**6.项目开发**

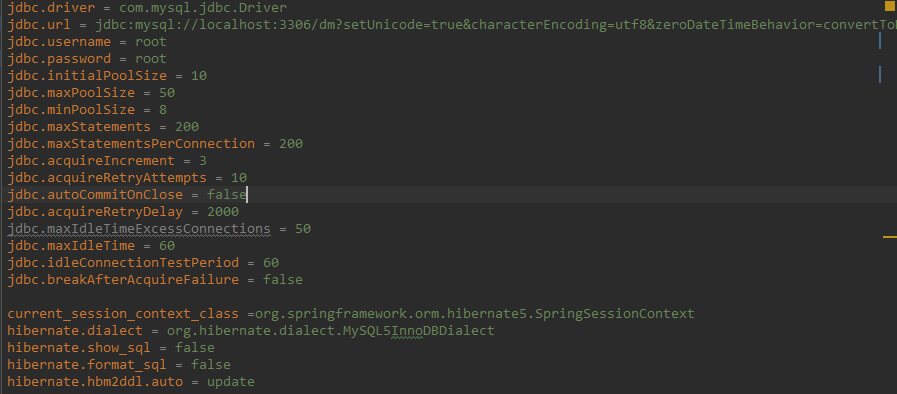
**6.1 技术方案**

本平台后台运用Struts2+Hibernate+Spring MVC技术组合实现，前后端完全分离，采用Resful API架构，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，使用Spring MVC设计模式。从Server传来的JSON“名称/值”对通过model接口读取传递给控制器controller，controller根据请求头调用业务逻辑层对应接口service，如DataSetService、StudentService等等，进而由DAO层实现对数据库表的操作。Hibernate主要用在系统中间持久化层，充当数据访问组件，并加入了DTO设计模式，减少了请求次数，一定程度上提高了数据访问安全。由于Spring MVC对Hibernate具有良好的支持，由Spring容器完成DAO类的实例化操作,并采用spring container管理所有系统的bean,将bean的生命周期托付给spring。同时在数据访问层加入REST架构的主要约束Spring HATEOAS。

前台使用html、css、node.js、vue.js、jquery框架打造轻便、简洁、流畅的表现层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。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。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。

**6.2关键模块的实现技术及数据结构等**

**6.2.1配置数据源**

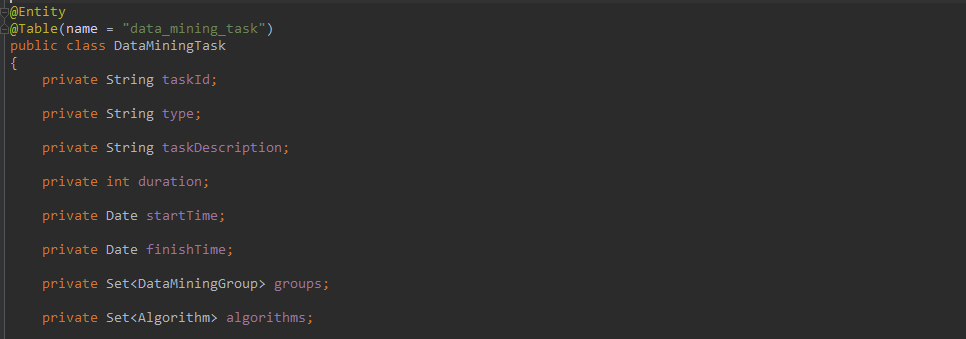
****

连接数据库的配置信息保存在src\main\resources\jdbc.properties文件中，由Spring负责创建及管理数据源，配置数据源的代码如上。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。另采用支持Hibernate的C3p0连接池。

**6.2.2配置映射关系**

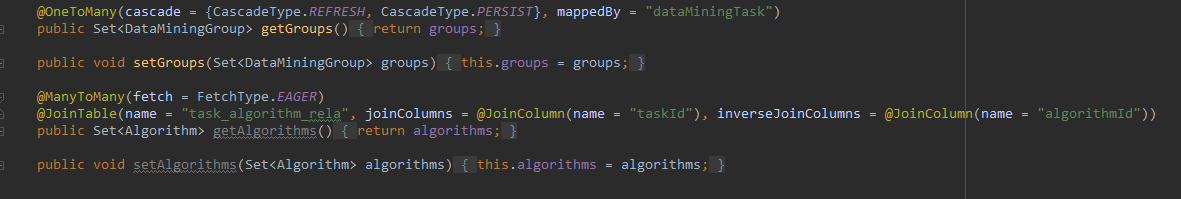
在应用程序与数据库建立连接后，为每一个对象对应生成SQL表单，由于本系统运用Hibernate框架进行数据库持久化操作，因此还需要为每个数据表和POJO类之间配置映射关系，即model模板。

