<JAVAEE 第三次作业>

设计文档

程威 16301032

张小明 16301026

那涛 16301043

编写：\_程威 张小明\_那涛\_ 日期：\_\_2019-6-18\_\_

检查：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_

审核：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_

批准：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_

目录

[**1. 引言** 3](#_Toc8842715)

[**1.1 编写目的和范围** 3](#_Toc8842716)

[**1.2项目背景** 3](#_Toc8842717)

[**2. 全局数据结构说明** 3](#_Toc8842718)

[**2.1 常量** 3](#_Toc8842719)

[**3. 模块实现** 7](#_Toc8842720)

[**3.1 系统整体设计图** 7](#_Toc8842721)

[**3.2 功能说明 （分页，ratelimiter , Hateoas等，分模块具体描述）** 8](#_Toc8842722)

[**3.2.1 View 层（./src/page）** 8](#_Toc8842723)

[**3.2.2 Controller层** 13](#_Toc8842724)

[**3.2.3 Service 层** 18](#_Toc8842725)

[**3.2.4 Entity 层** 21](#_Toc8842726)

[**3.2.5 JPA Repository 层** 25](#_Toc8842727)

[**3.2.6 Relative config** 27](#_Toc8842728)

[**3.2.7 Oauth2** 28](#_Toc8842729)

[**4. 数据库实现** 28](#_Toc8842730)

[**4.1数据库整体设计** 28](#_Toc8842731)

[**4.2各模块设计** 28](#_Toc8842732)

[**4.2.1用户表** 28](#_Toc8842733)

[**4.2.2用户-角色关系表** 29](#_Toc8842734)

[**4.2.3角色表** 29](#_Toc8842735)

[**4.2.4健身俱乐部表** 29](#_Toc8842736)

[**4.2.5教练表** 29](#_Toc8842737)

[**4.2.6用户教练关系表** 30](#_Toc8842738)

[**4.2.7设备表** 30](#_Toc8842739)

**1. 引言**

**1.1 编写目的和范围**

本实现文档说明编写的目的是说明程序模块的实现细节，包括程序描述、输入/输出、算法和逻辑流程等，为软件编程和系统维护提供基础。本文档的预期读者为系统设计人员、软件开发人员、软件测试人员和项目评审人员。

**1.2项目背景**

本项目是基于Springboot2.0 框架开发得Web应用，其实现具有以下具体内容：

1. 前后端分离
2. 多表连接
3. 基于Redis的数据缓存
4. 分页查询(API pagination)
5. 注册，登陆，浏览健身俱乐部，选择私人教练四个功能实现。
6. 基于Spring JPA实现数据库访问 ，并实现多表CRUD操作
7. 多数据源：主数据源为Mysql，从数据源为H2.
8. Rate limiting 实现
9. Swagger Document 实现
10. 基于Oauth2 实现github 第三方登陆
11. Http caching 实现
12. Hateoas 约束搭建与实现
13. Kafka 异步方法处理

