**数据挖掘实践教学平台研究与实现**

**详细设计说明书**

目录

[1. 引言 5](#_Toc495501660)

[1.1 背景 5](#_Toc495501661)

[1.2 目的 5](#_Toc495501662)

[1.3 数据挖掘概览 5](#_Toc495501663)

[1.3.1 概念 5](#_Toc495501664)

[1.3.2 步骤 5](#_Toc495501665)

[1.3.3 系统需求 7](#_Toc495501666)

[2. 功能需求 7](#_Toc495501667)

[2.1 数据集管理部分 7](#_Toc495501668)

[2.2 学生信息管理部分 8](#_Toc495501669)

[2.3 实践任务管理部分 8](#_Toc495501670)

[2.4 数据挖掘算法库管理部分 9](#_Toc495501671)

[3. 系统总体设计 9](#_Toc495501672)

[3.1 构架 9](#_Toc495501673)

[3.1.1 Web三层架构 10](#_Toc495501674)

[3.1.2 表现层 11](#_Toc495501675)

[3.1.3 业务层 12](#_Toc495501676)

[3.1.4 数据访问层 13](#_Toc495501677)

[3.2 系统结构 14](#_Toc495501678)

[3.3 系统流程 15](#_Toc495501679)

[3.3.1 登录流程 15](#_Toc495501680)

[3.3.2 系统功能选择流程 16](#_Toc495501681)

[3.3.3 数据集管理模块流程 17](#_Toc495501682)

[3.3.4 算法库管理流程 18](#_Toc495501683)

[3.3.5 学生管理流程 19](#_Toc495501684)

[3.3.6 实践任务管理流程 20](#_Toc495501685)

[4. 数据库设计与构架 21](#_Toc495501686)

[4.1 E-R图 21](#_Toc495501687)

[4.1.1 登录模块 22](#_Toc495501688)

[4.1.2 数据集管理 23](#_Toc495501689)

[4.1.3 算法库管理 24](#_Toc495501690)

[4.1.4 学生管理 25](#_Toc495501691)

[4.1.5 实践任务分配 26](#_Toc495501692)

[4.2 数据库逻辑结构设计 27](#_Toc495501693)

[4.2.1 权限 27](#_Toc495501694)

[4.2.2 算法库管理 28](#_Toc495501695)

[4.2.3 实践任务管理 28](#_Toc495501696)

[4.2.4 学生管理 29](#_Toc495501697)

[4.2.5 数据集管理 30](#_Toc495501698)

[5. 模块设计 32](#_Toc495501699)

[5.1 详细模块设计 32](#_Toc495501700)

[5.1.1 登录注册模块 32](#_Toc495501701)

[5.1.2 数据集管理模块 33](#_Toc495501702)

[5.1.3 学生信息管理模块 35](#_Toc495501703)

[5.1.4 实践任务管理模块 36](#_Toc495501704)

[5.1.5 数据挖掘算法库管理部分 37](#_Toc495501705)

[6. 项目开发 38](#_Toc495501706)

[6.1 技术方案 38](#_Toc495501707)

[6.1.1 Sring Java Bean管理 38](#_Toc495501708)

[6.1.2 Spring MVC支撑下的数据驱动技术 39](#_Toc495501709)

[6.1.3 Vue.js UI渐进式界面开发 40](#_Toc495501710)

[6.1.4 前后端联调的分离模式 41](#_Toc495501711)

[6.2 关键模块的实现技术及数据结构等 42](#_Toc495501712)

[6.2.1 访问层数据源的配置 42](#_Toc495501713)

[6.2.2 ORM映射关系的建立 42](#_Toc495501714)

[6.2.3 DAO层的泛型的支持 44](#_Toc495501715)

[6.2.4 Controller层的业务委托 45](#_Toc495501716)

[6.2.5 DTO传输的暴露策略 46](#_Toc495501717)

[6.3 核心功能实现 46](#_Toc495501718)

[6.3.1 登录授权模块 46](#_Toc495501719)

[6.3.2 数据集的上传与下载 49](#_Toc495501720)

[6.3.3 数据挖掘任务创建与分配 52](#_Toc495501721)

[6.3.4 算法库管理集 52](#_Toc495501722)

[6.4 特殊问题及解决办法 55](#_Toc495501723)

[6.4.1 系统整合框架版本兼容冲突 55](#_Toc495501724)

[6.4.2 Hibernate的Session关闭 55](#_Toc495501725)

[6.4.3 SQL表产生了多余字段 56](#_Toc495501726)

[6.4.4 API鉴权消息摘要不一致 56](#_Toc495501727)

[6.4.5 Options请求造成的403状态的跨域请求问题 56](#_Toc495501728)

[6.4.6 文件上传中的请求格式与文件写入出错 56](#_Toc495501729)

[6.4.7 页面快速切换情况下的API并发访问授权错误 57](#_Toc495501730)

[6.4.8 列表分页控件崩溃 57](#_Toc495501731)

[6.4.9 浅拷贝使请求参数增加额外字段 57](#_Toc495501732)

[6.4.10 Vue计算属性与Prop属性切换错误 57](#_Toc495501733)

[6.4.11 三级联动勾选文件的列表显示出错 57](#_Toc495501734)

[6.4.12 循环渲染组件或表单元素出错 58](#_Toc495501735)

[6.4.13 提取后台数据方法的API书写问题 58](#_Toc495501736)

[6.4.14 导航栏的路由生成错误 58](#_Toc495501737)

[7. 个人体会 58](#_Toc495501738)

[7.1 成员1 刘祥德的个人小结 58](#_Toc495501739)

# 引言

## 背景

近年来，数据挖掘引起了信息产业界的极大关注，每时每刻产生的海量数据存在极具价值的分析发掘潜能。其目的是发现未知的关系和以数据拥有者可以理解并对其有价值的新颖方式来总结数据，广泛用于包括商务管理，生产控制，市场分析，工程设计和科学探索等各种应用。

## 目的

本系统意在提供一个为老师、学生提供一个实践教学平台，管理员可以在平台给学生分配指定类型的数据发掘任务，并同步追踪学生的数据发掘进度，进行有效的教学评估。学生可以根据任务在平台上传数据集，进行数据挖掘流程的分类、估计、预测、分析、总结工作，并上传期望进度，汇报发掘结果。系统会提供诸如图形、表格的数据呈现手段，帮助教学任务的较好完成。

## 数据挖掘概览

### 概念

数据挖掘是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程。数据挖掘可以描述为:按企业既定业务目标，对大量的企业数据进行探索和分析，揭示隐藏的、未知的或验证己知的规律性，并进一步将其模型化的有效方法。数据挖掘相近的同义词包括：数据融合、数据分析和决策支持等。

### 步骤

**1.业务理解：**

#### 确定业务目标：分析项目的背景，从业务视点分析项目的目标和需求，确定业务角度的成功标准；

#### 项目可行性分析：分析拥有的资源，条件和限制，风险估计，成本和效益估计；

#### 确定数据挖掘目标：明确确定数据挖掘的目标和成功标准，数据挖掘的目标和业务目标是不一样的，前者指技术上的，例如生成一棵决策树等；

#### 提出项目计划：对整个项目做一个计划，初步估计用到的工具和技术。

**2.数据理解：**

#### 收集原始数据：收集本项目所涉及到的数据，如有必要，把数据装入数据处理工具，并作一些初步的数据集成的工作，生成相应报告;

#### 描述数据：对数据做一些大致的描述，例如记录数、属性数等，给出相应报告；

#### 探索数据：对数据做简单的统计分析，例如关键属性的分布等；

#### 检查数据质量：包括数据是否完整、数据是否有错、是否有缺失值等问题。

**3.数据准备：**

数据清洗：

#### 数据的完整性：例如PM2.5的属性缺少风向、气压等属性；

#### 数据的唯一性：例如不同来源的数据出现重复的情况；

#### 数据的权威性：例如同一个指标下出现多个不同来源的数据且数值不同；

#### 数据的合法性：例如获取的数据与常识不符，如风向出现南北的方向；

#### 数据的一致性：例如不同来源的不同指标，实际内涵是一样的，如气压和压强,或者同一指标内容不一致；

针对方案：

#### 通过其他数据补全：如通过身份证号补全一个人的籍贯，出生年月；

#### 通过前后数据补全：如通过前后时间均值来补全缺失时间；

#### 舍弃补不全的数据：从列表剔除残缺的数据，但仓库里予以保留；

数据的唯一性：

#### 按主键去整理：保留唯一数据；

#### 按算法去整理：对不同来源的数据按一定规则合并或者只保留一个；

#### 数据的权威性：对数据来源进行权威性分级；

#### 数据的合法性：

#### 值域合法性：设定数值上下界来确保数据值的合法性；

#### 值类型合法性：如日期格式必须为2017/5/6；

（2）数据集成：

在原有数据的基础上生成新的属性或记录，利用表连接等方式将几个数据集合并在一起，把数据转换成适合数据挖掘处理的格式，将数据放到数据仓库中。

**4.建立模型阶段：**

#### 数据挖掘算法和参数选取：确定数据挖掘算法和参数，可能会利用多个算法；

#### 测试方案设计：设计某种测试模型的质量和有效性的机制；

#### 模型训练：在准备好的数据集上运行数据挖掘算法，得出一个或者多个模型；

#### 模测试评估：根据测试方案进行测试，从数据挖掘技术的角度确定数据挖掘目标是否成功。

**5.模型评估：**

#### 结果评估：从商业角度评估得到的模型，甚至实际试用该模型测试其效果；

#### 过程回顾：回顾项目的所有流程，确定每一个阶段都没有失误；

#### 确定下一步工作：根据结果评估和过程回顾得出的结论，确定是部署该挖掘模型还是从某个阶段开始重新开始。

### 系统需求

# 功能需求

本系统需要为教师提供一个友好的数据挖掘教学平台，让学生可以体验到数据挖掘基本流程。系统应包括数据集管理功能，学生基本信息概览，实践任务分配以及数据挖掘算法纳入等几个大方向的功能。学生可以在系统进行下载数据集，查看所分配的任务，进行数据挖掘任务几个过程。管理员可以管理学生信息，进行任务分配，上传、修改数据集等。