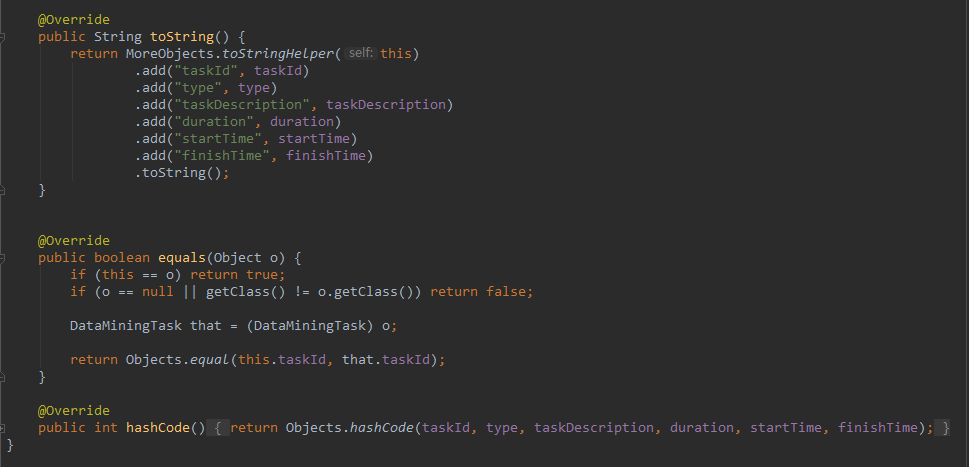
以任务DataMiningTask为例，表@Table(name="data\_mining\_task")对应的DataMiningTask.java如下：





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的繁杂分离配置，使用java config全注解代码的优化系统的启动与维护，故后台中并没有映射文件DataMiningTask.xml的存在。



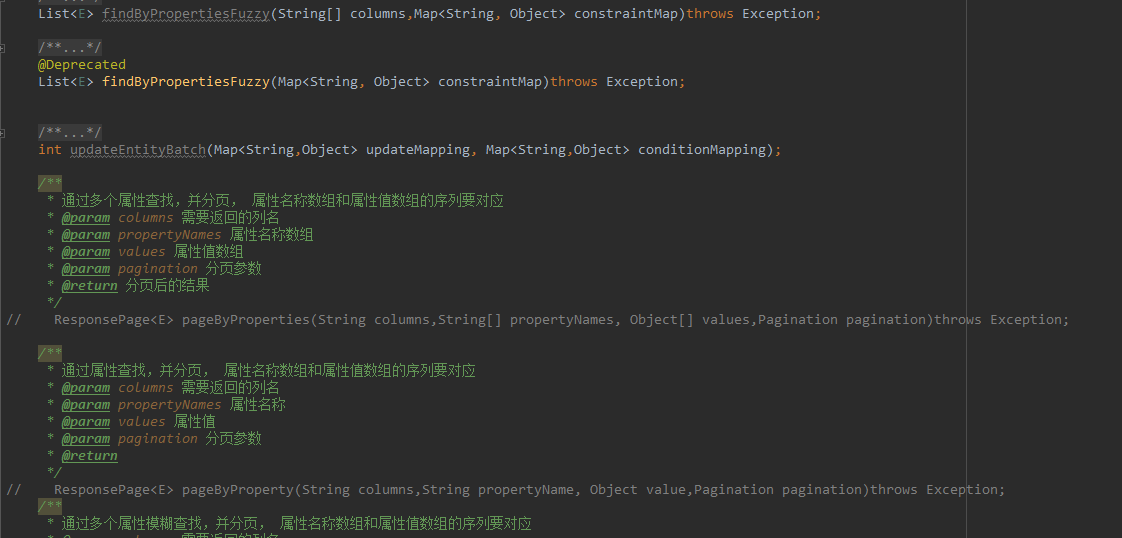
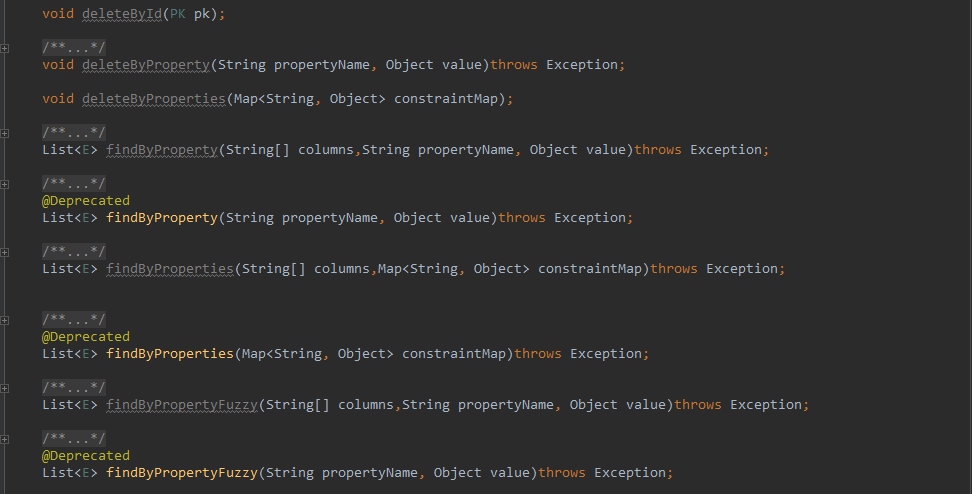
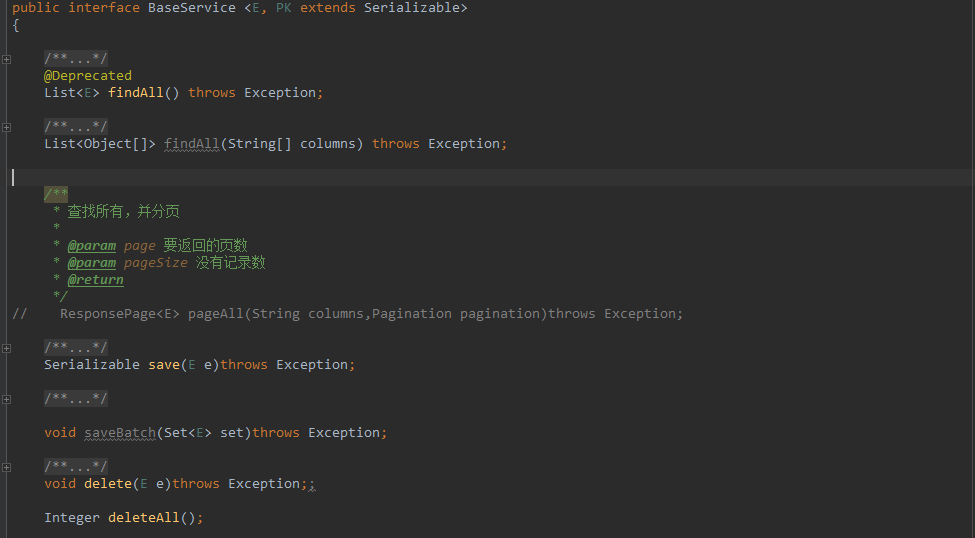


