队信息控制器类描述如表 6-22 所示。

表 6-22 团队信息控制器类描述

|  |  |  |
| --- | --- | --- |
| 团队信息控制器类 | | |
| 类的描述：团队信息控制器类 | | |
| 操作描述 | GetMember () | 获取团队人员 |
| ModifyMember() | 修改团队人员 |
| DeleteMember() | 删除团队人员 |
| UpdateMember() | 新增团队人员 |

### 团队信息服务层类描述如表 6-23 所示。

表 6-23 团队信息服务层类描述

|  |  |  |
| --- | --- | --- |
| 团队信息服务层类 | | |
| 类的描述：团队信息服务层类 | | |
| 操作描述 | GetMember () | 获取团队人员 |
| ModifyMember() | 修改团队人员 |
| DeleteMember() | 删除团队人员 |
| UpdateMember() | 新增团队人员 |

### 团队信息数据库操作类描述如表 6-24 所示。

表 6-24 团队信息数据库操作类描述

|  |  |  |
| --- | --- | --- |
| 团队信息数据库操作类 | | |
| 类的描述：团队信息数据库操作类 | | |
| 操作描述 | GetMember () | 获取团队人员 |
| ModifyMember() | 修改团队人员 |
| DeleteMember() | 删除团队人员 |
| UpdateMember() | 新增团队人员 |

### 顺序图设计

团队信息管理顺序图如图 6-32 所示。

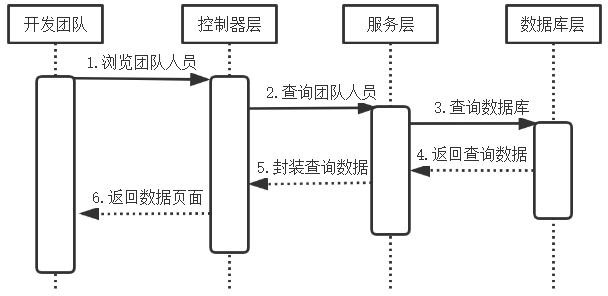


图 6-32 团队信息管理顺序图

### 在团队信息管理顺序图中，开发团队浏览团队人员，此时前台请求通过控制器层进行处理，然后查询的请求就可以进入服务层，服务层再调用数据库层，数据库层拿到数据库更新结果就可以返回更新后的数据，然后进行封装数据再由控制器层返回到前台的展示数据页面。

* + 1. 核心处理流程设计

### 团队信息管理活动图如图 6-33 所示。

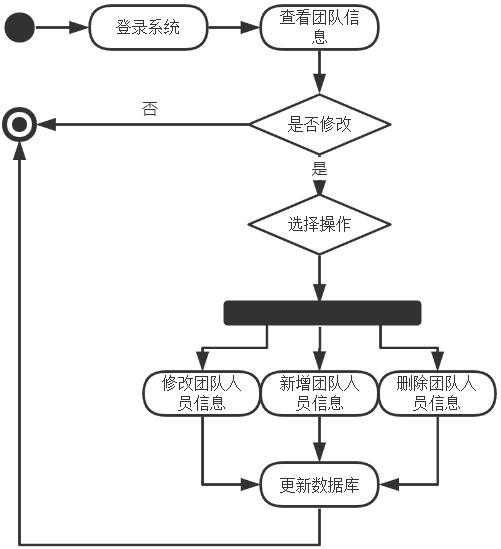


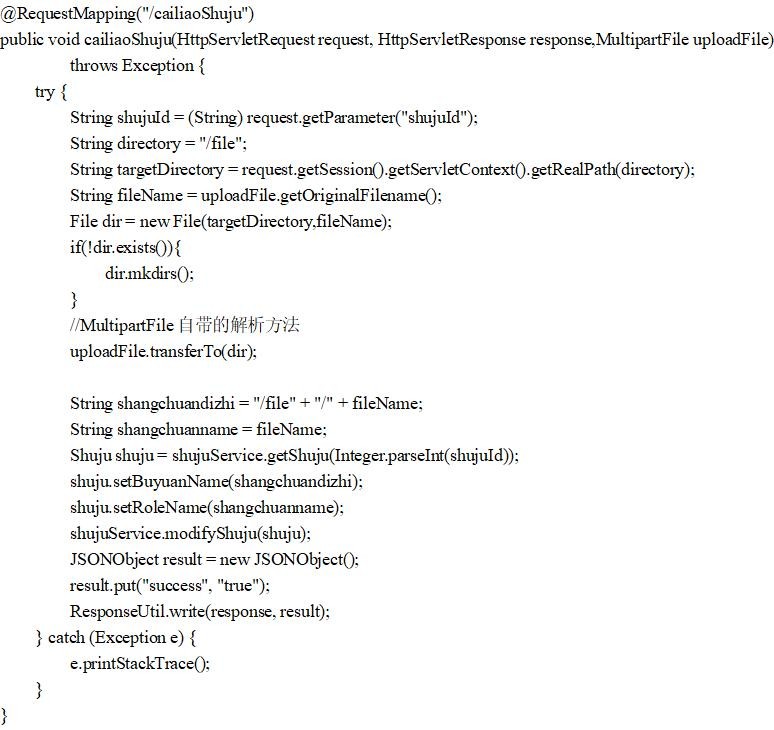
图 6-33 团队信息管理活动图

在团队信息管理活动图中，开发团队可以在登录系统后查看团队信息、修改团队人员信息、删除团队人员信息、新增团队人员信息。

* 1. **代码实现与核心算法**

# 编码

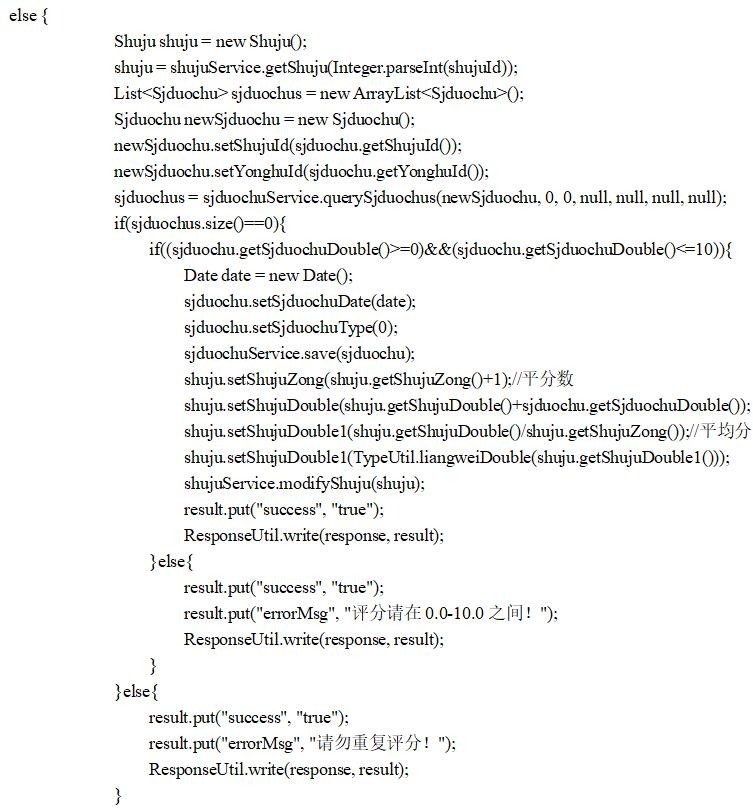
### 上传作品核心代码，本段代码主要是实现作品上传功能，前台发送请求给控制器层， 控制器层根据发送过来的信息做处理，首先先把文件放到指定位置，然后根据作品信息编号将文件地址和文件名填入，最后提示上传成功。



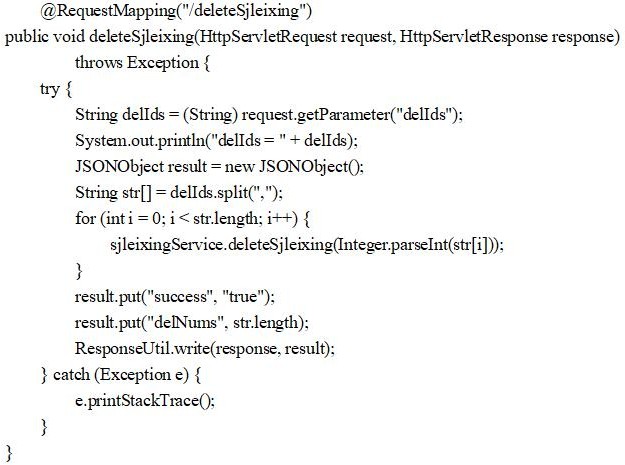
1. 不同角色登录进不同页面核心代码，本段代码主要是实现判断不同角色登录进入不同页面的功能。判断登录进来的是什么角色，系统就从数据库中查询该角色所对应的用户账号和密码是否正确，如正确则跳转到相应页面，如不正确则提示用户名或密码错误。