具体的功能包括：

#### 首页登录部分

#### 注册用户信息：学生可以通过页面注册自己的账号及密码。

#### 用户登录：系统根据账号及密码进行登录授权

#### 展示系统基本信息：包括系统的介绍、功能、联系方式、分享方式等。

## 数据集管理部分

学生：

#### 浏览数据集库：可以查看所展示的所有数据集的种类、数量、主题等概括性基本信息，支持根据不同的信息进行分类查看。

#### 浏览数据集：浏览数据集基本描述信息及具体内容，基本的描述信息包括：数据集的名称、年份、实例数、所属的类型（分类、聚类）等。数据集的具体内容可以查询，并且能够按照不同的条目（如年份、类型）进行分类排序浏览。

管理员：

#### 上传数据集：上传数据集的文件，以文件或二进制流的方式保存在服务器或mysql数据库中去。

#### 下载数据集：提供数据集的下载功能。

#### 删除数据集：删除已上传的数据集。提示相应的删除提示：如已被关联的数据集删除警告、数据量大的评估警告。

#### 更新数据集：可更新数据集的基本信息、属性、批量删除部分实例。

#### 文件读入数据集信息：根据管理员上传的属性描述文件，读入数据集的基本信息。

#### 浏览数据集：可以查看所展示的所有数据集的概括性基本信息，支持根据不同的信息进行分类查看。

## 学生信息管理部分

管理员：

#### 添加学生信息：管理员对学生信息进行添加，具体内容包括学生的姓名、学号、班号、专业等。

#### 修改学生信息：管理员对学生信息进行修改。

#### 删除学生信息：管理员对学生信息进行删除。

#### 查询学生信息：管理员对学生信息进行查询。

## 实践任务管理部分

管理员：

#### 任务数据集指定：合理默认为几个任务指定数据集库里的数据集。

#### 任务学生分组指定：合理为默认任务指定几组不同的学生，每组学生每次只能分配一组任务，一个任务可以由多组学生参与。

#### 任务算法配置：为不同类型任务配置适量的算法。

#### 新增任务：新增数据挖掘任务。

#### 修改任务：对任务所分配的数据集、学生、算法进行添加、删除、修改。

#### 跟踪进度：记录学生执行数据挖掘任务的进度，呈现学生任务的阶段。

#### 挖掘评测：调用学生完成任务后得到的结果存储记录，比较结果算法、算法参数、实验方法等因素造成的差异。

学生：

#### 查看分配任务：学生查看系统分配给自己的任务。

#### 提交任务：对阶段性的任务进行提交、总结。

#### 查看提交结果：对已提交的结果进行查看。

## 数据挖掘算法库管理部分

#### 算法浏览：展示系统内数据挖掘算法。支持分类、聚类算法分类浏览，

#### 算法呈现：对具法算法进行阅读。包括算法参数、类型、接口的综合描述。

# 系统总体设计

## 构架

### Web三层架构

图 3‑1 系统构架层次图

### 表现层

#### 模型（Model）：在MVC中模型保持着一个应用程序的状态，和相应视

图中来自用户交互的状态变化。在上图中我们可以看到model会接受来之控制器的状态变化响应和视图的显示状态查询view渲染的数据来源。同时model还有通过事件机制（观察者模式）通知view状态的改变要求view渲染响应。view和模型之间存在一定的耦合，view必须了解model，这也是MVP模式出现原因之一。正在这里的模型model可以是来之分布式soap或者resetfult的DTO（数据传输对象），也可以直接是我们的领域对象（DO）或者数据层返回的数据集，并不严格的要求。

#### 控制器（Controller）：控制器由view触发，响应用户界面的交互，并

根据表现层逻辑改变model状态，以及中间层的交互，最终修改model，controller不会关心视图的渲染，是通过修改model，model的事件通知机制，触发view的刷新渲染。控制器和模型的交互式一种“发出即忘”，或者分布式OneWay的调用。控制器Controller不会主动了了解view和view交互，除了唯一的视图选择外，控制器需要选择下一次显示的视图view是什么。一般控制器会请求全局应用程序路由下一个需要显示的view，在使其呈现出来。

#### 视图（View）：视图是表现层模式出现的原因，因为他的多样性和变化的

频繁性，不易测试，所以理想的视图尽可能透明，被动，视图只负责渲染呈现给用户交互。视图由一系列GUI组件组成，响应用户行为触发控制器逻辑，修改model状态使其保持view同步。视图并需要相应model的变化被动的接受model状态变化刷新相应给用户。

本系统使用以MVVM为设计理念的vue.js新型渐进式框架打造轻便、简洁、流畅的后台系统UI。表现层作为前端主要的内容，只负责页面的渲染与数据可视化加工。在如今restful微服务大行其道的大潮下，小组内负责表现层前端的人员只通过aixos封装的ajax向后台发送GET、PUT、DELETE、PATCH等异步请求，与后端进行数据交互。其中的API与后端约定好swagger生成文档，spring mvc会统一处理所有合法的http请求，以json或xml的数据格式返回、或响应response。

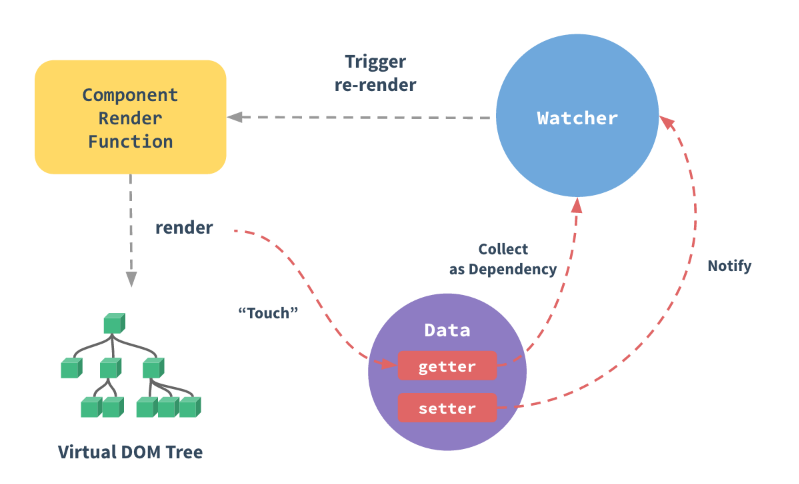


图 3‑2 vue.js框架渲染UI流程图

### 业务层

系统业务层的实现主要体现在各种组合的JavaBean。为体现系统的高内聚与低耦合特性，系统采用spring container管理所有系统的bean,将bean的生命周期托付给spring。在系统启动后，所有组件都被构造生成。利用spring的依赖注入和控制反转的先进技术，诸如DataSetService、StudentService、DataMiningTaskService等承载着复杂业务逻辑的服务性组件会被注入Controllers中去。同时业务层开发亦约定了一系列开发原则。



图 3‑3 MVC模式逻辑控制图

#### 面向接口编程：对于所有service的开发，都有之对应接口声明规范，每个方法体应该提供对应的joc文档注释；

#### 限定数据库事务边界：对于所有与业务逻辑的代码应该定义适当的事务边界，事务的管理应交付给spring 与对应ORM工具匹配的Transation Manager管理。一旦在处理业务逻辑中出现异常，要进行事务回滚。

#### 注解优先：所有业务代码压缩xml的使用比率，统一使用新型简洁的java annotation进行编码；如在后台中所有框架（诸如shiro、spring mvc、hibernate）的整合都应摒弃传统的xml的繁杂分离配置，使用java config全注解代码的优化系统的启动与维护；

#### 全局异常捕捉，返回统一状态：在service层以下发生的异常应该在controllers处进行异常捕捉，每个异常提供专一的处理方法，并在response中纳入对应的错误码或状态码，方便前端根据状态码进行页面逻辑的跳转与渲染。

### 数据访问层

数据访问利用全自动的hibernate对象——关系映射框架，构造整个系统后台的数据库支持，舍弃传统的jdbc繁杂的模板访问代码。

#### 数据库选用：利用开源免费MySQL，主要考虑其强大的事务支持特性，为数据挖掘准备良好的支撑；后期或可加入Redis键——值形式的数据库模型，处理相应的图数据；

#### 数据库连接池：选用c3p0连接池

#### 数据访问：对于DAO层的代码，一主要利用hql面向对象的语句执行CRUD等操作，二看齐JAP统一类型安全的查询规范，少用Criteria已被deprecated的非类型安全的Hibernate API。

#### 性能优化：在重复利用较多数据的上考虑使用二级缓存；对于批量插入数据量比较大的数据库访问操作，考虑绕开Hibernate API，直接使用原生的jdbc语句；建立必要的主键约束跟表索引；

#### 级联控制：用代码进行控制对象实体的关联与级联关系，取代数据库底层的数据表的级联；

## 系统结构



图 3‑4系统功能模块图

## 系统流程

### 登录流程



图 3‑5 登录流程图

### 系统功能选择流程



图 3‑6 系统功能流程图

### 数据集管理模块流程

 图 3‑7 数据集管理模块流程图

### 算法库管理流程

图 3‑8 算法库管理模块流程图

### 学生管理流程



图 3‑9学生管理模块流程图

### 实践任务管理流程



图 3‑10 实践任务管理模块流程图

# 数据库设计与构架

## E-R图

本模块主要展示数据库核心的ER设计关系（限于篇幅，或有细节关系没列出）

### 登录模块



图 4‑1权限验证核心ER图

### 数据集管理



图 4‑2 数据集管理核心ER图

### 算法库管理

图 4‑3 算法库核心ER图

### 学生管理



图 4‑4 学生管理核心ER图

### 实践任务分配

图 4‑5 数据挖掘实践任务分配核心ER图

## 数据库逻辑结构设计

### 权限

1、teacher

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| TeacherId | int | 11 | 否 | 学号 | 主键 |
| TeacherName | Varchar | 20 | 否 | 教师姓名 |  |

2、student

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| studentId | int | 11 | 否 | 学号 | 主键 |
| classId | int | 11 | 否 | 班号 |  |
| professtions | Varchar | 20 | 否 | 专业 |  |

3、user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| userId | Varchar | 32 | 否 | 用户编号 | 主键 |
| userName | Varchar | 20 | 否 | 用户姓名 |  |
| password | Varchar | 45 | 否 | 密码 |  |
| userAcessStatus | Varchar | 45 | 否 | 用户权限状态 |  |

4、role

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| roleId | Varchar | 32 | 否 | 学号 | 主键 |
| roleName | Varchar | 20 | 否 | 班号 |  |
| description | Varchar | 20 | 否 | 专业 |  |
| isAvailable | boolean |  | 是 | 可否为空 | 默认可以 |

5、permission

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| permissionId | Varchar | 32 | 否 | 权限编号 | 主键 |
| perName | Varchar | 20 | 否 | 权限名称 |  |
| description | Varchar | 45 | 否 | 权限描述 |  |
| isAvailable | boolean |  | 否 | 权限可用 | 默认true |

### 算法库管理

1、algorithm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 |  | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| algorithmId |  | Varchar | 32 | 否 | 算法编号 | 主键 |
| type |  | Varchar | 20 | 否 | 算法类型 |  |
| interfaceDesc |  | Varchar | 45 | 是 | 接口描述 | 默认为空 |

2、algorithm\_para

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| algorithmId | Varchar | 32 | 否 | 算法编号 | 外键 |
| paramId | int | 11 | 否 | 参数编号 | 主键 |
| dataType | Varchar | 20 | 否 | 数据类型 |  |
| paramDesc | Varchar | 100 | 是 | 参数描述 | 默认为空 |
| isNecessary | boolean |  | 是 | 是否可选 | 默认为空 |

### 实践任务管理

1. data\_mining\_task

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| taskId | Varchar | 32 | 否 | 任务编号 | 主键 |
| type | Varchar | 20 | 否 | 任务种类 |  |
| taskDesc | Varchar | 45 | 是 | 任务描述 | 默认为空 |
| finishTime | Date |  | 是 | 截止时间 | 默认为空 |
| startTime | Date |  | 是 | 开始时间 | 默认为空 |
| duration | Varchar | 45 | 是 | 持续时间 | 默认为空 |

1. data\_mining\_result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| resultId | Varchar | 32 | 否 | 结果编号 | 主键 |
| type | Varchar | 20 | 否 | 结果种类 |  |

3、datasets\_assigned

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| taskId | Varchar | 32 | 否 | 任务编号 | 主键 |
| collectionId | Varchar | 32 | 否 | 数据集编号 | 主键 |

4、algorithm\_config

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| taskId | Varchar | 32 | 否 | 任务编号 | 主键 |
| algorithmId | Varchar | 32 | 否 | 算法编号 | 主键 |