代码目录：src\main\java\com.dm.org\model\DataMiningTask.java

同时以注解方式写好了任务与学生分组一对多关系、任务与算法多对多关系。

**6.2.3 业务逻辑层的入口**

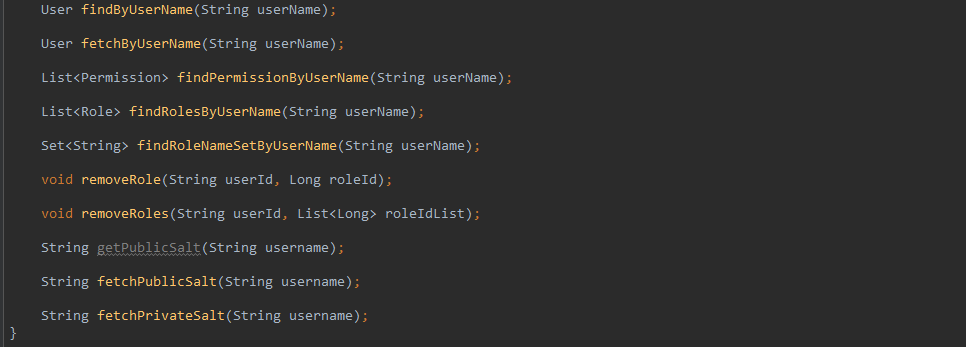
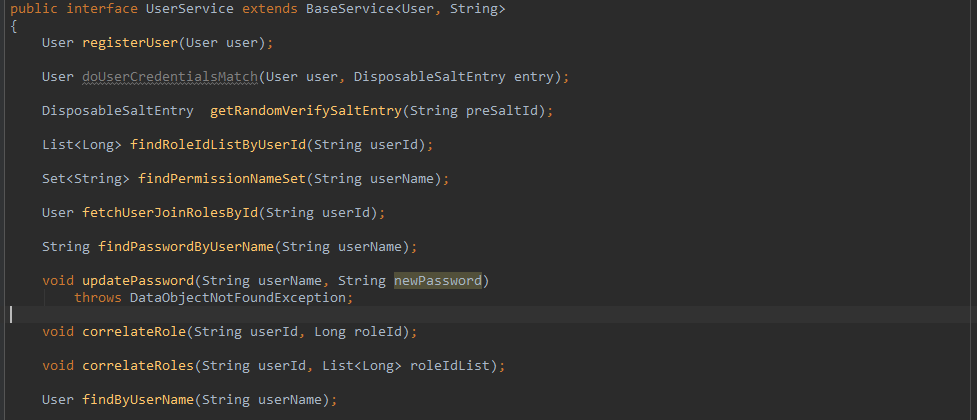
本系统采用DAO设计模式，以面向接口的思想来编程，对于所有service的开发，都有之对应接口声明规范，每个方法体应该提供对应的joc文档注释。所以在设计所有其他实体类之前，首先涉及了一个公共的DAO接口BaseService.java，同时通过BaseDao类来实现该接口，这个接口中定义了其他实体类都要用到的方法，如增、删、改、查等操作，另外在设计过程中同时定义了模糊查找、分页、更新的方法。其他实体在创建实体对象的接口后可以通过继承此接口来实现对该实体对象的增、删、改、查等操作。此实现类中用到的事务是由Spring容器提供的事务管理器，同时运用注解的方式将其注入使得编码更加方便。公共接口代码如下：