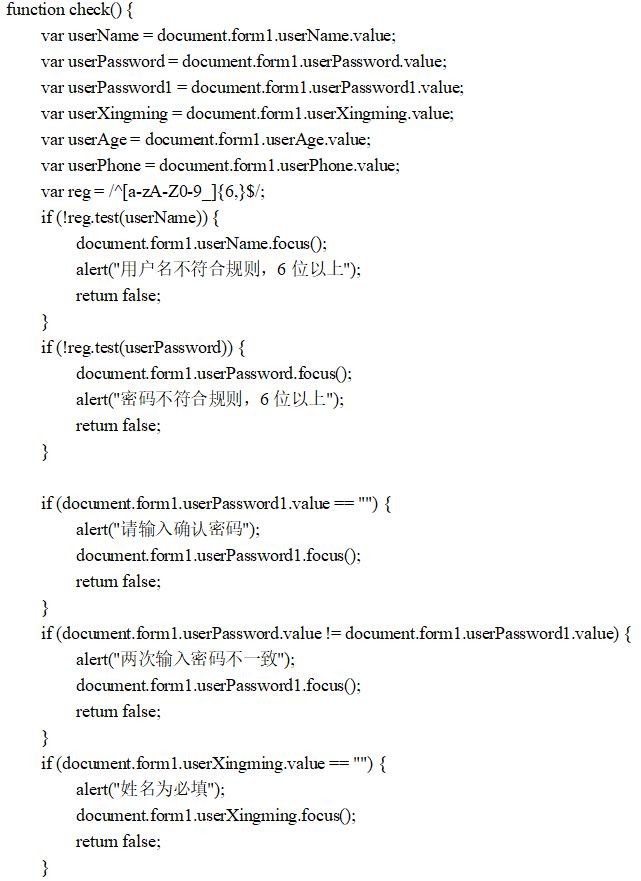
### 作品评分统计核心代码，本段代码主要是实现作品评分统计功能。首先通过前台传过来的请求进入控制器层，将评分总次数增加一次，将本次的评分加上之前的总分然后除以总次数即可得到平均分，然后再更新数据库。



1. 删除主题核心代码，本段代码主要是实现删除作品主题功能。首先前台发送请求给控制器层，控制器层接受请求删除指定的作品主题，最后提示删除成功。

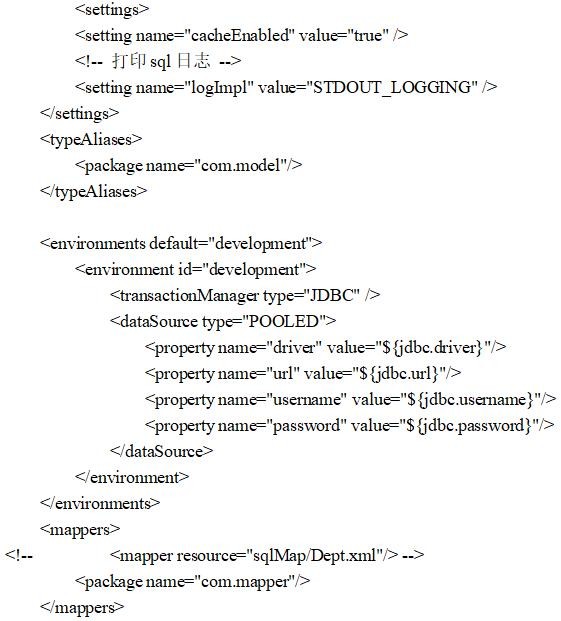


1. 团队注册前台核心代码，本段代码主要是前台实现团队注册功能。首先先在前台根据输入的数据进行检查，使用正则表达式来规范密码。通过比较两次输入的密码是否一致如果不一致则给出提示。判断其他信息输入是否为空，如果为空则给出提示。



* 1. **代码优化分析**

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



1. 前端代码复用。将前端界面划分成几个区域，有些区域内容几乎一样，可以把它们的共性抽取出来，在页面设计时引入即可，如果不进行代码复用可能会导致在一些页面间样式错乱，这样很不美观，极大影响用户的观感。



# 测试

## 测试方案设计

### 测试策略

在作品展示系统的开发这一漫长阶段中，会出现非常多的问题，所以在软件生命周期的每个阶段都不可避免会出现差错，开发人员需要尽可能多地去发现存在于系统当中地错误，而一旦某些错误如果不能在投入生产性运行之前发现，则这些差错一定会在生产性运行时发现，此时如果再来改正这些错误代价会更高，会产生更加恶劣的后果，所以测试的对于一个系统来说是非常重要的。

### 作品展示系统的测试方法为白盒测试和黑盒测试。测试的步骤为先进行单元测试，然后进行子系统测试，接着进入系统测试,再进行验收测试，最后是平行运行。

* + 1. 测试进度安排

### 作品展示系统的测试进度安排如下表 8-1 所示。

表 8-1 测试进度安排表

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

### 测试资源

作品展示系统作为一套 B/S 架构的项目，如果他在线上发布，则可能会有不同用户通过不同的浏览器进行访问，故测试资源中必须包含不同浏览器以测试在不同浏览器环境下是否能够正常运行，测试资源表(硬件环境)如表 8-2 所示。

表 8-2 测试资源表

|  |  |  |
| --- | --- | --- |
| 硬件环境 | | |
| 资源类型 | 资源描述 | 数量 |
| 内存 | 8G DDR3L | 1 |
| 硬盘 | 256G SATA3 接口 | 1 |
| 处理器 | Intel I5-4210M | 1 |

续表 8-2

|  |  |  |
| --- | --- | --- |
| 操作系统 | Windows 10 1903 企业版 | 1 |
| JDK 版本 | JDK 9.0.1 | 1 |
| 浏览器 1 | Chrome 浏览器 | 1 |
| 浏览器 2 | FireFox 浏览器 | 1 |
| 浏览器 3 | 360 安全浏览器 | 1 |
| 浏览器 4 | Microsoft Edge 浏览器 | 1 |
| 浏览器 5 | IE 浏览器 | 1 |

### 测试资源表(软件环境)如表 8-3 所示。

表 8-3 测试资源表(软件环境)

|  |  |
| --- | --- |
| 软件环境 | |
| 资源类型 | 软件 |
| 应用数据库 | MySQL 5.7 |
| 应用服务器 | Tomcat 8.5.16 |

### 关键测试点

1. 开发团队是否能够添加作品材料。

### 教师审核材料功能是否可行。

1. 评委能否对作品进行打分。

### 开发团队能否完成作品信息的删除，以及在进行删除操作时，系统是否会给出提示框进行提示。

1. 系统管理员能否对作品主题进行修改操作。

### 所有人员是否能够点赞作品、下载作品。

1. 系统管理员能否对评分标准进行修改操作。

## 测试用例构建

### 测试用例编写约定

1. 用例编号：测试用例的编号，具有唯一性。

### 用例名称：测试用例的命名。

1. 前提条件：该测试用例执行前需要满足的条件。

### 操作步骤：测试用例如何执行。

1. 预期结果：理论上测试用例应该得到的结果。

### 实际结果：实际上测试用例应该得到的结果。

1. 问题描述：测试用例执行后的错误表现。

### BUG 编号：出错的编号。

1. 测试结果与结论：选择通过或者不通过。

### 测试执行者：测试用例执行的人员。

* + 1. 测试用例设计

### 添加作品材料测试用例表如表 8-4 所示。

表 8-4 添加作品材料测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL01 |
| 用例名称 | 添加作品材料 |
| 前提条件 | 开发团队成功登录系统 |
| 操作步骤 | 点击上传材料按钮，再选择相应的材料，然后点击确定。 |
| 预期结果 | 系统显示上传成功并刷新页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 审核材料测试用例表如表 8-5 所示。

表 8-5 审核材料测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL02 |
| 用例名称 | 审核材料 |
| 前提条件 | 教师成功登录系统 |
| 操作步骤 | 1. 教师首先下载材料然后审核，根据材料质量决定将材料通过。 2. 点击通过。 |
| 预期结果 | 界面显示材料为通过状态 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 评委打分测试用例表如表 8-6 所示。

表 8-6 评委打分测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL03 |
| 用例名称 | 评委打分 |
| 前提条件 | 评委成功登录系统 |
| 操作步骤 | 1. 评委先查看作品。 2. 选择需要打分的项目，选择评分标准，打分。 |
| 预期结果 | 系统显示打分成功，然后刷新页面。 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 作品删除测试用例表如表 8-7 所示。