5、task\_result\_re

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| taskId | Varchar | 32 | 否 | 任务编号 | 主键 |
| resultId | Varchar | 32 | 否 | 结果编号 | 主键 |

6、group\_task\_re

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| taskId | Varchar | 32 | 否 | 任务编号 | 主键 |
| groupId | Varchar | 32 | 否 | 分组编号 | 主键 |

### 学生管理

1、student

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| studentId | Varchar | 32 | 否 | 学号 | 主键 |
| studentName | Varchar | 20 | 否 | 学生姓名 |  |
| classId | int | 11 | 否 | 班级号 |  |
| profession | Varchar | 45 | 否 | 专业 |  |

2、group

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| groupId | Varchar | 32 | 否 | 组号 | 主键 |
| groupLeaderId | Varchar | 32 | 否 | 组长学号 |  |
| groupName | Varchar | 45 | 否 | 小组名 |  |
| setupDate | Date |  | 否 | 设立时间 | 默认为空 |

3、group\_compos\_re

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| studentId | int | 11 | 否 | 学号 | 主键 |
| groupId | int | 11 | 否 | 组号 | 主键 |

### 数据集管理

1、provider

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| id | Varchar | 32 | 否 | 提供者 | 主键 |
| name | Varchar | 20 | 否 | 提供者姓名 |  |
| description | Varchar | 45 | 否 | 描述 |  |
| email | Varchar | 45 | 否 | 邮箱 |  |

2、data\_set\_collec

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | 备注 |
| id | Varchar | 32 | 否 | 数据集合编号 | 主键 |
| characteristics | Varchar | 20 | 否 | 特征 |  |
| enableMissing | Varchar | 45 | 否 | 允许缺省 |  |
| description | Varchar | 45 | 否 | 描述 |  |
| dateDonated | Varchar | 45 | 否 | 捐赠时间 |  |
| relevantePapers | Varchar | 255 | 是 | 相关论文 |  |
| abstract | Varchar | 255 | 否 | 描述摘要 |  |
| associatedTasks | Varchar | 255 | 是 | 任务相关 |  |
| topic | Varchar | 100 | 是 | 数据主题 |  |
| area | Varchar | 20 | 是 | 来自地区 |  |

3、data\_set\_container

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | | 备注 | |
| id | Varchar | 32 | 否 | | 数据集编号 | 主键 |
| collectionId | Varchar | 32 | 否 | | 数据集合编号 | 外键 | |
| fileType | Varchar | 20 | 否 | | 文件类型 |  | |
| data | Varchar | 45 | 否 | | 数据 |  | |
| instances | Varchar | 45 | 否 | | 实例数 |  | |
| size | Varchar | 45 | 否 | | 大小 |  | |
| setName | Varchar | 255 | 是 | | 数据集名称 |  | |
| attributeType | Varchar | 255 | 否 | | 属性类型 |  | |

4、data\_set\_attribute

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 | 允许为空 | 中文说明 | | 备注 |
| id | Varchar | 32 | 否 | | 数据集编号 | 主键 |
| containerId | Varchar | 32 | 否 | | 数据容器编号 | 外键 |
| filed | Varchar | 32 | 否 | | 属性数据域 |  |
| description | Varchar | 20 | 否 | | 属性描述 |  |
| attributeFeature | Varchar | 45 | 否 | | 属性特征 |  |

# 模块设计

## 详细模块设计

### 登录注册模块

**1、在线注册**

在登录之前，未拥有账号的学生可以选择创建新账号，采用学号注册。注册时需要输入账号（学号）、姓名、密码和再次确认密码，提交后系统会对输入数据的有效性（例如输入项是否符合格式、两次密码是否一致、学号是否已存在）进行验证，若符合条件则注册成功，否则提示重新注册，返回注册界面。

**2、用户登录**

学生和管理员统一由此入口登录，学生输入学号和密码，管理员（即教师）输入工号和密码。输入的账号跟密码会验证是否合法。

输入登录所需要的账号密码后，提交登录信息，后台的权限管理框架会验证提交信息的有效性，返回给前台一个令牌。所有用户的访问需要携带给令牌访问系统，令牌存在有效期。任何对于系统背后的模块访问（即使url暴露），都需要验证用户授权信息，无权限的访问操作将被重定向到登录页面。则进入系统内部，否则提示出错信息，返回登录界面。

### 数据集管理模块

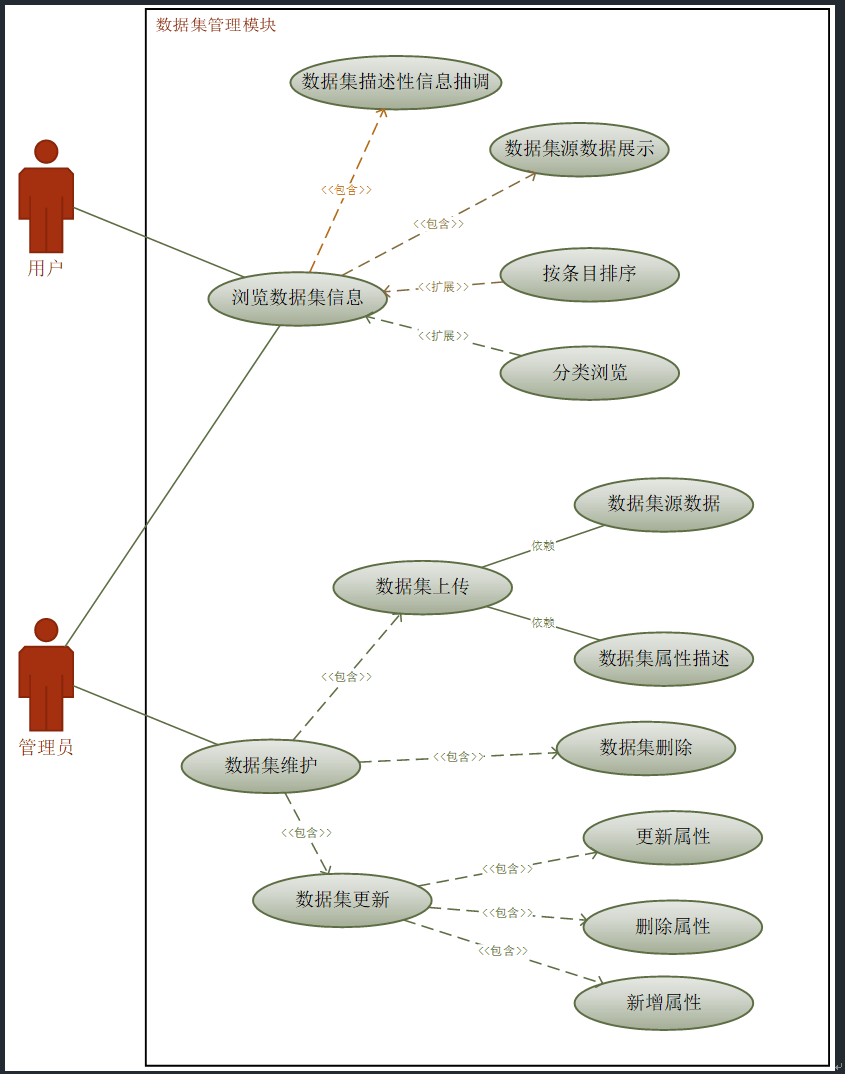
****

图 5‑1数据集管理模块用例图

**1、数据集概况展示**

该板块的首页是展示数据集的概况，学生和管理员都可通过从首页了解到该管理系统中所有数据集的数据类型、数据数量、数据挖掘形式等概括性信息，从总体上给用户直观的印象。

该板块支持数据集库根据不同的分类进行排序查看，即用户可以在表中点击不同的基本信息，数据集库就会根据内容的字母或数字等顺序，进行排序展示数据集库。

同时该板块也支持分类浏览数据集信息，用户可以通过点击分类框内的不同分类内容，查看所需要的类型的数据集。用户也可以通过表格查询关键字的方式直接对所需要的类型进行查询筛选。

**2、数据集具体展示**

数据集的具体内容是由首页的数据集库选择进入单个数据集后展示的，它是数据集内容的具体呈现，包含了数据集的基本描述信息以及具体内容，同时可以在此页面进行数据集的。

基本描述信息包括数据集的名称、年份、实例数、所属的类型（分类、聚类）等，数据集的具体内容通过表格的形式呈现，支持用户进行查询和排序的简单操作，并且能够按照不同的条目（如年份、类型）进行分类排序浏览。对于管理员而言，在该页面中还支持对数据集的基本信息及数据集内的具体属性进行增加、删除。

**3、数据集信息管理**

对于学生而言，对数据集可以使用的操作仅有下载，学生可以根据自己的任务所需数据集，到该数据集所在页面，下载数据集及其描述文件，进行数据挖掘的具体操作。

而对于管理员而言，对数据集的处理包括数据集的上传、删除、更新及下载。上传数据集是提供一个单独页面进行数据集的上传，以及对于数据集描述信息进行填写，由后台保存和读入。删除则是通过表单选择需要删除的数据集，直接对数据集进行删除，删除的同时会提示相应的删除信息，如已被关联的数据集删除警告、数据量大的评估警告。更新则是可以对数据集的基本信息，以及数据集内的具体属性进行增加删除，具体是在各个数据集页面中实现。