代码目录：src\main\java\com.dm.org\dao\impl\BaseDao.java

**6.2.4 DAO实现类**

DAO接口是对数据库表各种操作的抽象表示，即相关的service,对应的DAO实现类是对接口的实现，也就是对所有数据库表共同操作的具体实现。以用户表为例，给出该表的DAO接口UserDAO.java及实现类UserService.java代码如下：



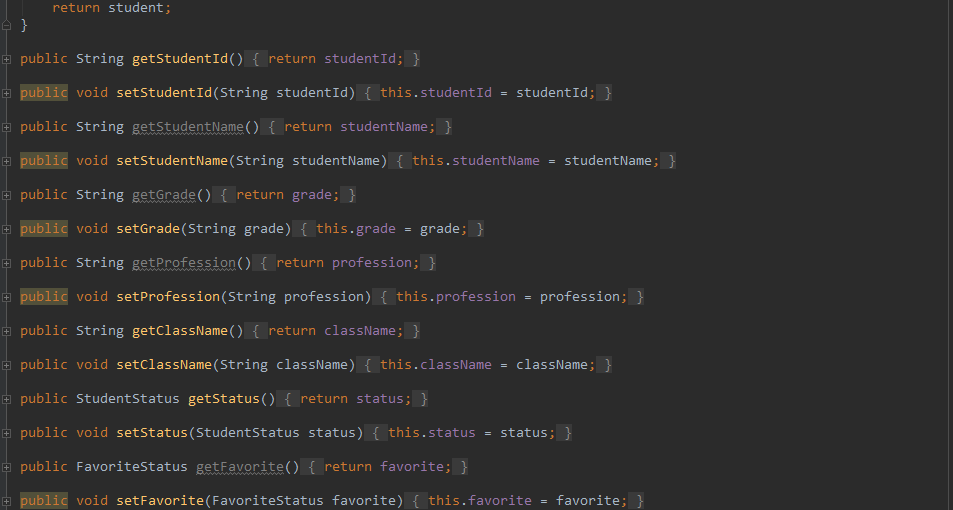
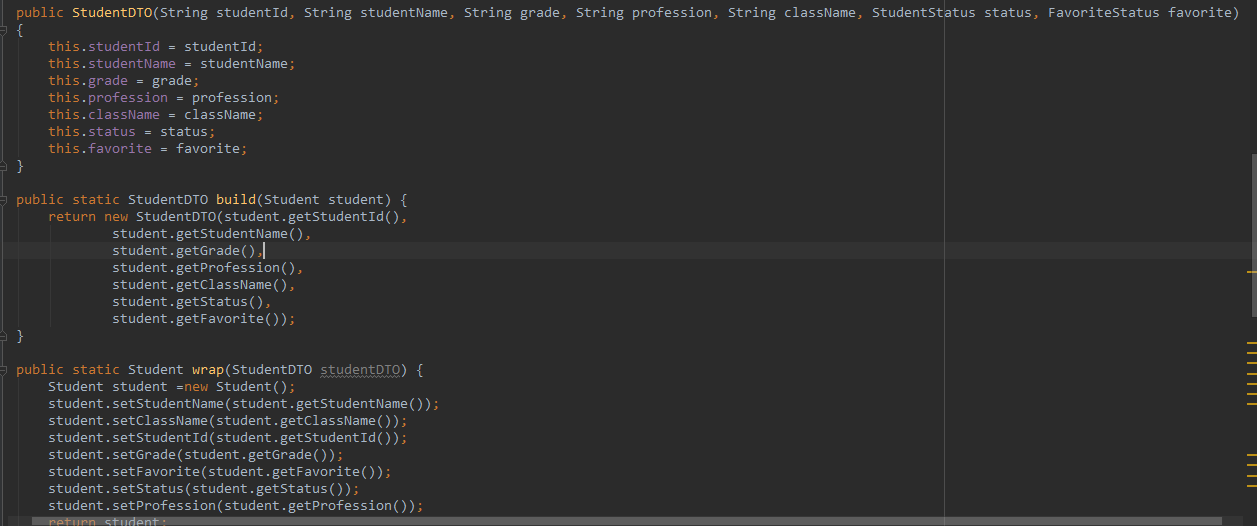