表 8-7 作品删除测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL04 |
| 用例名称 | 作品删除 |
| 前提条件 | 开发团队成功登录系统 |
| 操作步骤 | 开发团队选择一个作品删除，系统提示是否删除，点击确定。 |
| 预期结果 | 系统显示删除成功并刷新页面。 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 关键测试用例

* + - 1. 修改作品主题测试用例表如表 8-8 所示。

表 8-8 修改作品主题测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL05 |
| 用例名称 | 修改作品主题 |
| 前提条件 | 管理员成功登录系统 |
| 操作步骤 | 1. 管理员选择一个主题，点击修改按钮。 2. 修改主题相关信息后点击确定。 |
| 预期结果 | 系统提示修改成功并刷新页面。 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 点赞作品测试用例表如表 8-9 所示。

表 8-9 点赞作品测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL06 |
| 用例名称 | 点赞作品 |
| 前提条件 | 所有人员成功登录系统，并进入详情页 |
| 操作步骤 | 浏览后点击点赞按钮 |
| 预期结果 | 系统提示点赞成功并且可看到点赞数增加 1 个 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 修改评分标准测试用例表如表 8-10 所示。

表 8-10 添加作品材料测试用例表

|  |  |
| --- | --- |
| 用例编号 | CSYL07 |
| 用例名称 | 修改评分标准 |
| 前提条件 | 管理员成功登录系统 |

续表 8-10

|  |  |
| --- | --- |
| 操作步骤 | 1. 管理员选择一个评分标准，点击修改按钮。 2. 修改评分标准相关信息后点击确定。 |
| 预期结果 | 系统提示修改成功并刷新页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 测试用例维护

1. 及时添加新的测试用例。在系统开发周期过程中，一些需求可能会被添加进来，而已有的测试用例可能覆盖不到新的需求，此时需要增加测试用例来覆盖到新的需求中。

### 及时删除多余的测试用例。在测试用例设计时，由于考虑不周可能导致多组测试用例经过相同的执行通路，此时需要剔除多余的测试用例保持一组即可。

1. 改进测试用例。在系统开发周期过程中，可能在初期一组测试用例覆盖多条测试路径， 但由于后期系统改进导致此组测试用例覆盖度不如之前，但也覆盖到多数，这是需要改进测试用例。
2. 及时删除无效的测试用例。在系统开发周期过程中，可能由于功能的删除而导致此组测试用例无效，这时就需要删除这组无效的测试用例。
   1. **设计工作总结**

# 总结与展望

### 本次实现了xxxx作品展示系统，系统从零到一按照传统软件工程方法经历了一个漫长的过程。

首先是可行性研究和需求分析，在需求分析过程中，通过分析划分出 8 个功能模块， 然后对每个功能模块进行具体分析。然后进行总体设计与详细设计，通过在之前需求分析中得出的结果，发现可能作品评分模块这里比较难实现，它涉及到的方面比较多，通过仔细分析各个功能之间的关联还是解决的这里的问题。接下来进行编码，这部分也应该是决定系统能不能正确呈现的最后关键一步，在采用 SSM 框架进行编码过程中，本人也对这套框架背后的思想理解的愈发深刻，分层是为了让复杂的问题简单化，由于每一层的职责不一样，所以一个复杂的问题可以拆分成若干简单的问题。最后是进行测试，测试的重要性怎样强调都不为过，一切从用户的角度出发，比如在用户注册时，如果用户在填写密码时没有六位，那可能造成用户的密码很容易被破解，这对于任何一个系统来说都是不可接受的。

伴随着论文的编写，这里面也有很多值得总结的地方。比如对于一个英文名字中大小写约定，应该要以官方为准。所有的西文符和数字在最后都要统一设定字体。论文的内容中也不能出现口语化的东西等。

* 1. **未来工作展望**

由于个人能力有限，作品展示系统还存在一些缺陷，有些功能还需要更进一步去优化和完善，如增加用户数据分析和统计的功能，使用 EChart 等插件，可以更加直观地了解到不同的角色用户偏爱什么类型的作品。方便用户更好的体验系统。还有增加用户评论功能，使所有用户都能对作品进行评论，提高本系统的交互性。对于系统的 UI 设计上，还可以更丰富与美观，使用更加精美的框架或者样式，增加更多的内容。另外，由于个人对于需求调研的经验不足以及需求分析样本数量不够，可能存在一些设计不合理的地方，本人计划在以后的实战演练中对它进行不断的改进和完善，这样也能提升自己的技术水平。

# 谢 辞

# 参考文献

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

[3]王善勤,吴昌雨,邹军国.基于 Java EE 的大学生作品展示系统[J].南昌工程学院学报,2017,34(3):42-46. [4]于化礼. 浅析“大学生艺术作品展示 App 平台”建设及影响[J]. 明日风尚, 2019(3).

梁俊, 奚泱, 李雁波, et al. 智慧工地物联网数据传输与展示系统[J]. 科技创新与品牌, 2019(2).

1. 徐平, 沙丛术, 马登军, et al. 电喷发动机燃油喷射展示系统的设计与开发[J]. 河南工程学院学报(自然科学版), 2019(2).
2. 李艳, 方娟, 董帅. 基于先进可视化引擎的态势展示系统研究与实现[J]. 无线互联科技, 2019(8). [7]魏曦, 孙艳芝. 基于自动建模的室内实景三维展示系统设计与实现[J]. 科技视界, 2019(12).
3. 杨丰宇, 于本成. 基于互联网+ 的企业推广平台设计与研究[J]. 电脑知识与技术, 2018, 014(006):61-62.
4. 赵春刚. UML 实用基础教程[M]. 北京大学出版社, 2018.
5. 薛均晓 李占波. UML 系统分析与设计[M]. 机械工业出版社, 2018. [11]陶华亭. 管理信息系统实用教程(第 2 版)[M]. 电子工业出版社, 2017.
6. 赵新苗, 谢清辉, 颜依铃, 等. 基于 Django 的干果超市信息管理系统[J]. 微型电脑应用, 2020(1):26-29.
7. 杜文洁, 景秀丽. 计算机专业指导教程[M]. 清华大学出版社, 2017. [14]陈承欢. UML 与 Rose 软件建模案例教程[M]. 人民邮电出版社, 2017.

[15]郭文俊. 站酷:靠设计师社区掘金[J]. 财会月刊:财富文摘, 2017. [16]娄月. 站酷:让设计更有价值[J]. 创业家, 2018(9).

[17]章仁棠. 基于 Web 三层架构的高校科研管理系统的设计与实现[J]. 信息通信, 2019, 195(03):187-188. [18]邓以彬. 基于三层架构的中小学校人事管理系统研究与实现[D]. 2019.

1. 郑强程. 基于 SSM 应急资源管理系统的设计与实现[D]. 2019.
2. 杨帆. 基于 Spring 和 MyBatis 框架的物业管理系统的设计与实现[D]. 2019.
3. 陈倩怡, 何军. Vue+Springboot+MyBatis 技术应用解析[J]. 电脑编程技巧与维护, 2020(1):14-15.
4. 刘杰, 孙浩, 郭东旭,等. 基于Spring MVC 及MyBatis 框架的在线教育平台的设计与实现[J]. 沈阳师范大学学报(自然科学版), 2019(3).
5. 齐琦. 基于云环境下的家校协作平台的设计与实现[D]. 2019.
6. 杨正午. 基于 WEB 前端开发技术的网站设计——以连锁超市商品销售管理系统为例[J]. 山西科技, 2019, 34(03):56-58+62.
7. 魏鹏. 基于 Web 的超市销售管理系统的设计与实现[D]. 2019.
8. Hollinger J P , Peirce J L , Poe G A . SSM/I instrument evaluation[J]. IEEE Transactions on Geoscience & Remote Sensing, 2018, 28(5):781-790.
9. Ryu J , Kamata S I . SSM-HPC: Front View Gait Recognition Using Spherical Space Model with Human Point Clouds[C]// 18th IEEE International Conference on Image Processing, ICIP 2011, Brussels, Belgium, September 11-14, 2017. IEEE, 2017.
10. Anatoly Akkerman, Alexander Totok, Vijay Karamcheti. Infrastructure for Automatic Dynamic

Deployment of J2EE Applications in Distributed Environments[C]// Component Deployment, Third International Working Conference, Cd, Grenoble, France, November. Springer-Verlag, 2018.

1. Cheema H , Watson J , Shinde P S , et al. Precious Metal-Free Solar-to-Fuel Generation: SSM-DSCs Powering Water Splitting with NanoCOT and NiMoZn Electrocatalysts[J]. Chemical Communications, 2020. [30]Chang-Zhi L I , Li L I , Guo-Qing Y , et al. Design and Implementation of Two-Dimensional Code Meal Ticket Management System for Staff Canteen Based on SSM[J]. computer knowledge and technology, 2019. [31]Jie L , Hao S , Dongxu G , et al. Design and implementation of online education platform based on Spring MVC and MyBatis framework[J]. Journal of Shenyang Normal University(Natural ence Edition), 2019.