**2. 全局数据结构说明**

**2.1 常量**

|  |  |  |
| --- | --- | --- |
| 数据文件名 | 所在目录 | 功能说明 |
| JwtUserDetails.java | bo | Spring Security 用户实体类，用于认证 |
| AuthUserDetailsService.java | Auth | spring security认证获取用户方式 |
| MyUserDetails.java | Spring Security 用户实体类，用于认证 |
| RestAuthenticationEntryPoint.java | 认证失败处理（响应401错误） |
| WebSecurityConfig.java | Spring Security 配置 |
| Global500Exceptionhandler.java | components | 全局服务端异常控制 |
| JwtAuthenticationSuccessHandler.java | 登录成功后操作控制 |
| JwtAuthenticationTokenFilter.java | 登录验证过滤器 |
| RestAccessDeniedHandler.java | 认证失败发送401错误消息 |
| RestAuthenticationEntryPoint.java  RestAuthenticationFailureHandler.java | 认证失败发送401错误消息  认证失败发送401错误消息 |
| RateLimitAOP.java | 通过自定义注解以及切片的方式实现限流，此为切片的定义类 |
| RateLimitAspect.java | Common.annotation | 通过自定义注解的方式，实现限流操作。 |
| DataSourceConfig.java | config | 多数据源配置 |
| PrimaryConfig.java | MySql数据库配置 |
| RedisConfig.java | redis缓存配置 |
| SecondaryConfig.java | H2数据库配置 |
| WebMvcConfigurer.java | （弃用）FastJson使用配置 |
| WebSecurityConfig.java | Spring Security 配置 |
| Swagger2Config.java | 【swagger】 api 配置 |
| WebMvcConfigurer.java | Http缓存相关配置，实现Http 缓存。 |
| GymController.java | controller | gym分页请求控制器 |
| TrainerController.java | trainer分页请求，用户私人trainer添加及检索控制器 |
| UserInfoController.java | 用户信息检索及修改控制器 |
| UserSignController.java | 用户登录注册控制器 |
| AuthController.java | 用户登录控制器 |
| Gym.java | Domain.primary | 俱乐部实体 |
| Role.java | 角色实体 |
| Trainer.java | 教练实体 |
| UmUser.java | 用户实体 |
| UserInfo.java | Domain.secondary | 个人信息实体 |
| Equipment.java | 设备实体 |
| UserEntity.java | domain | 用户实体 |
| UserGithubEntity.java | Github用户实体 |
| GymRepository.java | Repository.primary | 体育馆JPA Repository |
| RoleRepository.java | 角色JPA Repository |
| TrainerRepository.java | 教练JPA Repository |
| UserRepository.java | 用户JPA Repository |
| UserInfoRepository.java | Repository.secondary | 个人信息 JPA Repository |
| EquipmentRepository.java | 设备 JPA Repository |
| EquipmentResource.java | Dto.hateoas | 设备资源类 |
| GymResource.java | 俱乐部资源类 |
| TrainResource.java | 教练资源类 |
| UserInfoResource.java | 用户个人信息资源类 |
| UserResource.java | 用户资源类 |
| EquipmentResourceAssembler.java | Dto.hateoas.hateoas | 设备资源组装类 |
| GymResourceAssembler.java | 体育馆资源组装类 |
| TrainerResourceAssembler.java | 教练资源组装类 |
| UserInfoResourceAssembler.java | 用户个人信息资源组装类 |
| UserResourceAssembler.java | 用户资源组装类 |
| GithubAuthServiceResult.java | Dto | github oauth认证服务结果对象 |
| RestResponse.java | Rest响应实体 |
| UserProfile.java | 用户信息 |
| UserSignUpRequest.java | 用户登录参数实体 |
| DataService.java | Service | gym和trainer实体类分页检索，相关实体数据修改，多表查询，添加private trainer接口 |
| JwtUserDetailsService.java | spring security获取用户方式 |
| UserService.java | 用户登录，注册，信息检索及修改接口 |
| GithubService.java | Github授权服务 |
| DataServiceImpl.java | Service.impl | gym和trainer实体类【分页】检索，【服务层必要数据缓存】,数据修改，多表查询，添加private trainer实现类 |
| UserServiceImpl.java | 用户登录，注册，信息检索及修改实现类，【H2使用】 |
| KafkaConsumer | Kafka 消息消费者，监听器方法，进行异步业务处理 |
| KafKaProducerService | Kafka 消息生产者，异步发送端 |
| GithubServiceImpl.java | 实现类 |
| JwtTokenUtil.java | Util | JwtToken生成，校验，刷新，从token获取信息工具类 |
| RedisOperator.java | Redis操作工具类 |
| UserUtil.java | 用户工具类 |
| application.yml | resources | spring boot 服务器配置 |
| gym.jpg | Resources. Static.img |  |
| head.png |  |
| trainericon.jpg |  |
| GymClubSystemApplicationTests.java | Test.java.com.xming.gymclubsystem | 测试类 |

**3. 模块实现**

**3.1 系统整体设计图**



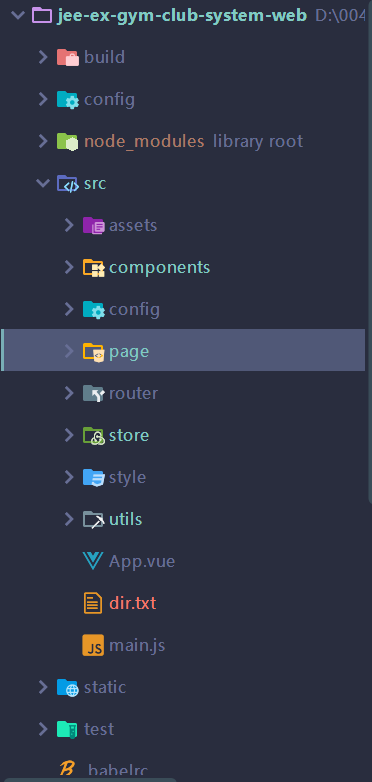
系统整体设计图

**3.2 功能说明 （分页，ratelimiter , Hateoas等，分模块具体描述）**

从上图可以看出，我们的整体系统划分为view层、controller层、service层、entity 层、 JPA repository层五个模块，分别具体实现项目简介功能。