### 学生信息管理模块

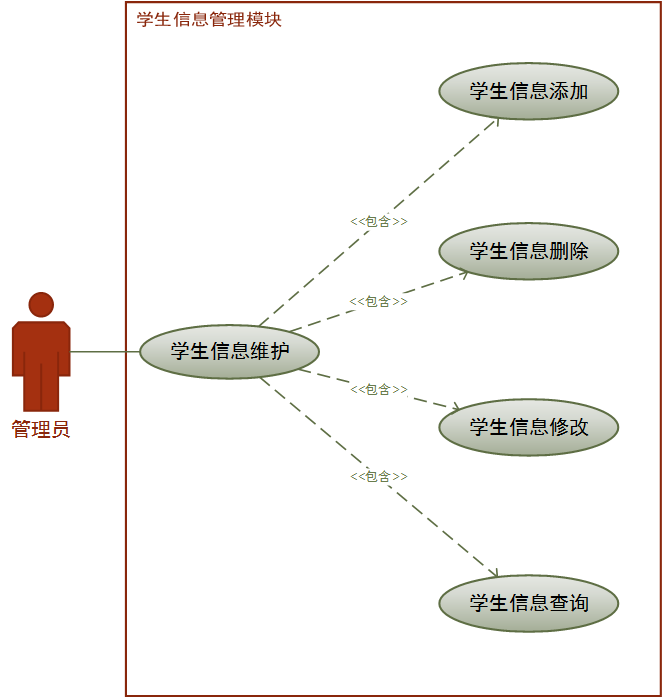


图5.1.3 学生信息管理模块用例图

**1、学生概况展示**

该模块主要是管理员对于学生信息的管理，其首页主要展示学生的信息概况，包括系统内已记录的学生的信息数、一天内新增学生数量、学生增加量趋势等一些数据信息。

**2、学生信息管理**

管理员（教师）可以对学生的信息进行添加、修改、删除和查询，具体的学生信息包括学生的姓名、学号、登录密码、班号、专业等，教师可以根据自己的需求进行修改。

### 实践任务管理模块

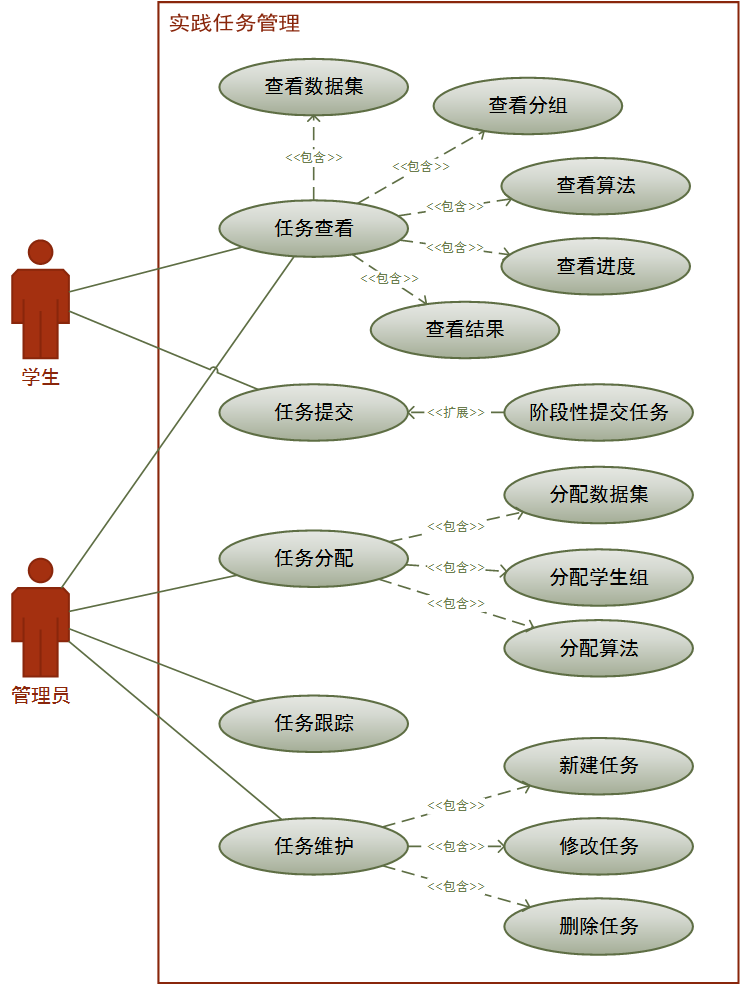


图5.1.4 实践任务管理模块用例图

**1、任务情况展示**

该模块主要是数据挖掘任务的管理和分配，对于学生而言，主要是任务分配后情况的展示，主要呈现的自己所执行的任务、使用的数据集、以及使用的一个或多个算法、执行进度。通过任务情况的展示，学生可以了解自己所执行的任务。

**2、提交执行任务**

学生可以根据自己所执行的任务的进度，进行阶段性提交总结，任务可以多次提交，查看具体结果，也可以对已提交的结果进行查看。

**3、后台任务分配**

系统会根据历史的任务状况自动筛选合适的学生，运行默认的算法配置好任务，分配数据集库内的数据集，并为任务合理指定几组不同的学生。

每组学生每次只能分配一组任务，一个任务可以由多组学生参与，同时也配置相对应的几个算法。

同时除了系统默认生成任务配置，可由管理员执行偏好配置，学生当前的任务状态可在系统表现出来，对于任务冲突等学生被分配任务给予操作提示。

**4、管理修改任务**

管理员可以根据任务分配的情况进行修改。包括新增挖掘任务、对任务所分配的数据集定向修改、自由调整学生的分组状态、任务算法的。

**5、任务跟踪分析**

系统可以记录下学生执行数据挖掘任务的进度，呈现学生任务的阶段。同时可以调用学生完成任务后得到的结果存储记录，比较结果算法、算法参数、实验方法等因素造成的差异。

### 数据挖掘算法库管理部分

这个模块主要是对数据挖掘算法的提供和展示，支持算法（分类、聚类等）的分类浏览，可以对具体算法进行阅读和浏览，浏览包括算法参数、类型、接口的综合描述。

本系统的算法在底层默认实现，暂不提供对外的算法导入，以便统一在本平台进行数据发掘任务的接口与参数。

# 项目开发

## 技术方案

### Sring Java Bean管理

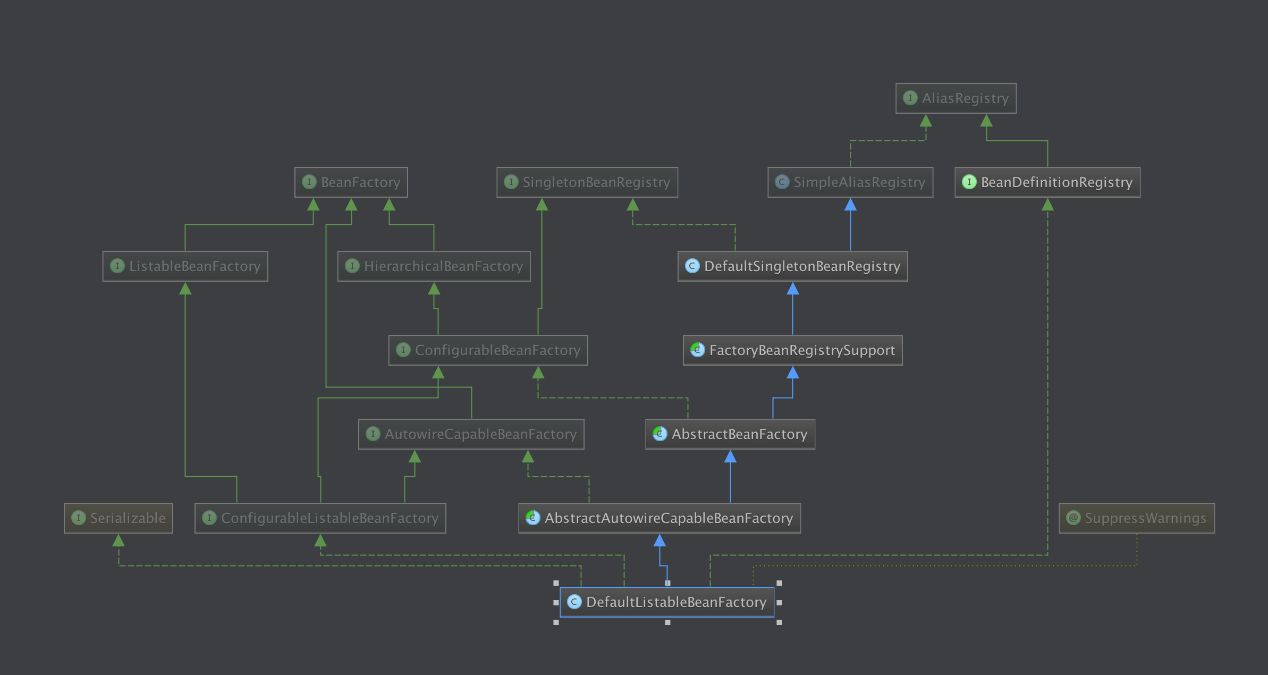


图 6‑1 Spring Bean 管理机制实现类图

利用Spring承载着工厂设计模式的BeanFactory，实现负bean配置文档的读取，管理bean的加载、实例化，维护bean之间的依赖关系，负责bean的生命周期。整个平台的Java Bean能在Spring的帮助下完成依赖注入和控制反转，使本系统更切合高内聚、低耦合的系统架构理念。

### Spring MVC支撑下的数据驱动技术

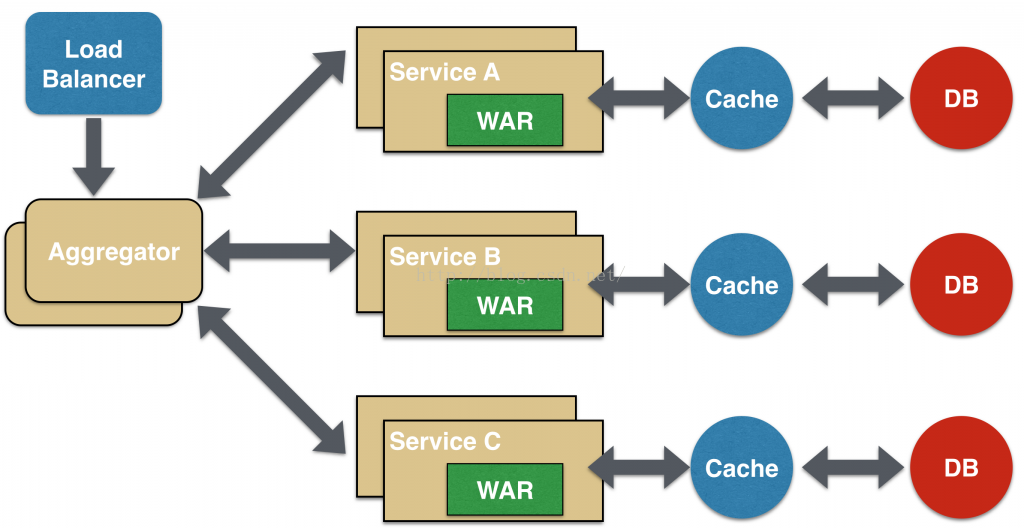


图 6‑2 集合器微服务设计模式

本系统后台运用Spring MVC+Hibernate+MySQL的框架组合，成功提供了完善和构造了Service 业务逻辑层、Controller控制层，DAO数据访问层，DTO数据传输层，后台通过Spring MVC实现了纯粹的Resful风格架构API，它结构清晰、符合标准、易于理解、扩展方便，具有高内聚低耦合的特点。控制层作为中间层将表现层页面的请求传到后台业务逻辑层。系统能从Client传来的Json序列通过ObjectMapper解析成对应的Model,也能根据相应的Model映射成对应Json序列,全自动而智能地快速实现数据转化与驱动，适应前端访问，也适配数据库的快速操作要求。

### Vue.js UI渐进式界面开发

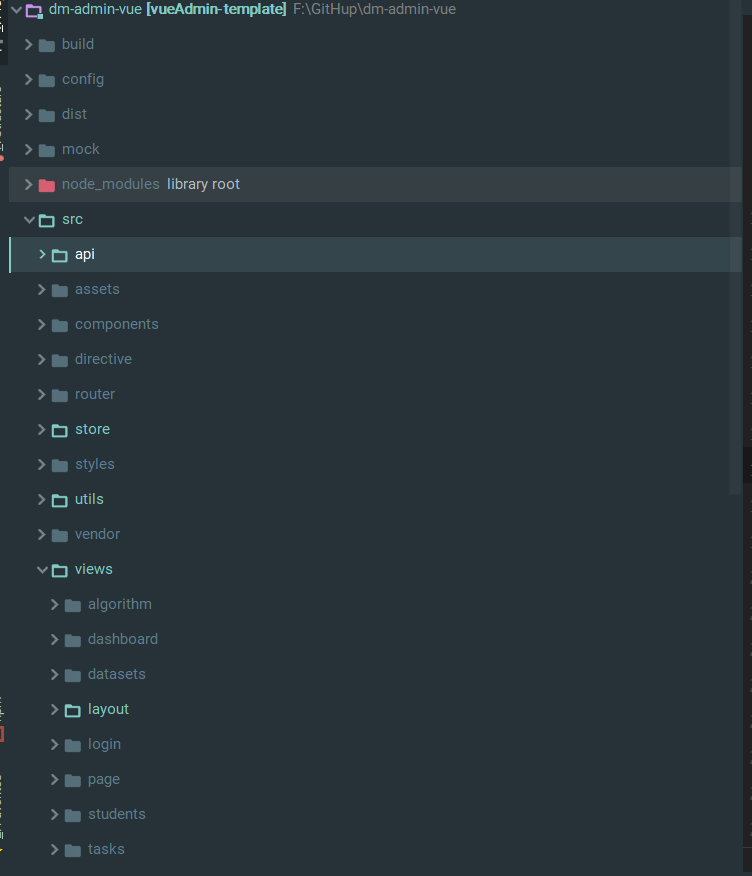


图 6‑3 前端开发工程结构图

本系统采用Vue.js打造了渐进式的用户界面。渐进式即根据系统的功能模块渲染特定的SPA（单页Web应用）格局，采用自底向上增量开发的设计。第一方面前端核心库只关注视图层，易于上手，且便于与第三方库或既有项目整合。另一方面，当与单文件组件和 Vue 生态系统支持的库结合使用时，Vue 也完全能够为复杂的单页应用程序提供驱动。最后凭借Vue优越的组件化设计理念，本系统能提供更多的“开发重用”的需求实现方案，前端在保持渐进式节奏的前提下，实现前端开发工程化。