# 附录 A 外文翻译—原文部分

**Original source:** International Working Conference on component deployment CD 2018: component deployment page 17-32

**Infrastructure for Automatic Dynamic Deployment**

**Of J2EE Application in Distributed Environments**

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

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

#### Introduction

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

* + Using properly designed applications, vertical distribution across wide-area networks improves user-perceived latencies.
  + Wide-area vertical layering requires replication of application components and maintaining consistency between replicas.
  + Additional replicas may be deployed dynamically to handle new requests.
  + Different replicas may, in fact, be different implementations of the same component based on usage (read-only, read-write).
  + New request paths may reuse components from previously deployed paths.

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

Through deployment of additional application components dynamically based on active monitoring. However, in order to achieve such dynamic adaptation, we need an infrastructure for automating J2EE application deployment in such an environment. This need is quite evident to anyone who has ever tried deploying a J2EE application even on a single application server, which is a task that involves a great deal of

configuration of both the system services and application components. For example one has to set up JDBC data sources, messaging destinations and other resource adapters before application components can be configured and deployed. In a wide area deployment that spans multiple server nodes, this proves even more complex, since more system services that facilitate inter-node communications need to be configured and started and a variety of configuration data, like IP addresses, port numbers, JNDI names and others have to be consistently maintained in various configuration files on multiple nodes.

This distributed deployment infrastructure must be able to:

* + address inter-component connectivity specification and define its effects on component configuration and deployment,
  + address application component dependencies on application server services, their configuration and deployment,
  + provide simple but expressive abstractions to control adaptation through dynamic deployment and undeployment of components,
  + enable reuse of services and components to maintain efficient use of network nodes’ resources,
  + provide these facilities without incurring significant additional design effort on behalf of application programmers.

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

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

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

Sample J2EE applications – Java Pets tore [23], Rubies [20] and TPC-W-NYU [32]. The infrastructure implementation utilizes the JBoss’s extendable *micro-kernel* architecture, based on the JMX [27] specification. Componentized architecture of JBoss allows incremental service deployments depending on the needs of deployed applications. We believe that dynamic reconfiguration of application servers through dynamic deployment and undeployment of system services is essential to building a resource-efficient framework for dynamic distributed deployment of J2EE applications. The rest of the paper is organized as follows. Section 2

provides necessary background for understanding the specifics of the J2EE component technology which are relevant to this study. Section 3 gives a general description of the infrastructure architecture, while section 4 goes deeper in describing particularly important and interesting internal mechanisms of the infrastructure. Section 5 describes the implementation of the framework, and related work is discussed in section 6.

#### J2EE Background

* 1. **Introduction**

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

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

* + - Component programming model,
    - Component contracts with the hosting server,
    - Services that the platform provides to these components,
    - Various human roles,
    - Compatibility test suites and compliance testing procedures.

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

Specific J2EE standards.

#### Presentation or Web tier.

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

#### Business or EJB tier.

This tier consists of Enterprise Java Beans (EJBs) [24] that model the business logic of the enterprise application. These components provide persistence mechanisms and transactional support. The components in

the EJB tier are invoked through remote invocations (RMI), in-JVM invocations or asynchronous message delivery, depending on the type of EJB component. The EJB specification defines several types of components. They differ in invocation style (synchronous vs. asynchronous, local vs. remote) and statefulness: completely stateless (e.g., Message-Driven Bean), stateful non-persistent

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

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

#### Enterprise Information System (EIS) or Data tier.

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

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

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

#### J2EE Component Programming Model

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

* + - EJBs (session, entity, message-driven),
    - Web components (servlets, JSPs),
    - messaging destinations,
    - Data sources,

EJB and Web components are deployed into their corresponding containers provided by the application server vendor. They have well-defined contracts with their containers that govern lifecycle, threading, persistence and other concerns. Both Web and EJB components use JNDI lookups to locate resources or other EJB components they want to communicate with. The JNDI context in which these lookups are performed is

maintained separately for each component by its container. Bindings messaging destinations, such as topics and queues, are resources provided by a messaging service implementation. Data sources are resources provided by the application server for data access by business components into the enterprise information services (data) tier, and most commonly are exemplified by JDBC connection pools managed by the application

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

#### Links Between Components

* + 1. **Remote Interactions**

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

* Remote EJB invocation: synchronous EJB invocations through EJB Home and EJB Object interfaces.
* Java Connector outbound connection: synchronous message receipt, synchronous and asynchronous message sending,

Database query using Connection Factory and Connection interfaces.

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

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

* Deployment of target component classes (optional for some components, like destinations),
* Creation of a special Java object to be used as a target component’s proxy,
* Binding of this object with component’s host naming service (JNDI or JMX),
* Start of the target component,
* Deployment of referencing component classes,
* Creation and population of referencing component’s run-time context in its host naming service,
* start of the referencing component.

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

* Ability to deploy messaging destinations and data sources dynamically,
* Ability to create and bind into JNDI special objects to access messaging destinations and data sources,
* Ability to specify initial binding of EJB Home objects upon EJB component deployment,
* Ability to specify a JNDI reference 5 in the referencing component’s run-time context to point to the EJB Home binding of the referenced EJB component. In our infrastructure which is limited to homogeneous application servers, these options are sufficient to control intercomponent links through simple deployment descriptor manipulation. However, in context of heterogeneous application servers, simple JNDI references and thus simple descriptor manipulation are insufficient due to cross-application-server

Classloading issues.

#### Local Interactions

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

#### Deployment of J2EE Applications and System Services

* + 1. **Deployment of Application Components**

Deployment and undeployment of standard J2EE components has not yet been standardized (see JSR 88

1. for standardization effort 6). Therefore, each application server vendor provides proprietary facilities for component deployment and undeployment. And while the J2EE specification does define packaging of standard components which includes format and location of XML-based deployment descriptors within the package, this package is not required to be deployable by an application server without proprietary transformation. Examples of such transformation are
   * Generation of additional proprietary descriptors that supplement or replace the standard ones,
   * Code generation of application server-specific classes.

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

Bundled into an archive. The archive may optionally include Java classes that implement the component and any other resources that the component may need. Alternatively, the deployment descriptors may simply

have URL references to codebases. We assume presence of a dynamic deployment/undeployment service on all compliant J2EE servers and a robust application server classloading architecture capable of repeated deployment cycles without undesired classloading-related issues. Most modern application servers (e.g., JBoss

1. and Geronimo [1]) do provide such facilities.

#### Deployment of System Components (Services)

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

# 附录B 外文翻译—译文部分

原文出处：组件部署国际工作会议 CD 2018：组件部署 第 17-32 页

基于J2EE 在分布式环境下的底层结构的自动动态配置的应用

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

关键词：J2EE；动态配置；分布式；组件；

1. 前言

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

* 1. 应用合适的应用程序，在广域网中的垂直负荷可以察觉的延迟。
  2. 广域垂直层需要复制应用层组件而且需要维持和原组件间的一致性。
  3. 新加的复制组件可以被动态配置以满足新的需要。
  4. 事实上，不同的复制组件可能会根据应用不同的方式实现相组件。
  5. 新的请求路径可以复用先前的组件配置路径。

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

1. 声明内部组件一致性规范和定义它对组件配置部署的影响。
2. 声明应用程序组件对应用服务器，以及它们的配置和部署的依赖性。
3. 提供简单但可表达的抽象方法去控制通过部署和拆卸组件获得的适用性。
4. 能够复用服务和组件从而高效的利用网路节点资源。
5. 提供上述便利而不会增加应用程序员的设计负担。

在本论文中，我们提出自动动态部署 J2EE 应用程序的框架涉及了上面的所有问题，这种框架为组件定义了结构描述语言，链接说明和集合。这种组件说明语言用来描述应用程序组件和链接，它使得应 用组件与系统组件中清晰的分开。一种灵活的系统类型用来定义组件接口和端口的兼容性。一种为配置 组件属性而开发的定义和表述语言允许内部组件间独立的规范和组件间属性的继承。组件集合语言允许 先前定义的复制的组件通过连接合适的端口集合到应用路径，连接时通过链接复制对象和具体把这些复