**3.2.1 View 层（./src/page）**

前端工程目录：

****

**3.2.1.1** home模块

描述：首页，显示gym列表，trainer列表

组件：GymContainer：发送http请求，获取gym分页列表，调用 imgTextCard 组件渲染列表

TrainerContainer：发送http请求，获取Trainer分页列表，调用 trainerItem组件渲染列表

界面：





**3.2.1.2 layout模块**

描述：整个web页面总体框架

组件： appMain：页面内容容器，所有除登录注册外页面内容均在此渲染

headBar：顶部标题，显示路由标题信息

sidebar：侧边栏导航栏，动态路由到各页面

3.2.1.3 login模块

描述：登录页面

界面：



3.2.1.4 myTrainer

描述：显示private trainer 列表

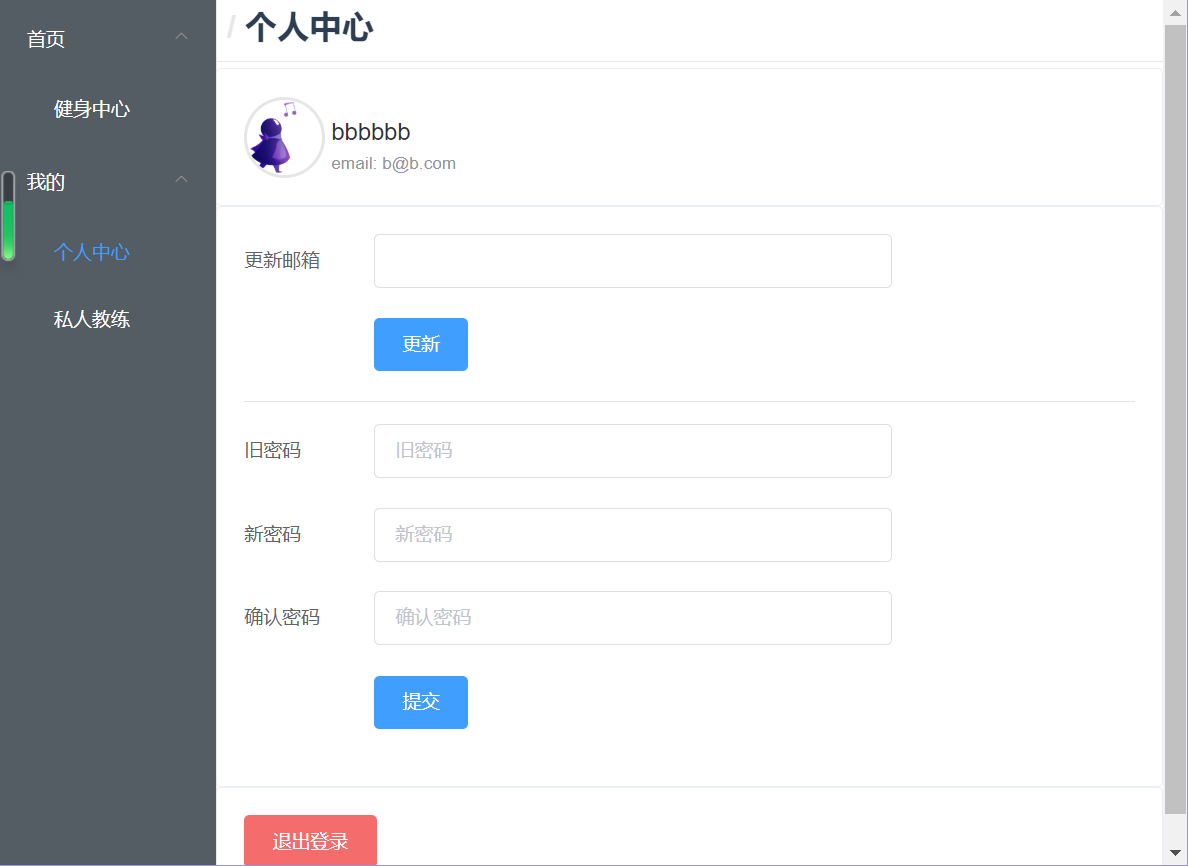
界面：



3.2.1.5 profile模块

描述：用户个人信息显示，及个人信息修改，包括邮箱和密码

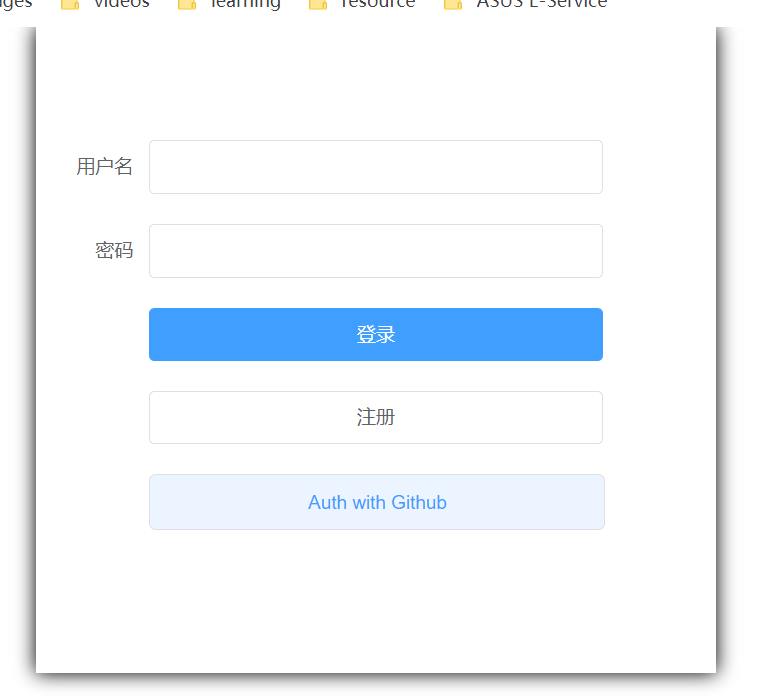
界面：



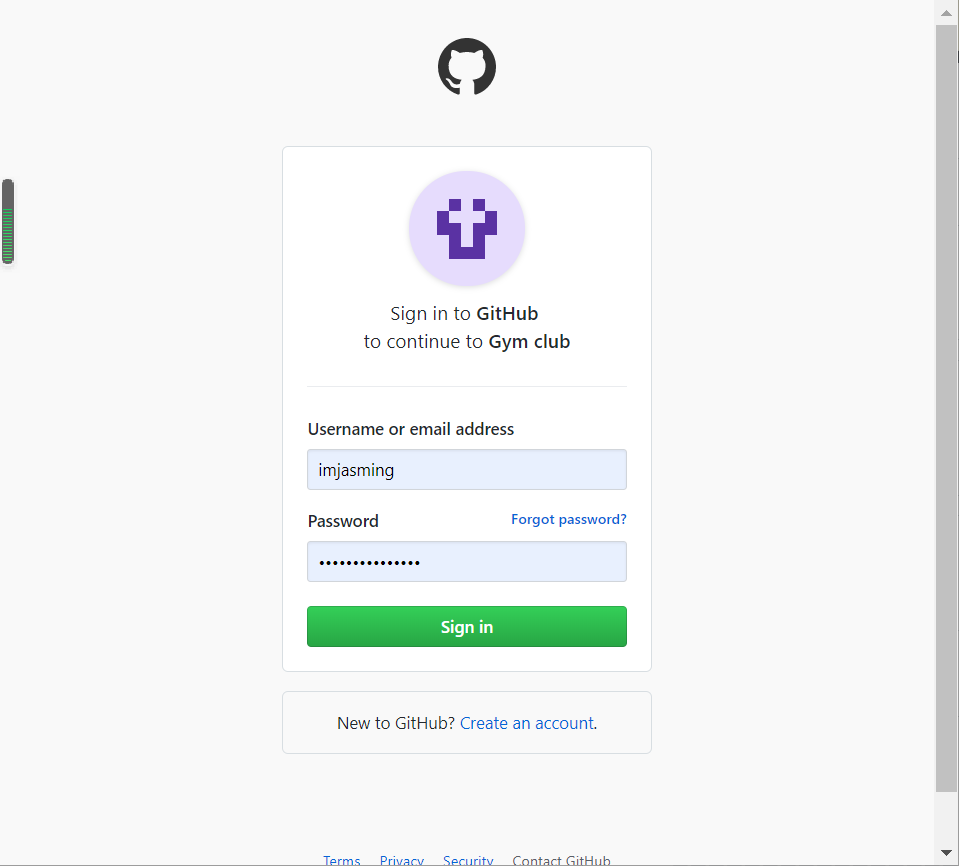
3.2.1.6 register模块

描述：注册页面

界面：



3.2.1.7 Oauth登陆模块

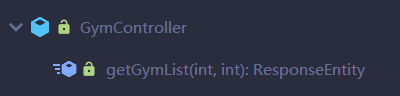


**3.2.2 Controller层**

**3.2.2.1 GymController**

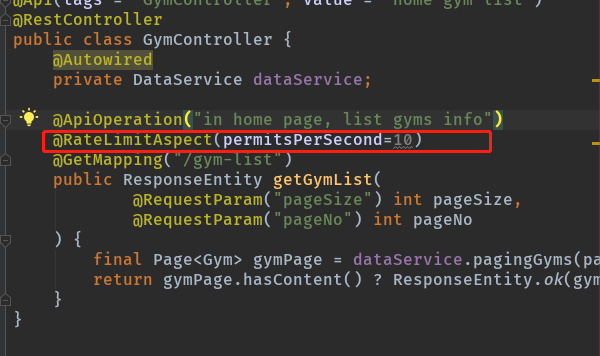
描述1：响应gym 数据请求，根据分页大小和分页编号返回gym列表分页

方法：



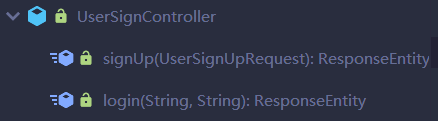