### 前后端联调的分离模式

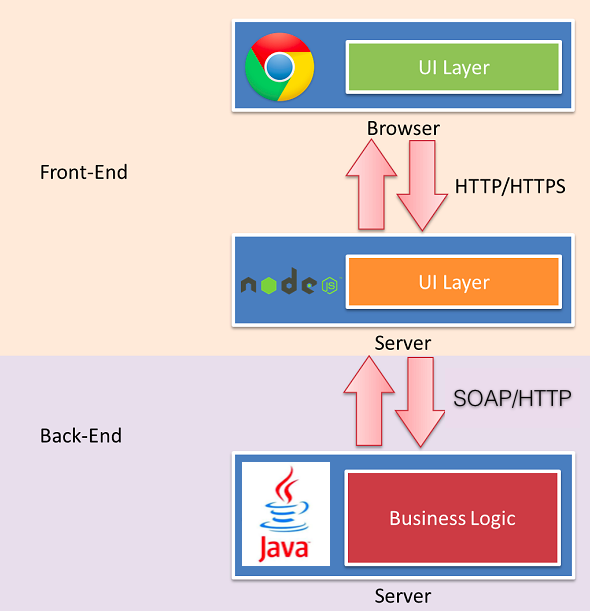


图 6‑4 Node.js下前后端分离模式

系统前端后端交互过程中，始终保持着HTTP+JSON的组合基调。后端专注于统一API的开发提供数据，同时维护系统架构的稳定，保证数据的安全。前端人员专注于前台使用Node.js、Vue.js等第三方开源库框架来打造轻便、简洁、流畅的表现层UI。

　　双方交互基于HTTP+JSON接口，后端人员基本只对接口负责，无需负责js和html的代码。前端人员只对界面展示交互负责，对于后端http接口如何提供正确的数据无需负责。当然前端成员可以随意的使用Mock等数据虚拟技术，完成界面的自模块测试；后端亦有针对地对具备复杂业务逻辑的API进行单元测试。

最后，在整合阶段中，表现层作为前端主要的内容，只负责页面的渲染与数据可视化加工。前端只通过aixos封装的ajax向后台发送GET、PUT、DELETE、FATCH等异步请求，与后端进行数据交互。而前端的部署的服务端口可以与后端非一致，通过跨域访问的技术，可以在遵循API文档约定的前提下，完成前后端分离、联调的协作任务。

## 关键模块的实现技术及数据结构等

### 访问层数据源的配置

@Bean(name = "dataSource")  
*public* DataSource loadDataSource()  
{  
 ComboPooledDataSource dataSource = *new* ComboPooledDataSource();  
 dataSource.setUser(environment.getProperty("jdbc.username"));  
 dataSource.setPassword(environment.getProperty("jdbc.password"));  
 dataSource.setJdbcUrl(environment.getProperty("jdbc.url"));  
 *try* {  
 dataSource.setDriverClass(environment.getProperty("jdbc.driver"));  
 }  
 *catch* (PropertyVetoException e)  
 {  
 e.printStackTrace();  
 }

图 6‑5 数据源配置的Java Config形式

LocalSessionFactoryBean sessionFactoryBean = *new* LocalSessionFactoryBean();  
sessionFactoryBean.setDataSource(dataSource);

图 6‑5 DataSource 被注入到Hibernate的Session Factory中

连接数据库的配置信息集成在dataSource.properties文件中，不再保持传统的.xml的配置方式，通过Java代码配合Spring创建、生成、管理数据源Bean。次数据源的Bean会被注入到Spring 托管下的Hibernate SessionFactory中去，系统此时已经完成了Spring Application Context下的LocalSessionFactoryBean的单例创建，此后DTO即可使用对应的Hibernate Session完成数据库访问、事务回滚等数据访问层的相关操作，支撑起系统的整个Service业务的运转。

### ORM映射关系的建立

在应用程序与数据库建立连接后，为每一个对象对应生成一个SQL数据表，本系统采用Hibernate框架进行数据库持久化操作，每个数据表和POJO类之间存在着一一对应的映射关系，即Model模板。

以任务DataMiningTask为例，表@Table(name="data\_mining\_task")对应的DataMiningTask.java如下：

@Entity  
@Table(name = "data\_mining\_task")  
*public class* DataMiningTask  
{  
 *private* String taskId;  
 *private* String taskName;  
 *private* String type;  
 *private* String taskDescription;  
 *private int* duration;  
 *private* String startTime;  
 *private* String finishTime;  
 *private* Set<DataMiningGroup> groups;  
 *private* Set<DataSetCollection> collections;  
 *private* Set<Algorithm> algorithms;  
 @Id  
 @GenericGenerator(name = "uuidGenerator", strategy = "uuid")  
 @GeneratedValue(generator = "uuidGenerator")  
 *public* String getTaskId()  
 {  
 *return* taskId;  
 }  
 *public void* setTaskId(String taskId)  
 {  
 *this*.taskId = taskId;  
 }

… … …

All base properties’ setters and getters

}

图 6‑6 DataMiningTask映射配置范例

由@Table、@Entity、@Id等可以看出本系统使用了典型的注解方式。所有业务代码压缩Xml的使用比率，统一使用新型简洁的Java Annotation注解技术进行编码；如在后台中所有框架（诸如Shiro、Spring mvc、Hibernate）的整合都摒弃了传统的Xml的繁杂配置，使用Java Config全注解配置的方式，优化了启动的配置解析过程，整合逻辑更为规范与整洁，开发更为集中与统一。

@ManyToMany(fetch = FetchType.EAGER)  
@JoinTable(name = "task\_algorithm\_rela", joinColumns = @JoinColumn(name = "taskId",referencedColumnName = "taskId")  
 , inverseJoinColumns = @JoinColumn(name = "algorithmId",referencedColumnName = "algorithmId"))  
*public* Set<Algorithm> getAlgorithms() {  
 *return* algorithms;  
}  
  
*public void* setAlgorithms(Set<Algorithm> algorithms)  
{  
 *this*.algorithms = algorithms;  
}

图 6‑7 关联关系的注解描述示例

以注解方式写好的一对多关系、多对多关系，可以减少大量冗杂的XML样板代码。

### DAO层的泛型的支持

本系统采用DAO设计模式，以面向接口的思想来编程，对于所有DAO的开发，都有之对应接口声明规范，每个方法体应该提供对应的java doc注释。所以在设计所有其他实体类之前，首先涉及了一个公共的DAO接口BaseRepository.java，同时通过BaseDao类来实现该接口，这个接口中定义了其他实体类都要用到的方法，如最基础的CRUD，模糊查找、分页、排序、聚合等方法。扩展DAO会统一继承该基类，获得访问数据库的基本API，提供代码重用率。同时具体类的DAO可实现与自身更密切相关的接口规范，为Service提供更复杂的数据库访问查询或操作。

*public interface* BaseRepository<E, PK *extends* Serializable>  
{CriteriaQuery<E> buildCriteriaQuery();  
 CriteriaDelete<E> buildCriteriaDelete();Serializable save(E entity) *throws* Exception;E findById(PK id); *int* saveBatch(Set<E> set);  
 *void* update(E entity);  
 *void* delete(E entity);List<E> get(Pageable pageable);

}

图 6‑8 基础DAO的接口约定

*public class* MiningTaskDao *extends* BaseDao<DataMiningTask, String> *implements* MiningTaskRepository {  
 *public* MiningTaskDao() {  
 *super*(DataMiningTask.*class*);  
 }  
 @Override  
 *public* List<DataMiningGroup> fetchInvolvedGroups(String taskId) {  
 String hqlString = "select t.groups from DataMiningTask t " +  
 "left join fetch t.groups g " +  
 "left join fetch g.groupMembers m " +  
 "where t.taskId = :taskId";  
 *return* getSession().createQuery(hqlString)  
 .setParameter("taskId",taskId)  
 .getResultList();  
 }

}

图 6‑9 继承基类并实现了额外接口规范的DAO

### Controller层的业务委托

Controller层定义并暴露面向前端的所有API，但Controller层并没有承载复杂业务逻辑，Restfult架构下的Controller只顾适配、调用逻辑函数，将所有业务逻辑委托给Service层。至于Service层如何实现其要求的逻辑，如何桥接DAO定义事务边界，如何组装并返回访问的数据，Controller并不知道。

@Service  
*public class* UserServiceImpl *extends* AbstractBaseServiceImpl<User, String> *implements* UserService  
{  
 *private* StatelessCredentialsService credentialsService;@Autowired  
 *public void* setCredentialsService(StatelessCredentialsService credentialsService) {  
 *this*.credentialsService = credentialsService;  
 }  
 *public* User registerUser(User user)  
 {  
 ……//实现了复杂的用户注册逻辑  
 }

图 6‑10 该服务类实现了用户注册（账户创建）的业务

以UserService为例，其实现了用户注册、登录、获取信息等基本业务，并提供了相应的方法供需求类调用，即可完成对应的业务。

@RestController  
@RequestMapping(value = "/user")  
*public class* UserController  
{  
 *private* UserService userService;  
 @RequestMapping(method = RequestMethod.POST)  
 *public* ResponseEntity<UserDTO> register(@RequestBody User user)  
 {  
 *return new* ResponseEntity<UserDTO>(userService.registerUser(user), HttpStatus.CREATED);  
 }

}

图 6‑11 Restful API的示例

如上，Controller只需要调用其被Spring注入的UserService的相应registerUser即完成了用户注册、账户创建功能，系统就能响应前端表现层的用户注册请求。

### DTO传输的暴露策略

表现层与业务逻辑层之间是通过数据传输对象（DTO）进行交互的，数据传输对象是没有行为的POCO对象，它的作用只是为了对领域对象进行数据封装，实现层与层之间的数据传递，它本身并不是业务对象。数据传输对象是根据UI的需求进行设计的，而不是根据领域对象进行设计的。例如Student作为领域对象User的子类，包含一些诸如密码、盐值等敏感的授权信息，系统运用DTO能根据设计需求隐藏这些字段，并能获取到前端要求作为UI渲染驱动数据。通过DTO实现表现层与Model之间的解耦，表现层无需引用Model，后续开发过程中我们的模型即使改变，而UI没更改，我们就只需要修改Model而不需要去修改表现层代码。