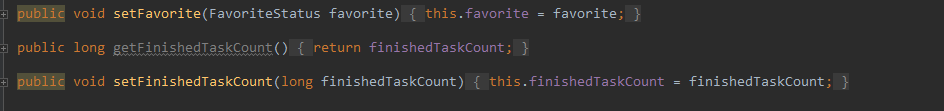
用户表接口实现类因篇幅只表一段，详见src\main\java\com.dm.org\ dao\impl\UserDao.java。

**6.2.5设计DTO接口及DTO实现类**

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。那么为何不直接将领域对象用于数据传递？因为领域对象更注重领域，而DTO更注重数据。不仅如此，由于“富领域模型”的特点，这样做会直接将领域对象的行为暴露给表现层，但数据传输对象DTO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。比如，Student领域对象可能会包含一些诸如StudentName, StudentId, grade,profession等信息。但如果UI上不打算显示grade的信息，那么StudentDTO中也无需包含这个grade的数据。简单来说Model面向业务，我们是通过业务来定义Model的。而DTO是面向界面UI，是通过UI的需求来定义的。通过DTO我们实现了表现层与Model之间的解耦，表现层不引用Model，如果开发过程中我们的模型改变了，而界面没变，我们就只需要修改Model而不需要去修改表现层中的东西。

以下给出StudentDTO.java的代码：





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

**6.2.6主要功能模块实现**

**1.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（例如输入项格式是否合法、两次密码是否一致、学号是否已存在）进行验证，若满足以上条件则注册成功，否则提示重新注册，返回注册界面。

【注册界面】

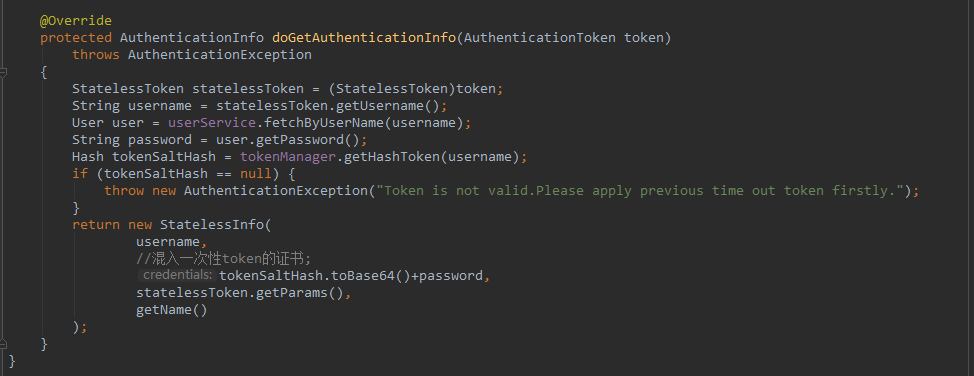
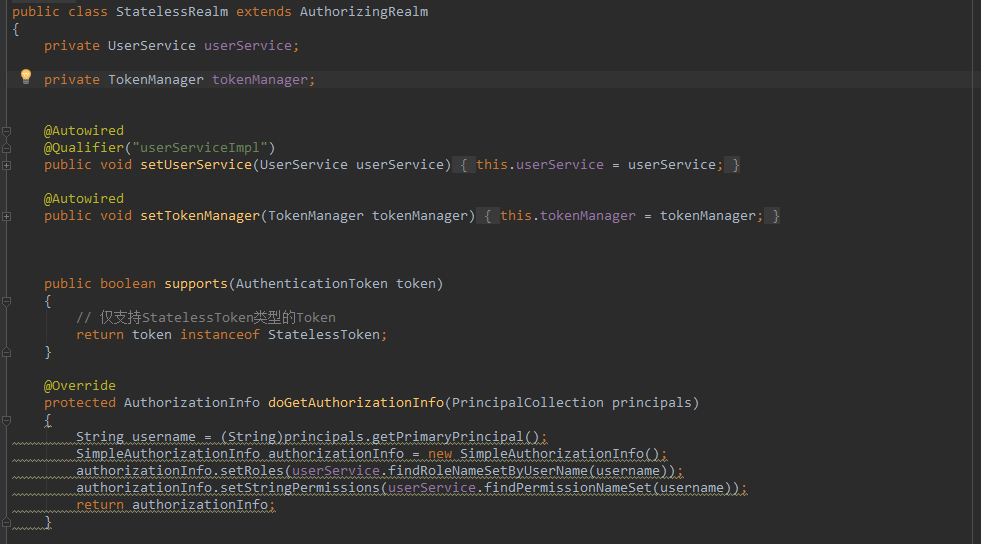
【前端的相关代码】