描述2：通过自定义的注解的方式，实现访问限流，限制为每秒最多10次请求。

方法：



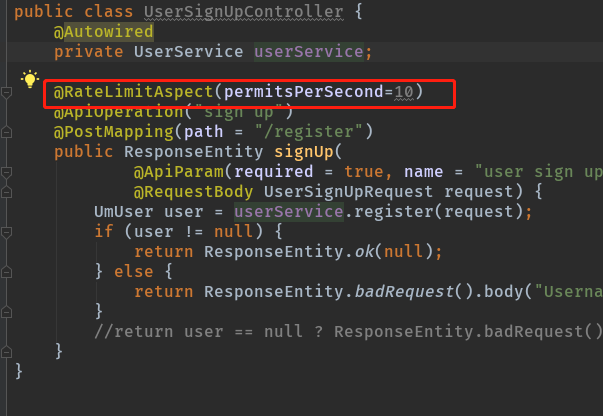
**3.2.2.2 UserSignUpController**

描述1：响应前端用户登录注册请求。登录成功后返回Token。

方法列表：

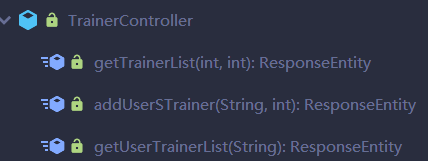
描述1：通过自定义的注解的方式，实现访问限流，限制为每秒最多10次请求。

方法：

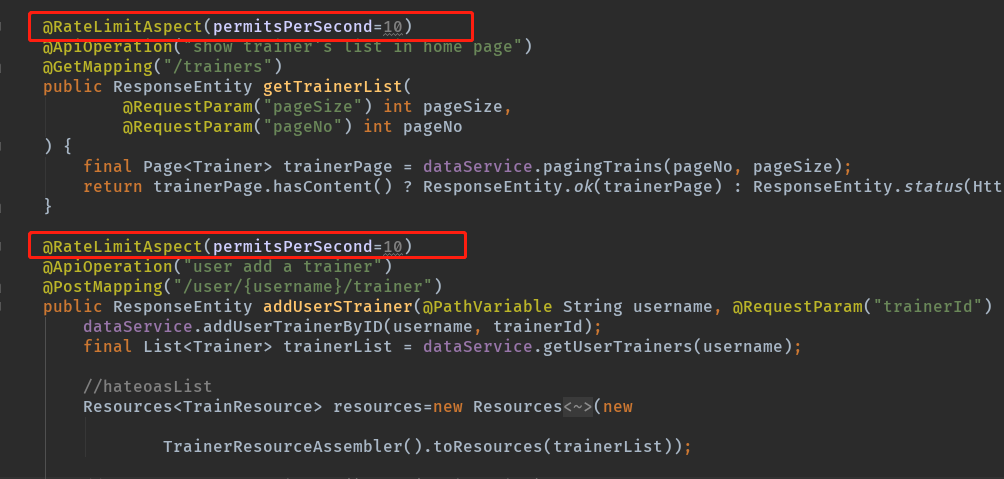


**3.2.2.3 TrainerController**

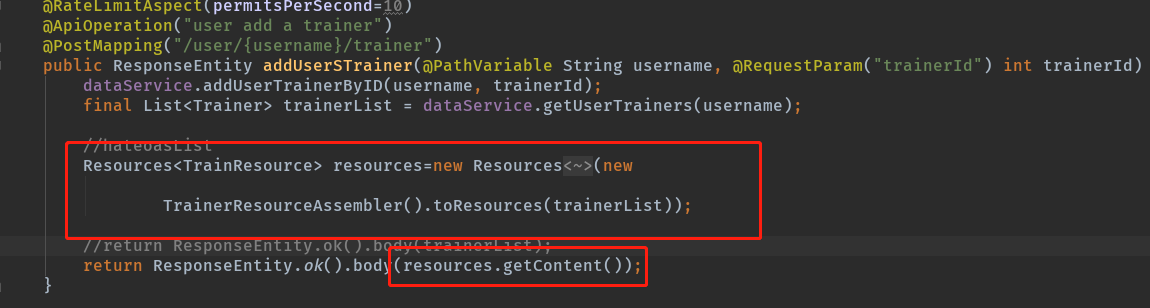
描述1：响应前端对 Trainer 数据请求，包括请求Trainer分页列表，获取用户private trainer，用户添加 private trainer