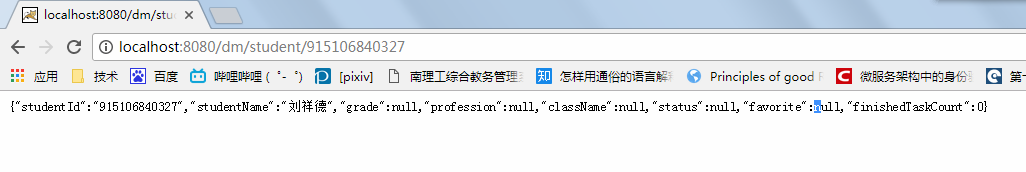


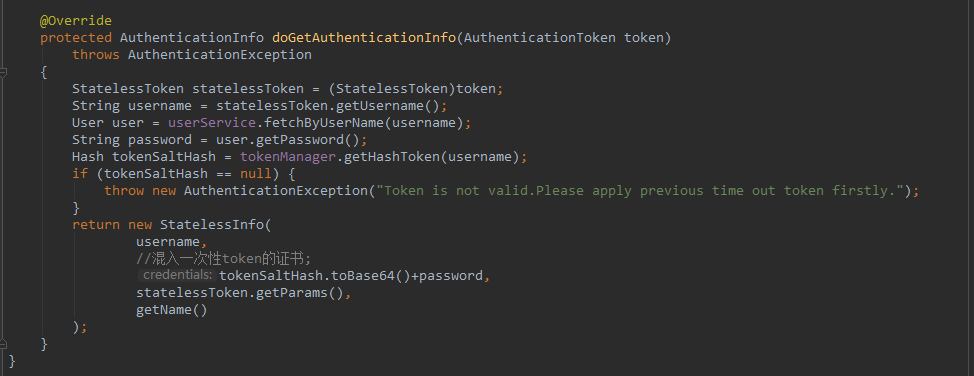
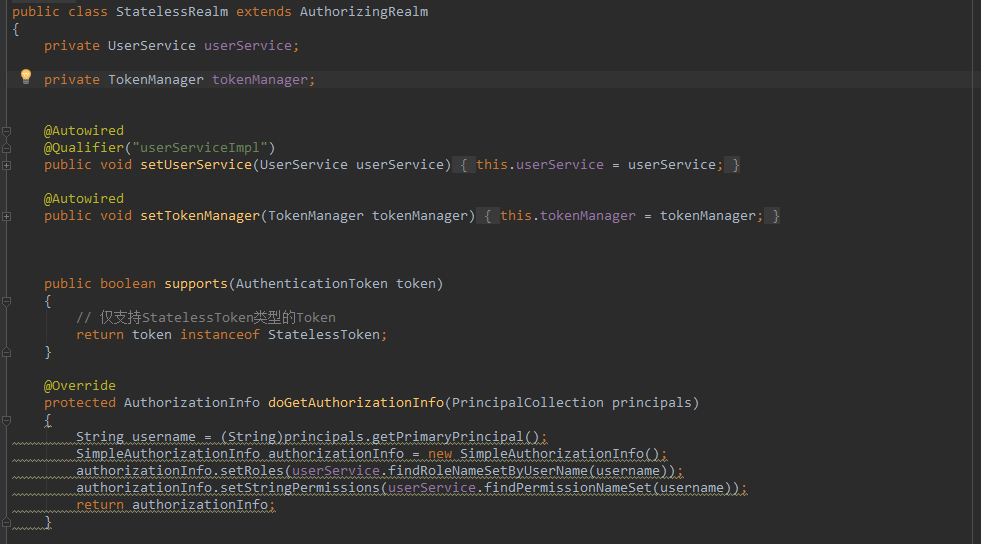
图 6‑12 studentDTO向前端暴露的字段

## 核心功能实现

### 登录授权模块

在传统的单体架构中，单个服务保存所有的用户数据，可以校验用户，并在认证成功后创建HTTP会话。在微服务架构中，用户是在和服务集合交互，每一个用户都有可能需要知道请求的用户是谁。本系统采用微服务的架构方案，服务系统完全是无状态的，不会执行像会话这种机制，所有的资源都可以通过URL（API）定位。每次客户端在提交相应的HTTP请求之后，服务器会根据协商约定重新计算并核对消息摘要，中途被拦截修改过的请求会被拒绝。

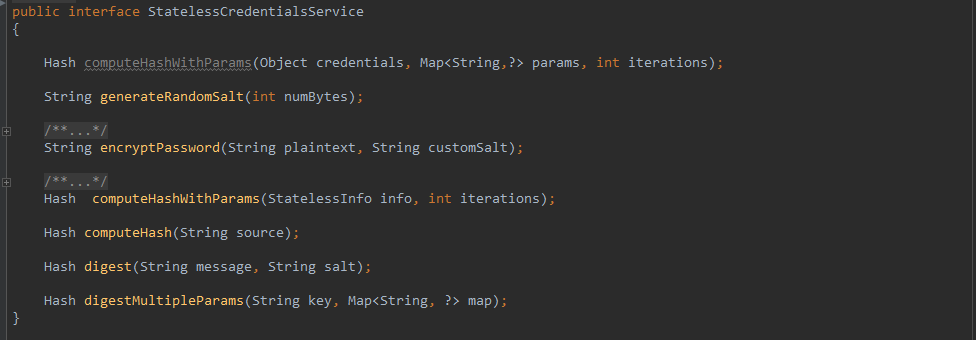
本系统在Shiro安全维护的框架基础上，针对每个请求帧，严格执行用户名的合法性验证机制，计算时间戳的差值，防止Token被截取下的重放攻击。

以StatelessRealm.java为例: 

代码目录：src\main\java\com.dm.org\security\realm\StatelessRealm.java

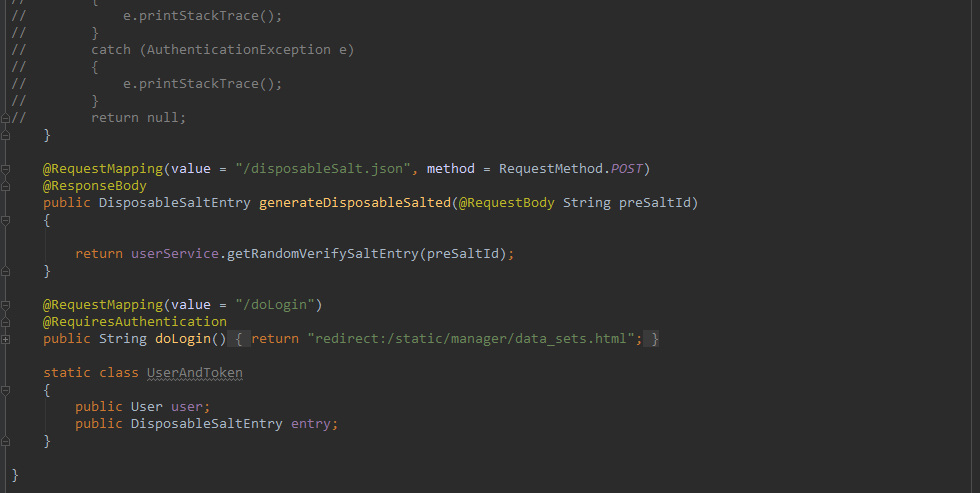
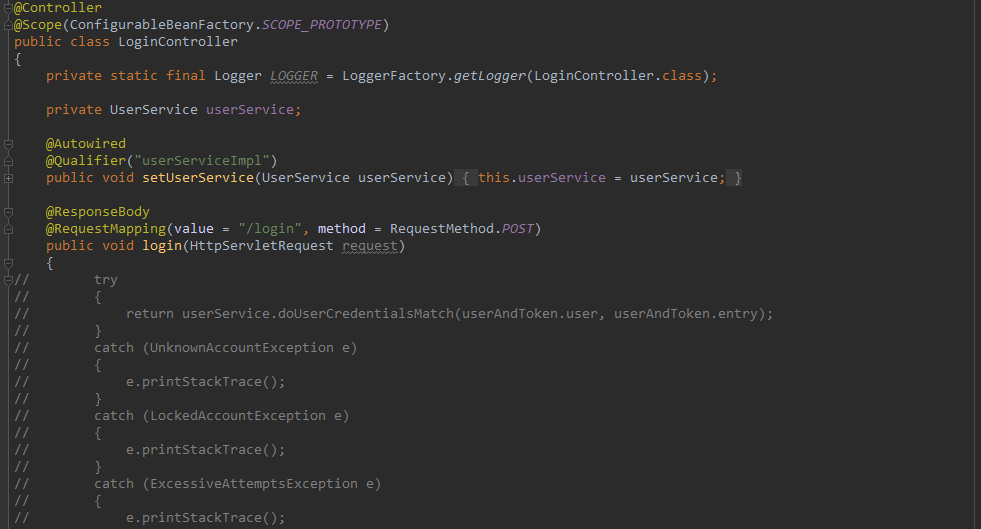
由于微服务系统中每个服务都要处理安全问题，可以使用如REST+HMAC认证的原理进行访问。HMAC（Hash-based Message Authentication Code）：基于散列的消息认证码，使用一个密钥和一个消息作为输入，生成它们的消息摘要。该密钥只有客户端和服务端知道，其他第三方是不知道的。访问时使用该消息摘要进行传播，服务端然后对该消息摘要进行验证。如果只传递用户名+密码的消息摘要，一旦被别人捕获可能会重复使用该摘要进行认证，使密码不“密”。

依此原理这里使用了token客户端令牌方案,此令牌在客户端生成，由身份验证服务进行签名，并且必须包含足够的信息，以便可以在所有微服务中建立用户身份。令牌会附加到每一个请求上，为微服务提供身份验证。



代码目录：src\main\java\com.dm.org\service\StatelessCredentialsService.java

学生和管理员统一由一个入口登录，学生输入学号和密码，管理员（即教师）输入工号和密码。输入的账号跟密码会验证是否合法。输入登录所需要的账号密码后，提交登录信息，后台的权限管理框架会验证提交信息的有效性，返回给前台一个令牌。所有用户的访问需要携带给令牌访问系统，令牌存在有效期。任何对于系统背后的模块访问（即使url暴露），都需要验证用户授权信息，无权限的访问操作将被重新定向到登录页面。则进入系统内部，否则提示出错信息，返回登录界面。