【代码分析】

**2）用户登录**

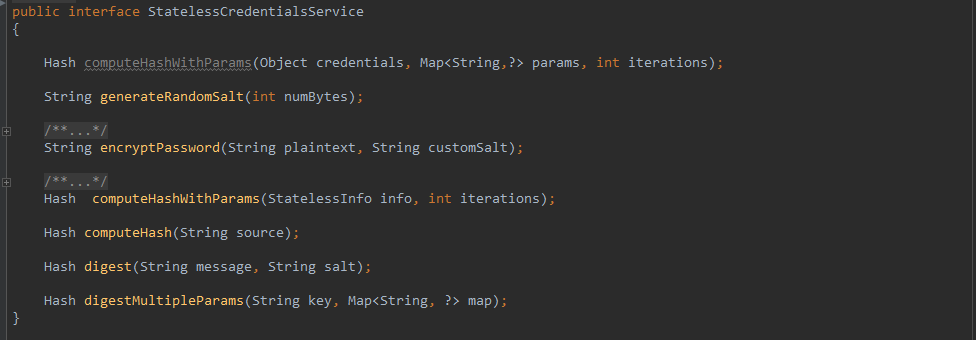
在传统的单体架构中，单个服务保存所有的用户数据，可以校验用户，并在认证成功后创建HTTP会话。在微服务架构中，用户是在和服务集合交互，每一个用户都有可能需要知道请求的用户是谁。本系统就采用了微服务的架构方案，并且服务器端是无状态的，就是说服务器端不会存储像会话这种东西，所有的资源都可以通过URL定位，而且这个定位与其他资源无关，也不会因为其他资源的变化而改变，而且每次请求时带上相应的用户名进行登录。

以StatelessRealm.java为例:【看不太懂这个，你解析下】



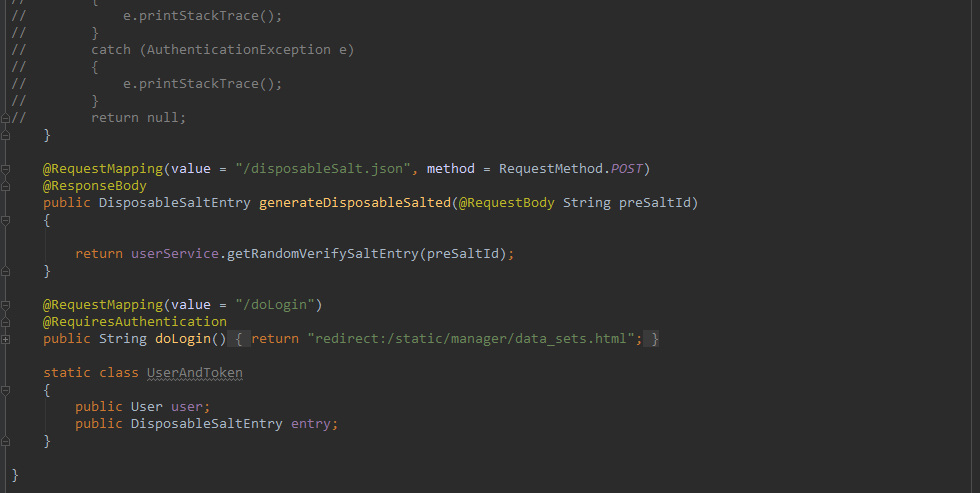
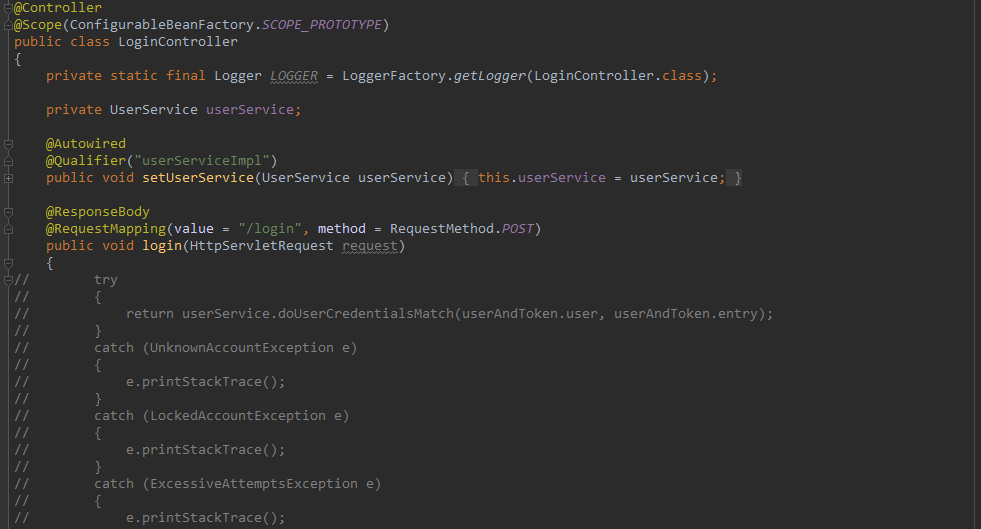
代码目录：src\main\java\com.dm.org\security\realm\StatelessRealm.java