 方法列表：

描述2：通过自定义的注解的方式，实现访问限流，限制为每秒最多10次请求。

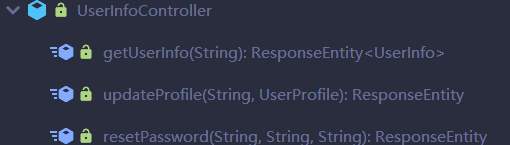
方法：

描述3：通过资源组装类组装trainer资源集合，实现hateoas，在返回链接上增加links信息。

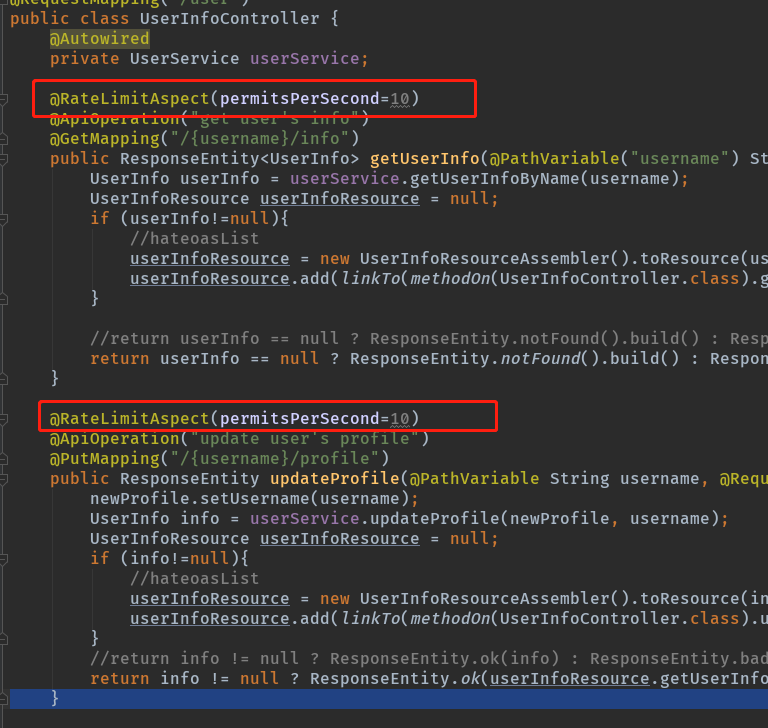
方法：

**3.2.2.4 UserInfoController**

描述1：响应前端对用户信息数据请求，包括获取用户信息，更新用户信息，重置密码。

 方法列表：

描述2：通过自定义的注解的方式，实现访问限流，限制为每秒最多10次请求。

方法：

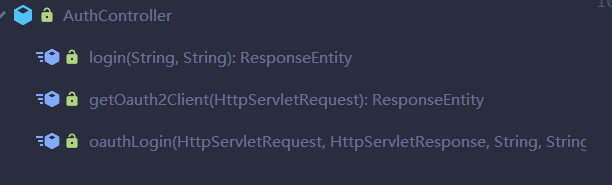