代码目录：src\main\java\com.dm.org\controller\LoginController.java

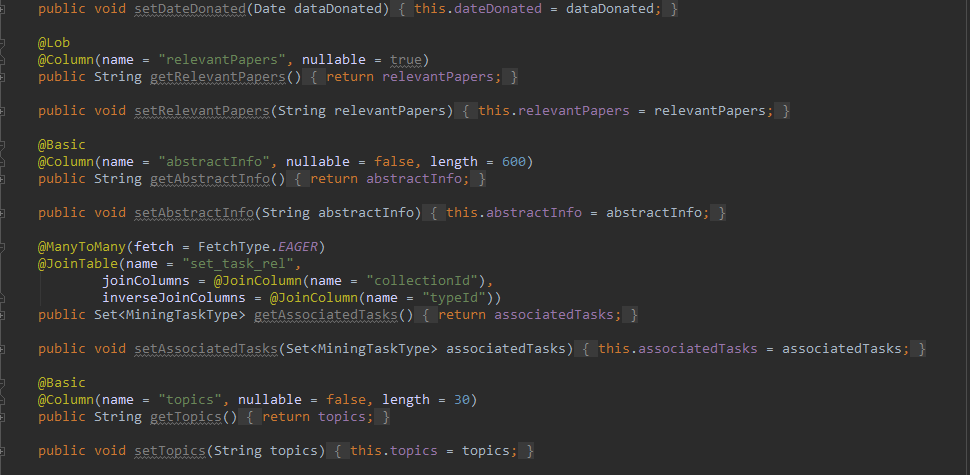
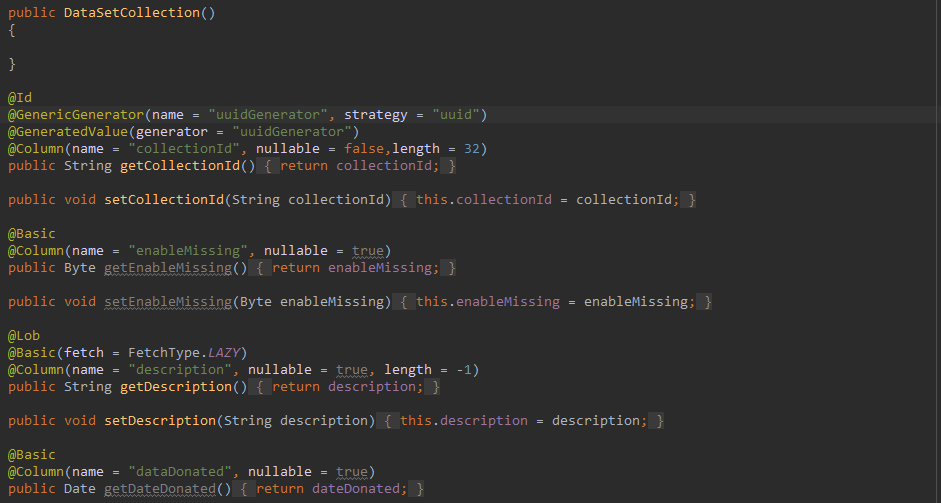
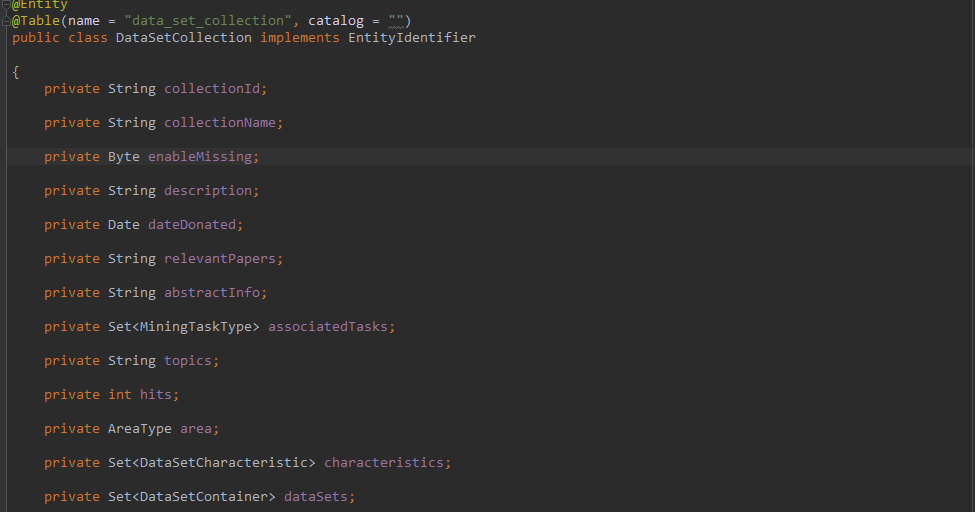
【登录界面】

### 数据集的上传与下载

数据集的管理

1、数据集概况展示

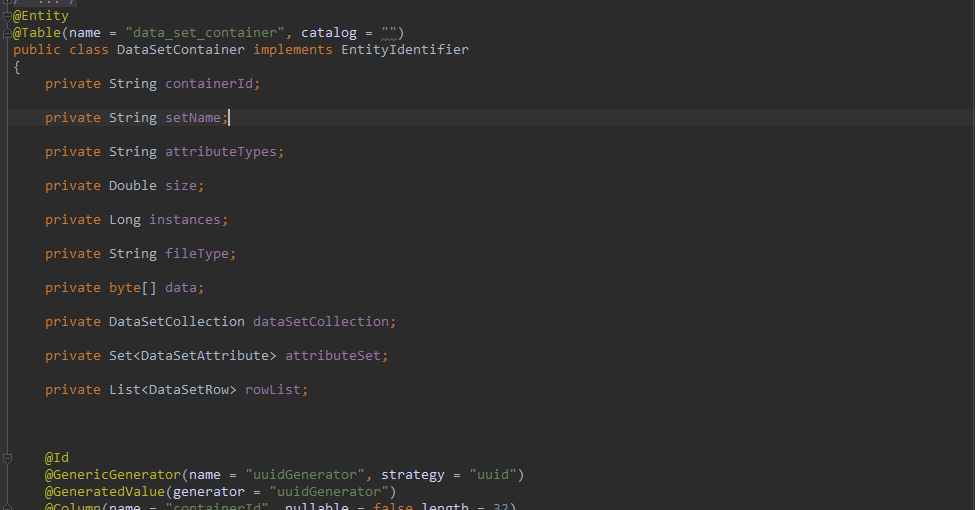
该板块的首页是展示数据集的概况，学生可通过从首页了解到该管理系统中所有数据集的数据类型、数据数量、数据挖掘形式等概括性信息，从总体上给用户直观的印象。该板块支持数据集库根据不同的分类进行排序查看，即用户可以在表中点击不同的基本信息，数据集库就会根据内容的字母或数字等顺序，进行排序展示数据集库。同时该板块也支持分类浏览数据集信息，用户可以通过点击分类框内的不同分类内容，查看所需要的类型的数据集。用户也可以通过表格查询关键字的方式直接对所需要的类型进行查询筛选。



代码目录：src\main\java\com.dm.org\model\DataSetCollection.java

2、数据集具体展示

数据集的具体内容是由首页的数据集库选择进入单个数据集后展示的，它是数据集内容的具体呈现，包含了数据集的基本描述信息以及具体内容，同时可以在此页面进行数据集的。基本描述信息包括数据集的名称、年份、实例数、所属的类型（分类、聚类）等，数据集的具体内容通过表格的形式呈现，支持用户进行查询和排序的简单操作，并且能够按照不同的条目（如年份、类型）进行分类排序浏览。对于管理员而言，在该页面中还支持对数据集的基本信息及数据集内的具体属性进行增加、删除。



代码目录：src\main\java\com.dm.org\model\DataSetContainer.java

3、数据集信息管理

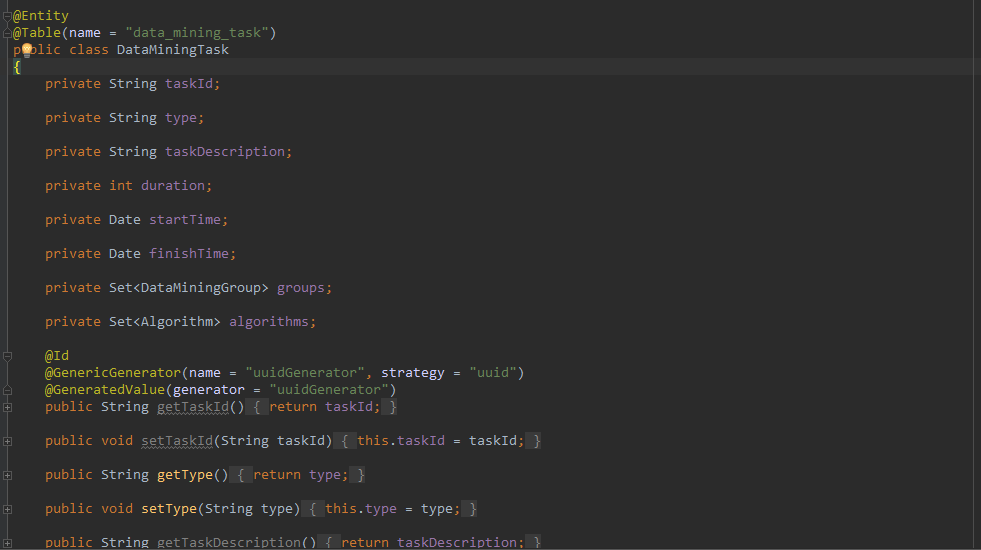
对于学生而言，对数据集可以使用的操作仅有下载，学生可以根据自己的任务所需数据集，到该数据集所在页面，下载数据集及其描述文件，进行数据挖掘的具体操作。

### 数据挖掘任务创建与分配

### 算法库管理集

1、任务情况展示

这个模块对于学生而言，主要是任务分配后情况的展示，主要呈现的自己所执行的任务、使用的数据集、以及使用的一个或多个算法、执行进度。通过任务情况的展示，学生可以了解自己所执行的任务。



代码目录：src\main\java\com.dm.org\model\DataMiningTask.java

2、提交执行任务

学生可以根据自己所执行的任务的进度，进行阶段性提交总结，任务可以多次提交，查看具体结果，也可以对已提交的结果进行查看。

【数据集界面】

**3.教师管理端**

**1）数据集管理**

1、数据集概况展示

管理员都可通过从首页了解到该管理系统中所有数据集的数据类型、数据数量、数据挖掘形式等概括性信息，从总体上给用户直观的印象。该板块支持数据集库根据不同的分类进行排序查看，即用户可以在表中点击不同的基本信息，数据集库就会根据内容的字母或数字等顺序，进行排序展示数据集库。

【】

2、数据集具体展示

同时该板块也支持分类浏览数据集信息，用户可以通过点击分类框内的不同分类内容，查看所需要的类型的数据集。用户也可以通过表格查询关键字的方式直接对所需要的类型进行查询筛选。

【】

3、数据集信息管理

而对于管理员而言，对数据集的处理包括数据集的上传、删除、更新及下载。上传数据集是提供一个单独页面进行数据集的上传，以及对于数据集描述信息进行填写，由后台保存和读入。删除则是通过表单选择需要删除的数据集，直接对数据集进行删除，删除的同时会提示相应的删除信息，如已被关联的数据集删除警告、数据量大的评估警告。更新则是可以对数据集的基本信息，以及数据集内的具体属性进行增加删除，具体是在各个数据集页面中实现。

【】

【数据集界面】

**2）算法库管理**

【算法库界面】

**3）学生信息管理**

1、学生概况展示

该模块主要是管理员对于学生信息的管理，其首页主要展示学生的信息概况，包括系统内已记录的学生的信息数、一天内新增学生数量、学生增加量趋势等一些数据信息。

【】

2、学生信息管理

管理员（教师）可以对学生的信息进行添加、修改、删除和查询，具体的学生信息包括学生的姓名、学号、登录密码、班号、专业等，教师可以根据自己的需求进行修改。

【】

【学生信息界面】

**4）实践任务管理**

1、任务查看

该模块主要是数据挖掘任务的管理和分配，对于学生而言，主要是任务分配后情况的展示，主要呈现的自己所执行的任务、使用的数据集、以及使用的一个或多个算法、执行进度。通过任务情况的展示，学生可以了解自己所执行的任务。