由于微服务系统中每个服务都要处理安全问题，一些REST风格的API，如果不使用OAuth2协议，就可以使用如REST+HMAC认证进行访问。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”。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

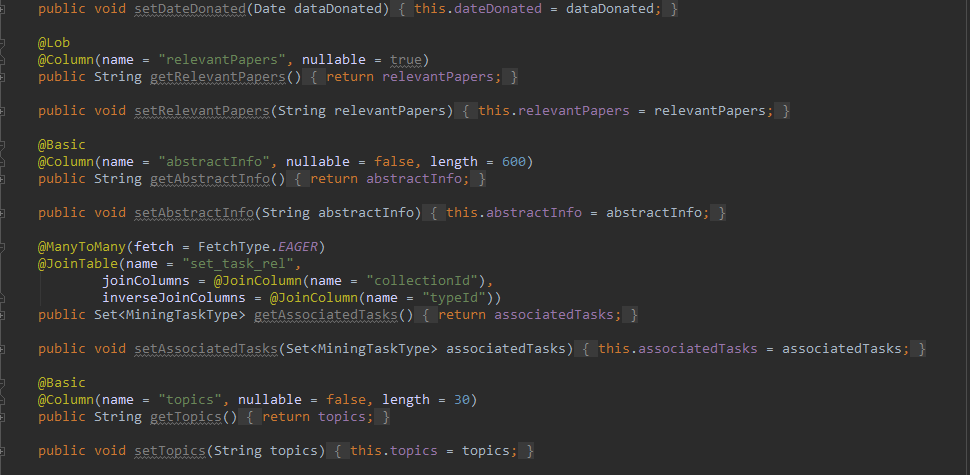
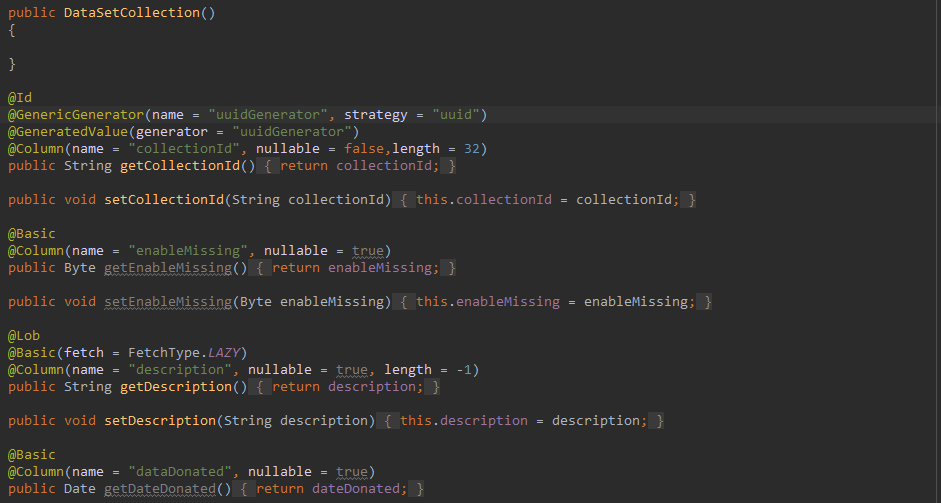
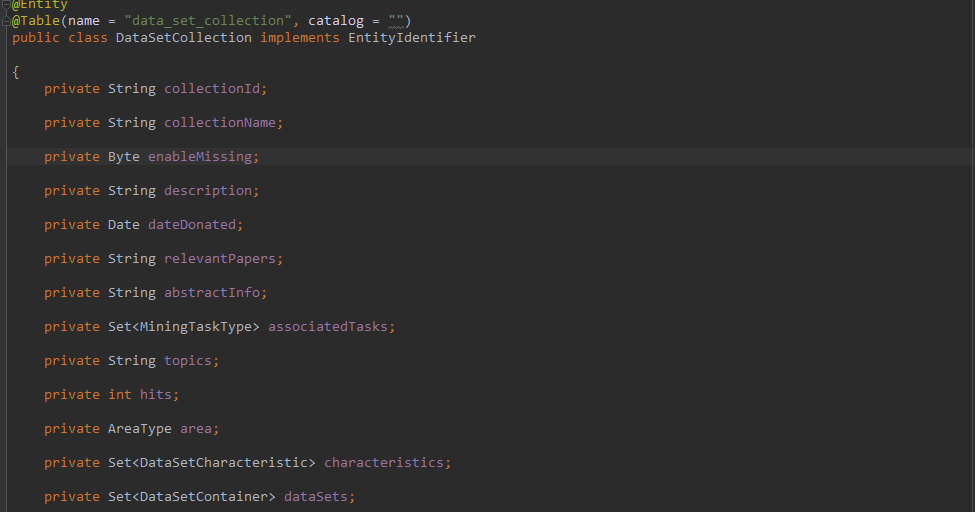
【登录界面】