制组件映射到目标应用服务器节点。组件配置过程评估了应用程序路径的正确性，确认在系统组件上的应用组件的独立性和完成复制组件的部署。根据这些配置使先前部署的复制组件在新的路径中得以匹配和复用的努力正在做出。我们把这种架构作为 JBoss 开源 java 应用服务器的一部分加以实现，在几个J2EE 样本程序比如 Java PetStore,，RUB 和 TPC\_W\_NYU 中进行测试。这种架构实现利用了 JBoss 的可扩展的微内核结构，基于 JMX 规范。JBoss 的组件结构允许根据部署应用程序的需要增加服务配置。我们相信通过动态部署和拆卸系统服务来重构应用服务器对构建高效资源框架的动态分布部署的 J2EE 应用程序来说是非常必要的。本文如下部分是这样组织的。第 2 部分提供了必要的背景以理解和研究有关的J2EE 组件技术规范。第 3 部分对这种架构给出了一般性的描述。第 4 部分更深入的描述了有关这种架构特别重要的和有趣的内部机制。第五部分描述了如何实现这种架构，相关联的工作将在第六部分介绍。

#### J2EE 背景知识

* 1. 介绍

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

* + 1. 组件编程模型。
    2. 组件和主服务器的链接。
    3. 服务器提供给组件的服务。
    4. 各种各样的人物角色。
    5. 兼容性检验装置和编译测试程序。

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

1、表现层或者网络层

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

2、业务层或者 EJB 层

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

被 EJB 部署者绑定在 JNDI 中。EJB 对象允许或者本地 EJB 对象是特定 EJB 实例的代理。

3、企业信息系统或者数据层

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

* 1. J2EE 组件编程模型

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

* + 1. EJBS（会话，实体，消息驱动）。
    2. Web 组件（Servlet、JSP）。.
    3. 消息目的。
    4. 数据源。

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

* 1. 组件间的链接
     1. 远程交互

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

* + - 1. 远程 EJB 调用：同步的 EJB 调用通过主 EJB 对象和 EJB 对象接口来实现。
      2. Java 连结器的外部连接：同步消息接收，同步和异步消息发送，用连接工厂和连接接口进行数据库查询。
      3. Java 连接器的内部连接：异步消息传递进入消息驱动 Bean 只能使用 Activation Spec 对象。

在前两个实例中，应用组件的开发者不仅书写执行在组件的运行时 JNDI 环境中的对象目录代码，

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

1. 部署目标组件类。
2. 创建一个特定的 Java 对象用作目标组件代理。
3. 用组件的命名服务去绑定目标。
4. 启动目标组件。
5. 部署指定的组件类。
6. 在主机的命名服务中，创建和进行指定组件的运行环境。
7. 启动指定的组件。

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

1. 动态部署消息目的和数据源的能力。
2. 创建和绑定特定的 JNDI 目标去访问消息目的和数据源的能力。
3. 把初始绑定的 EJB 对象到 EJB 部署组件的能力。
4. 用在参考组件运行环境中的 JNDI 指引去指出绑定的参考 EJB 的能力。

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

* + 1. 本地交互

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

* 1. 部署 J2EE 应用程序和系统服务
     1. 部署应用程序组件

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

* + - 1. 支持或者取代标准所有者解释器的新的所有者解释器的产生。
      2. 具体服务应用程序类的代码的更替。

为了着手构建一个能够部署不可网络的动态的分布式的架构，我们提出了一种普遍的部署单元即一 个简单的基于 xml 部署的解释器或者是一组类似的绑定到文档中的解释器。文档可能包含用于执行组件的Java 类或者任何其它的所需组件。相应地，部署解释器也可以简单地用 URL 来索引代码。我们假

设这种动态的部署和拆卸服务存在于所有的兼容的J2EE 服务器上而且在不理解类重载相关问题时一个健壮的类重载结结构的应用服务器就能够重复的部署生命周期。大多数现代的应用服务器都提供这样的 功能。