【】

2、任务分配

系统会根据历史的任务状况自动筛选合适的学生，运行默认的算法配置好任务，分配数据集库内的数据集，并为任务合理指定几组不同的学生。

每组学生每次只能分配一组任务，一个任务可以由多组学生参与，同时也配置相对应的几个算法。

同时除了系统默认生成任务配置，可由管理员执行偏好配置，学生当前的任务状态可在系统表现出来，对于任务冲突等学生被分配任务给予操作提示。

【】

3、任务维护

管理员可以根据任务分配的情况进行修改。包括新增挖掘任务、对任务所分配的数据集定向修改、自由调整学生的分组状态、任务算法的。

【】

4、任务跟踪

系统可以记录下学生执行数据挖掘任务的进度，呈现学生任务的阶段。同时可以调用学生完成任务后得到的结果存储记录，比较结果算法、算法参数、实验方法等因素造成的差异。

【】

【任务管理界面】

## 特殊问题及解决办法

### 系统整合框架版本兼容冲突

**问题描述：**在系统框架整合过程中，Hibernate4.9.1报出Invocation of init method failed; nested exception is java.lang.VerifyError，即Hibernate与现存的框架版本不符。

**解决方案：**更换Hibernate版本到5.2.1，同时Spring版本降低到4.3.9。

**解决结果：**war包成功部署到服务器启动。

### Hibernate的Session关闭

**问题描述：**一个Service组件调用的另一个Service组件中的方法返回一个代理对象时，使用对象获取懒加载数据出现No Session问题。

**解决方案：**重新调整Spring的声明式事务边界，使用其他实体的DAO作为数据组装驱动器，在方法体内可启用多个事务，但对可形成外一个整体事务。

**解决结果：**懒加载的对象成功被Hibernate发出的查询语句加载，且Service每个方法可以自由完成数据组装的任务。

### SQL表产生了多余字段

**问题描述：**创建一个新的实践任务时，任务种类、算法等类型属于多选值。前台通过勾选对应的多选值映射到对应的Entity使用Hibernate插入到数据库中，此时的Entity属于游离态非持久态，数据表产生了重复冗余的字段。

**解决方案：**通过传入的Id 号查询一次数据库，产生具备持久态的实体，再通过Setter给要进行持久化的实践任务赋值，最后执行保存操作。

**解决结果：**实践任务成功保存到数据库，多选值的表中不再产生多余字段。

### API鉴权消息摘要不一致

**问题描述：**前端通过js代码执行了协商好的HMAC SHA256哈希加密算法， 并对所有请求参数生成消息摘要，服务器再次进行哈希正向运算得到的消息摘要与客户端生成的不一致；

**解决方案：**检验加密过程的每一步运算，包括公盐混合哈希运算、时间戳混合私盐加密运算、请求参数混合消息摘要运算，每个步骤前后端都要需要严格统一。

**解决结果：**每个API成功被安全层拦截执行鉴权操作，具备权限的用户成功通过授权，得到期望数据，执行的相应业务逻辑。

### Options请求造成的403状态的跨域请求问题

**问题描述：**前端通过axios发送异步跨域请求时，会先发送请求头为OPTIONS的请求，试探服务器对请求的支持性，而Spring MVC默认对OPTIONS请求返回403状态码。

**解决方案：**添加一个访问过程拦截器AcessProcessInterceptor，对OPTIONS的请求进行状态码200的置换操作，且给对应的Response加入前端自定义Headers，让前端识别到服务器对自己的支持。

**解决结果：**OPTIONS试探请求收到状态码为200的Response，并成功得到了请求的数据。

### 文件上传中的请求格式与文件写入出错

**问题描述：**前端将数据集文件写入到MULTIPART\_FORM\_DATA的请求负载中，但Spring MVC 的Multipart 并没有得到负载中的文件，形成了一个空指针给数据持久层。

**解决方案：**给Spring MVC的注解配置中指定该API consumes为MULTIPART\_FORM\_DATA。

**解决结果：**SpringMVC成功解析到负载体中文件数据，保存到相应的目录。

### 页面快速切换情况下的API并发访问授权错误

**问题描述：**系统以用户名为键，对值放入带时间戳的私盐Token,一旦SPA发出了数个异步的并发Htpp请求，由于没有对缓存加锁，用于验证的Token就会产生脏读、不可重复读问题。

**解决方案：**借鉴线程安全实现机理，对于每次API的TOKEN申请，用UUID生成与对应的API\_KEY，即以（API\_KEY,TOKEN）键值词典保持请求与请求之间的独立性。

**解决结果：**成功解决了小剂量的并发访问的请求下的无状态验证。

### 列表分页控件崩溃

**问题描述：**分页查询封装体Query的Page参数从1起，后台永远查出第一页的内容，导致前端2页之后的内容缺失，发生崩溃。

**解决方案：**在后台的分页Page查询参数修正为0。

**解决结果：**成功实现了分页查询

### 浅拷贝使请求参数增加额外字段

**问题描述：**Js复制产生的是浅拷贝的值引用类型，在SPA页面中的查询参数一旦被重复出使用多次，Query后续被传入API进行类似的复制操作，就产生了额外的字段。

**解决方案：**在开始传入Query的请求参数，进行一次深拷贝或重置操作。

**解决结果：**维持了Query请求体参数一致性。

### Vue计算属性与Prop属性切换错误

**问题描述：**可重用组件的数据集概要信息页中Prop属性没有被赋值，页面成功渲染出DOM。

**解决方案：**在渲染之前判断一下Prop属性是否传入，最后执行对应渲染操作。

**解决结果：**在无数据的情况下会渲染出数据为空的提示，有数据的情况下会渲染出具体的信息。

### 三级联动勾选文件的列表显示出错

**问题描述：**多选分类下拉框的Label是根据后端拉取出的值生成的，Vue插件的语法要求每个OPTION有对应的Children键值。在产生计算属性时，由于忽略了Children的键值书写，三级联动分类出现了空白项，抛出异常。

**解决方案：**在计算属性dataCollections中加入语法要求的键值对。

**解决结果：**可以根据分类情况勾选对应的数据集，关联到对应的数据挖掘任务。

### 循环渲染组件或表单元素出错

**问题描述：**v-for是基于源数据多次渲染元素或模板块，为每项提供一个唯一 key 属性，以便它能跟踪每个节点的身份，从而重用和重新排序现有元素。因为组件有自己独立的作用域，所以任何数据都不会被自动传递到组件里。在这种情况下，不把 props把迭代数据传递到组件里，会出错。

**解决方案：**使用v-for动态渲染组件及表单元素，使得元素或模板块根据绑定的源数据。

### 提取后台数据方法的API书写问题

**问题描述：**前端发送请求时跟后台的URL映射不一致，导致404状态码。

**解决方法：**利用后台定义好的API向后台发起异步的post、delete、get、put请求，通过与后台相匹配，从而达成获取或者传输数据的目的。

### 导航栏的路由生成错误

**问题描述：**前后端分离下的页面跳转不与后台耦合，因为借助VUE-ROUTER组件的路由规则生成导航栏时，出现渲染错误。

**解决方法：**使用VUE-ROUTER进行动态路由匹配，把某种模式匹配到的所有路由，全都映射到同个组件。用<router-view>渲染最高级路由匹配到的组件，在 VueRouter的参数中使用children配置，实现嵌套的出口中渲染组件。通过children配置，嵌套递归多层路由，同时加入 <router-link>创建 a 标签来定义导航链接，实现页面间的跳转。

**解决结果：**生成可以一级、二级、三级、……选择的可视化导航栏，

# 个人体会

## 成员1 刘祥德的个人小结

从初步涉猎J2EE的规范下的Servlet、Jsp、Filter等Web容器服务组件，再到css、html、js前端基础的学习，最后再深入到Spring、SpringMVC、Hibernate、Shiro、Vue.js、Node.js等等WEB应用中比较著名的框架的学习，我逐渐具备了对整个系统的构思和实现能力。在本课程设计中，我负责对整个系统的架构设计、起草应用的技术方案、实现Restful API体系、完成复杂的数据库访问编程。

在协调前后端人员的开发过程中，我必须同时掌握后台开发技术的同时，还要进行前端代码的编程，以进行在前后端分离情况下前端业务逻辑的处理。前端队员无法解决的bug，我能给出解决方案排除，指导前端队友进行符合功能需求的UI设计。在后台没有完成相应的API开发时，我利用Easy Mock给前端提供了虚拟数据。

在后台API开发阶段，我运用HttpClient、Junit 、Mokito模拟向服务器的请求操作，完成了无服务认证、数据集上传与保存、实践任务分配等核心功能的单元测试。当然，也模拟了Web集成环境下的Controller核心API的微测试与调整。我亦协作了后端队友对于Entity模型的建立，成功构建了系统的数据库表。

身为本次课程设计的组长，我的角色类似与全栈式工程师（目前当然还是不敢担当），充当前后端沟通的桥梁，在后端系统整合、前端UI开发再到前后端联调的所有过程中，经历了太多太多的艰辛，每次启动一个功能出现相应的Bug的时候，我必须敏锐察觉到是界面中可能存在的代码错误，利用Debug技能疏通一个又一个功能。遇到的所有的问题太多，在这不可能一一列举（可参见6.4）。但是在整个系统逐渐成型时，我开始逐渐领悟到软件工程的规范与开发过程。通过Github进行的项目管理，我可以一步步整合队友贡献的代码，凝聚队友的所有逻辑思维与创作精髓，并且总结出系统开发的一系列关键问题防微杜渐。

回到整个系统大局思考，在设计伊始，我是存在一个疑问的，我们明明可以用简单的J2EE的组件可以做出拥有了相同功能的系统，却偏偏选择那些重量级的、轻量级的集成框架，我们是绕着弯路得不偿失吗？直到最后用微服务架构去设计重构整个系统时，我终于意识到系统是增量的。选择优良的架构原则、设计风格，的确让学习路线变得曲折不已。可当系统的功能不断拓展，优良的设计便发挥其巨大的优势了。相对其他课题的同学，我们的确慢了一点，但我知道，我们已经站在一个更高的起点。我们拥有的系统架构，可以快速创造更简洁美观的交互、更高内聚低耦合的核心，遵循更规范的工程开发流程，这就是我个在本次课设的一点小感受。

## 成员2 刘鑫容的个人小结

在这次的课程设计中，我负责大部分前端界面风格的设计和以及部分与后台交互的功能的实现。由于前端采用的是vue.js的新型渐进式框架搭配后台系统UI，与之前html+css+js的模式和写法都不太相同，因此前期花了一段时间进行vue.js的学习与配置。在这个过程中，我需要抛弃之前html冗长而复杂的写法，学习怎么进行组件的编写，怎么利用组件制造想要的效果和怎么绑定值和后台交互的相关操作。

在实际应用中，前端采用的是vue-cli+webpack的构架，页面的编写是利用单组件文件的形式，看起来更加清晰简洁。由于前期学习的是基础知识，在实际操作中，我还学习了vue-router的动态路由、嵌套路由等如何实现页面跳转，如何利用axios向后台发起异步的post、delete、get、put请求，从而使前端的表单、表格可以获取到后台的数据。

这个过程中，我也曾遇到因为UI使用问题而导致页面无法显示的问题，也因为v-for动态渲染组件及表单元素等的正确方式而多次尝试。根据组长所编写的api及与后台传送数据的代码，我基本实现了数据集、任务分配两个模块的功能，但是如果要实现更多功能，使界面更加友好，使用户满意，还需要继续学习相关的知识。

这次的课程设计让我在原先的网页基础上又提升了一大步，接触了企业级别的工程实现形式，在接下去的时间我会继续学习，将前端的界面功能做得更好更完善。

7.3马腾的个人小结