**2.学生访问端**

**1）数据集浏览**

1、数据集概况展示

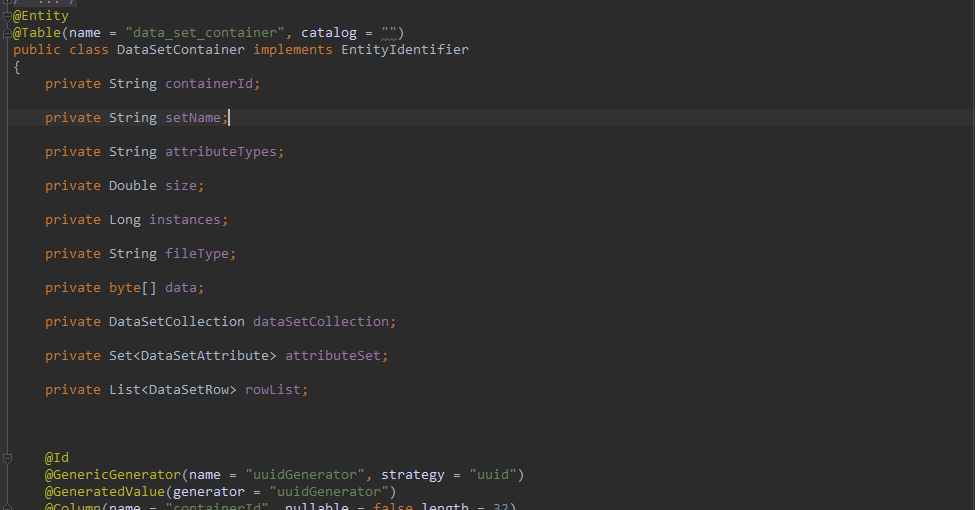
该板块的首页是展示数据集的概况，学生可通过从首页了解到该管理系统中所有数据集的数据类型、数据数量、数据挖掘形式等概括性信息，从总体上给用户直观的印象。该板块支持数据集库根据不同的分类进行排序查看，即用户可以在表中点击不同的基本信息，数据集库就会根据内容的字母或数字等顺序，进行排序展示数据集库。同时该板块也支持分类浏览数据集信息，用户可以通过点击分类框内的不同分类内容，查看所需要的类型的数据集。用户也可以通过表格查询关键字的方式直接对所需要的类型进行查询筛选。



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。

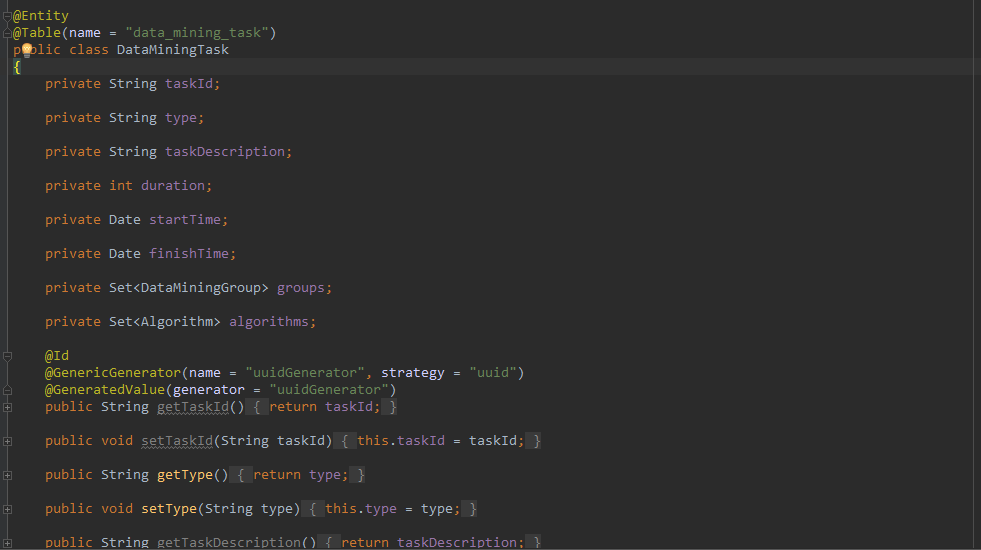
【】

【数据集界面】

**2)发掘任务**

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】

**6.3 特殊问题及解决办法**

**问题1**

**问题描述：**

**解决方案:**

**解决结果:**

**7.个人体会**

**7.1成员1 刘祥德的个人小结**

**7.2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成员3 马腾的个人小结**