* + 1. 部署系统组件

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

# 附录C 软件使用说明书

### 本系统通过指定 URL 即可进入网站主界面。

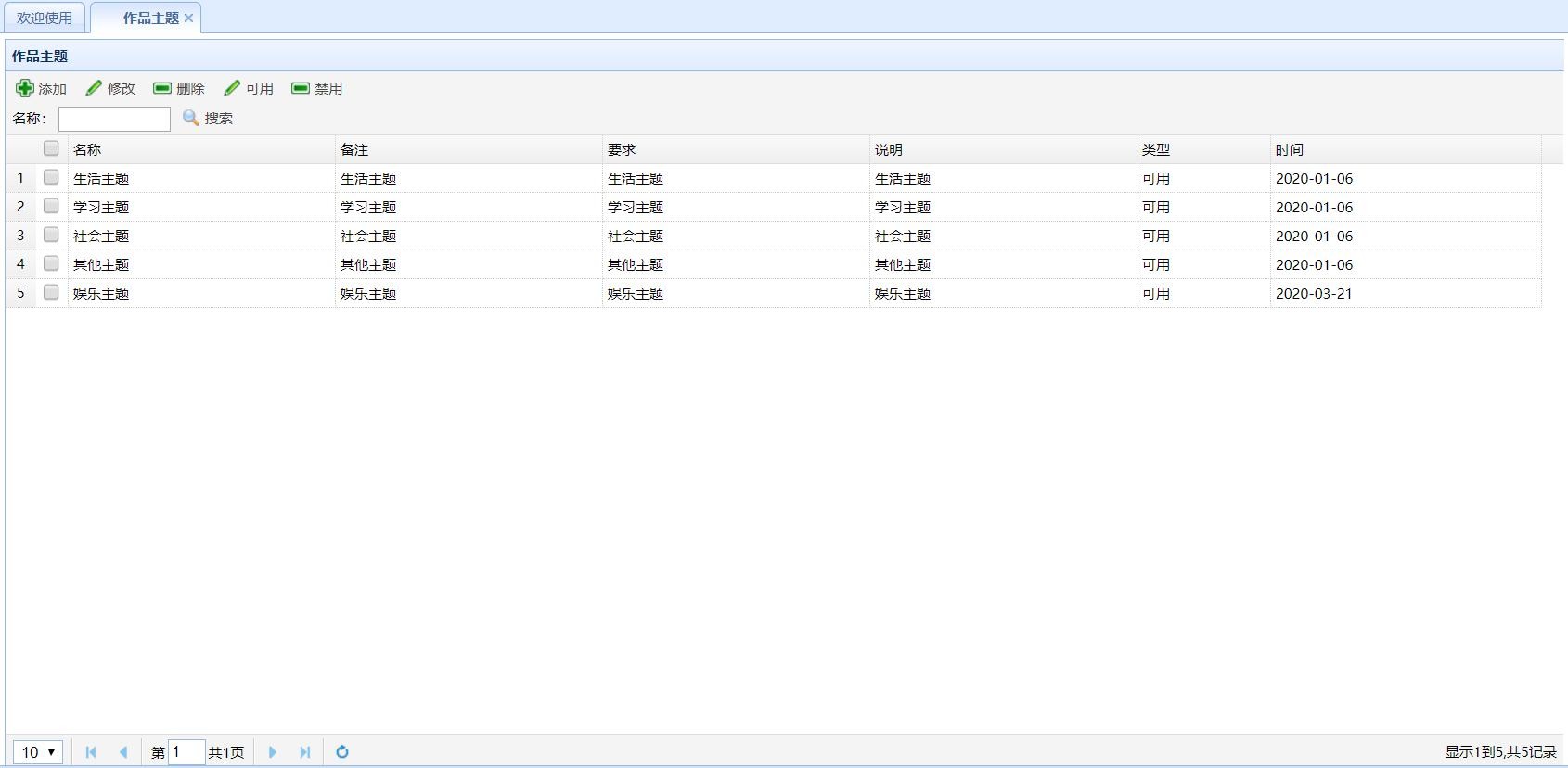
管理员登录后即可进入管理员主界面。



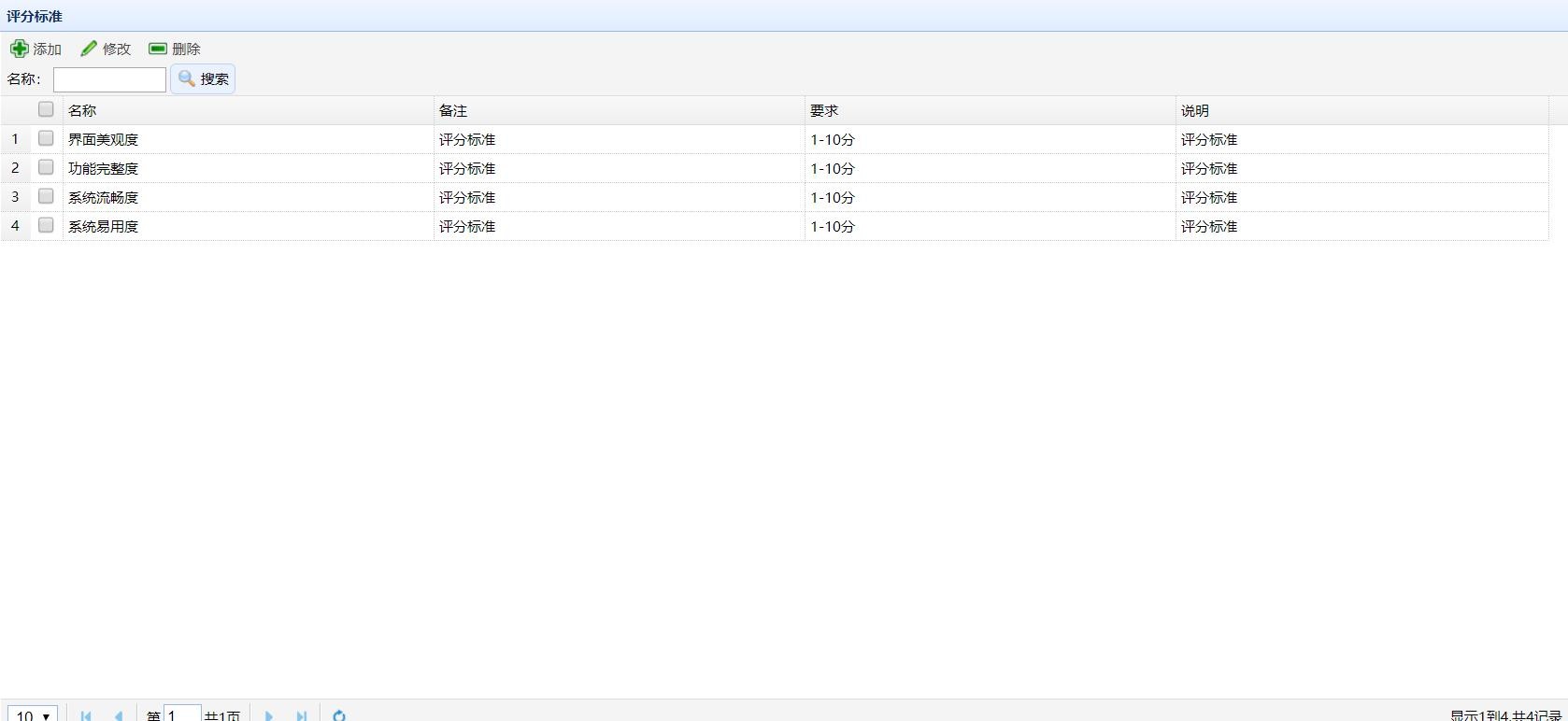
### 非管理员登录后即可进入非管理员主界面，其他角色的界面差不多，不同之处在于他们可以操作的功能不同。



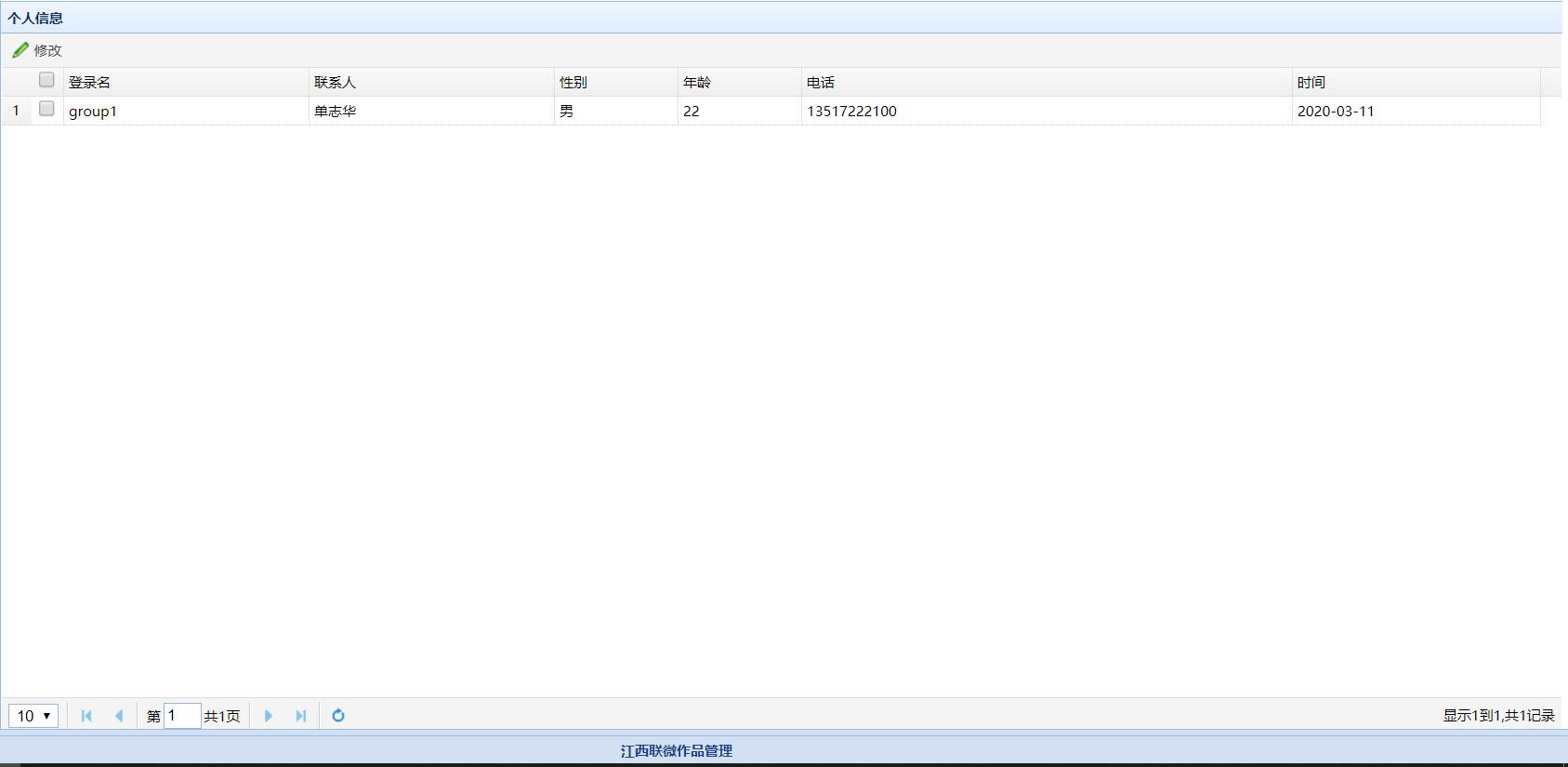
本系统登录即进入作品主题界面。



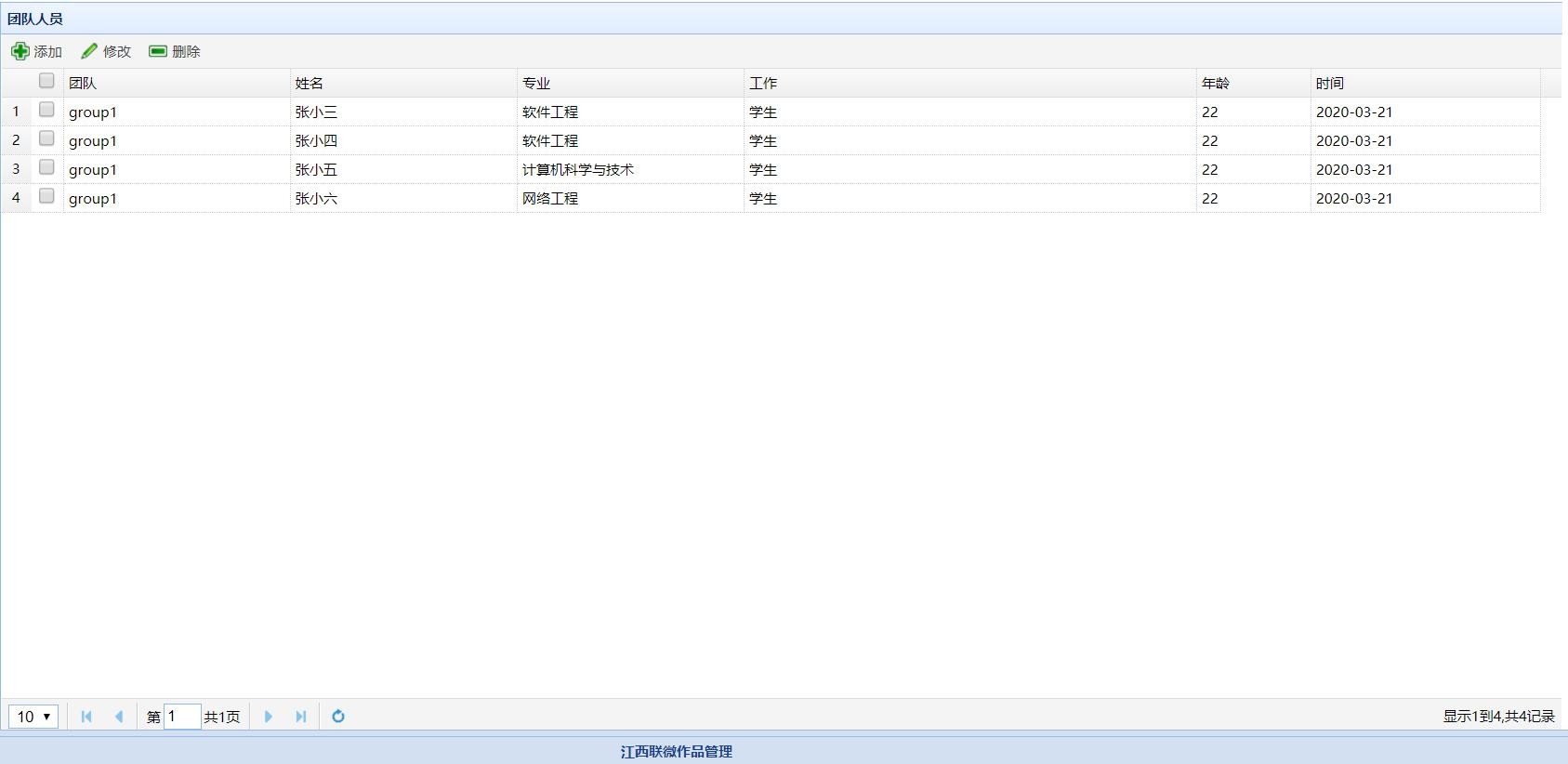
### 本系统登录即进入评分标准界面。



本系统登录即进入个人信息界面。

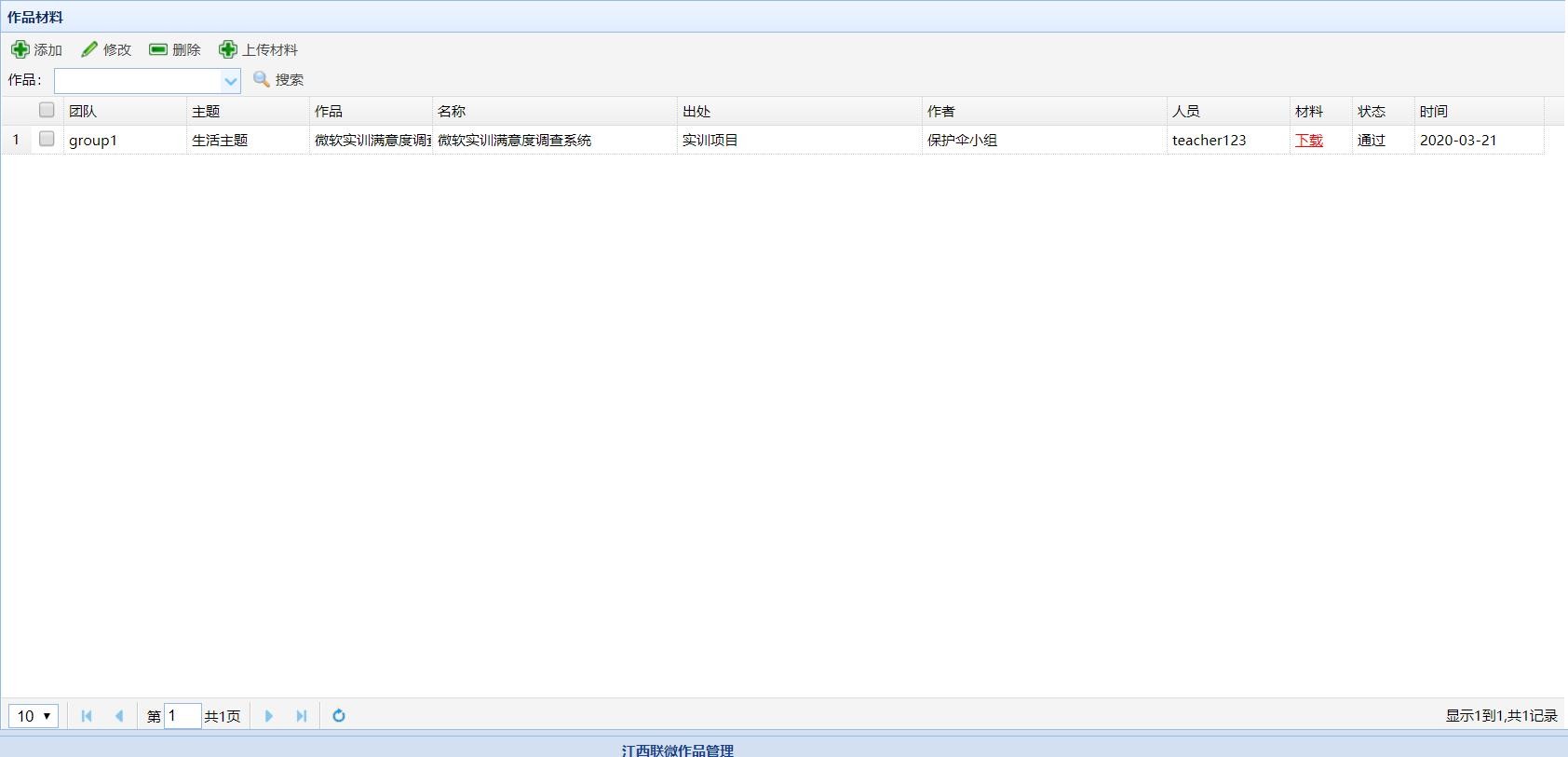


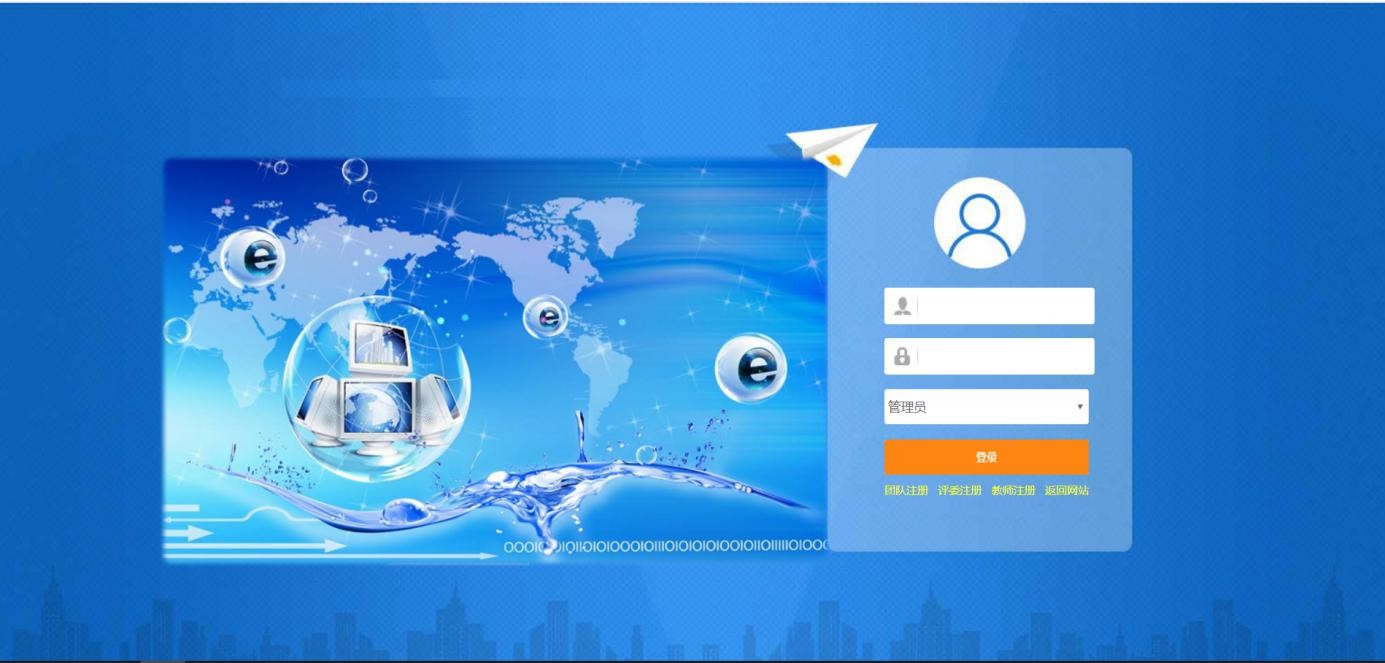
### 本系统登录即进入团队信息界面。



本系统登录即进入作品信息界面。

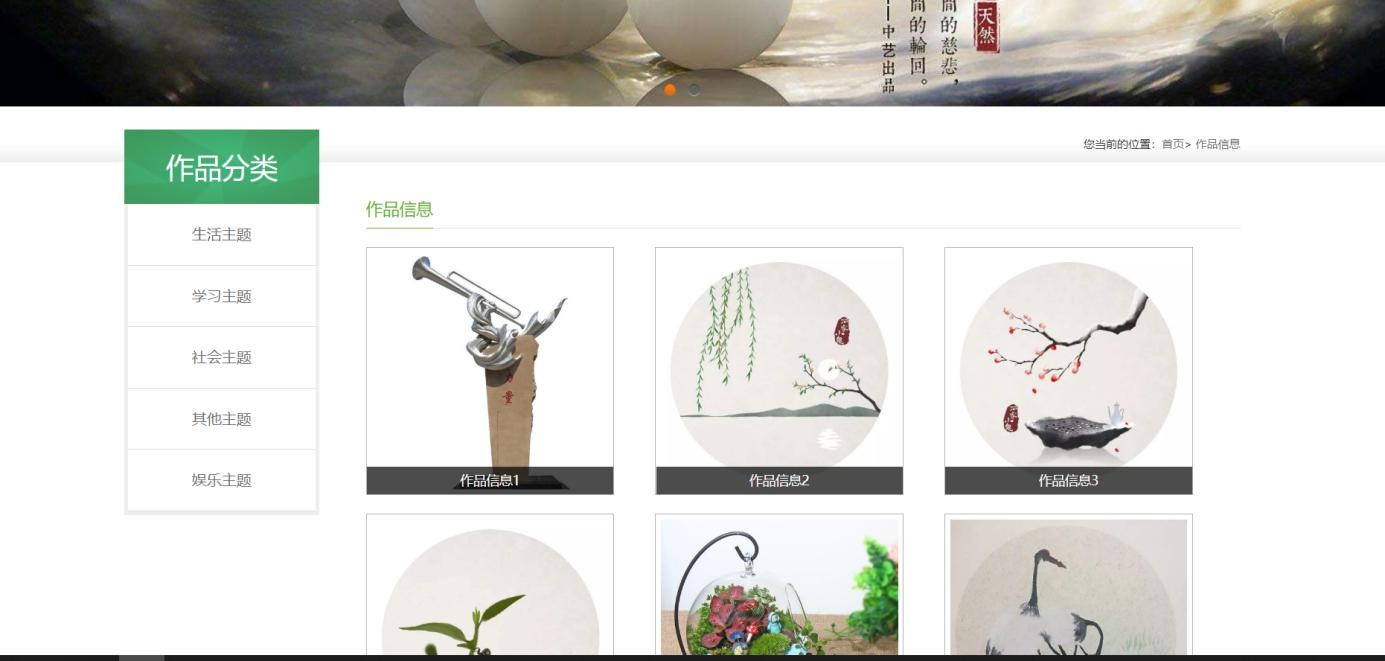
### 本系统登录即进入作品材料界面。



本系统首页选择可进入登录界面。

### 本系统登录界面可选择进入注册界面。



本系统登录即进入分类展示界面。

### 本系统通过展示界面可选择进入详情界面。



<!-- 数据源配置 -->

# 附录 D 主要源代码

### <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">

<property name="driverClassName" value="${jdbc.driver}" />

### <property name="url" value="${jdbc.url}" />

<property name="username" value="${jdbc.username}" />

### <property name="password" value="${jdbc.password}" />

<property name="maxActive" value="10" />

### <property name="maxIdle" value="5" />

</bean>

### <!-- 配置 sqlSessionFactory -->

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

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

<!-- 加载 sqlMapConfig.xml -->

### <property name="configLocation" value="classpath:sqlMapConfig.xml"></property>

</bean>

### <!-- 系统开始 -->

<!-- 配置 AdminDao -->

### <bean id="adminDao" class="com.dao.AdminDao">

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

### </bean>

@RequestMapping("/mimaAdmin")

### public void mimaAdmin(HttpServletRequest request, HttpServletResponse response) throws Exception {

try {

### JSONObject result = new JSONObject();

String adminPassword = (String) request.getParameter("adminPassword"); String adminPassword1 = (String) request.getParameter("adminPassword1"); Admin admin = new Admin();

### admin.setAdminName("admin"); admin.setAdminPassword(adminPassword);

if (adminService.queryAdmins(admin, 0, 0).size()==1) {

### admin = (Admin)(adminService.queryAdmins(admin, 0, 0)).get(0); admin.setAdminPassword(adminPassword1); adminService.modifyAdmin(admin);

request.setAttribute("error", "密码修改成功！"); request.getRequestDispatcher("adminmima.jsp").forward(request,

### response);

}else{

### result.put("success", "true");

request.setAttribute("error", "原密码错误，请重新输入！"); request.getRequestDispatcher("adminmima.jsp").forward(request,

### response);

}

### } catch (Exception e) { e.printStackTrace();

}

### }

@Repository

### public class AdminDao extends SqlSessionDaoSupport{ @Autowired

private AdminMapper adminMapper;

### public List getAdminList(Admin record,int page,int rows) { List<Admin> list = adminMapper.selectAll(record,page,rows); return list;

}

### public Admin getAdminById(int id){

Admin admin = adminMapper.selectByPrimaryKey(id);

### return admin;

}

### public void update(Admin admin) { adminMapper.updateByPrimaryKey(admin);

}

### public void delete(Integer id) { adminMapper.deleteByPrimaryKey(id);

}

### public void add(Admin admin) { adminMapper.insert(admin);

}

### }

public interface AdminMapper {

### /\*\*

* This method was generated by MyBatis Generator. This method corresponds to the database table t\_admin

### @mbggenerated

\*/

### int deleteByPrimaryKey(Integer adminId);

/\*\*

### This method was generated by MyBatis Generator. This method corresponds to the database table t\_admin

* @mbggenerated

### \*/

int insert(Admin record);

### /\*\*

* This method was generated by MyBatis Generator. This method corresponds to the database table t\_admin

### @mbggenerated

\*/

### Admin selectByPrimaryKey(Integer adminId);

/\*\*

### This method was generated by MyBatis Generator. This method corresponds to the database table t\_admin

* @mbggenerated

### \*/

List<Admin> selectAll(@Param("admin")Admin record,@Param("page")int page,@Param("rows")int rows);

### /\*\*

* This method was generated by MyBatis Generator. This method corresponds to the database table t\_admin

### @mbggenerated

\*/

### int updateByPrimaryKey(Admin record);

}

### @Service

public class AdminService { @Autowired

### private AdminDao adminDao;

public List queryAdmins(Admin record,int page,int rows) {

### // TODO Auto-generated method stub

return adminDao.getAdminList(record,page,rows);

### }

public Admin getAdmin(int parseInt) {

### // TODO Auto-generated method stub return adminDao.getAdminById(parseInt);

}

### public void modifyAdmin(Admin admin) {

// TODO Auto-generated method stub adminDao.update(admin);

### }

public void deleteAdmin(Integer id) {

### // TODO Auto-generated method stub adminDao.delete(id);

}

### public void save(Admin admin) {

// TODO Auto-generated method stub adminDao.add(admin);

### }

}

### <script type="text/javascript"> function check() {

var adminPassword = document.form1.adminPassword.value; var adminPassword1 = document.form1.adminPassword1.value; var adminPassword2 = document.form1.adminPassword2.value; var reg = /^[a-zA-Z0-9\_]{6,}$/;

### if (document.form1.adminPassword1.value == "") { alert("请输入确认密码"); document.form1.adminPassword1.focus(); return false;

}

### if (document.form1.adminPassword1.value != document.form1.adminPassword2.value) {

alert("两次输入密码不一致"); document.form1.adminPassword2.focus(); return false;

### }

}

### </script>

<body>

### <div id="login">

<div class="logo">修改密码</div>

### <div class="main">

<form class="formname" action="<%=basePath%>mimaAdmin" name="form1" method="post" onSubmit="return check()">

### <ul>

<li>原密码： <input class="wa" type="password" name="adminPassword" id="adminPassword"/></li>

### <li>新密码： <input class="wa" type="password" name="adminPassword1" id="adminPassword1"/></li>

<li>再输入： <input class="wa" type="password" name="adminPassword2" id="adminPassword2"/></li>

### <li><input type="submit" value="修 改" class="bnt" style="width:48%"

/>&nbsp;<input type="reset" value="重 置" class="bnt" style="width:48%" /></li>

### <li><font color="red">${error }</font></li>

</ul>

### </form>

</div>

### </div>