描述3：通过资源组装类组装userinfo资源，实现hateoas，在返回链接上增加links信息。

方法：

**3.2.2.5 AuthController**

描述1：分发前端登录请求

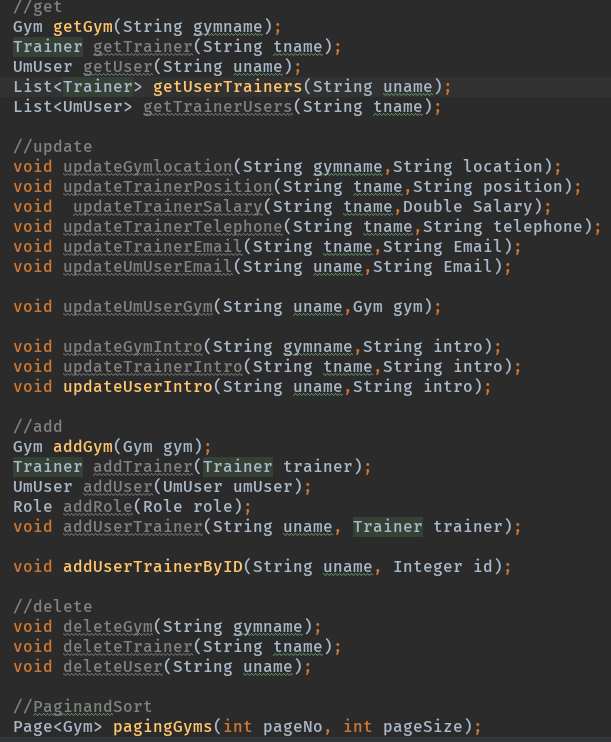
方法列表：



**3.2.3 Service 层**

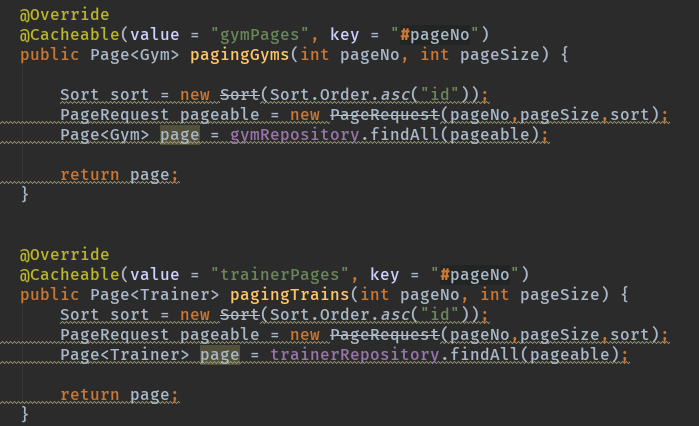
**3.2.3.1 DataService**

描述：1. 实现Gym表等其他7个表的CRUD操作

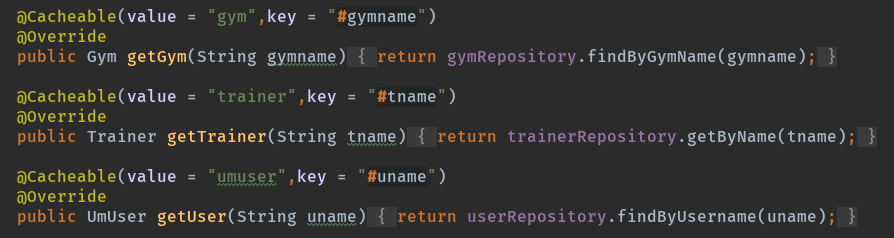
方法列表：

2. 实现分页操作

方法列表：



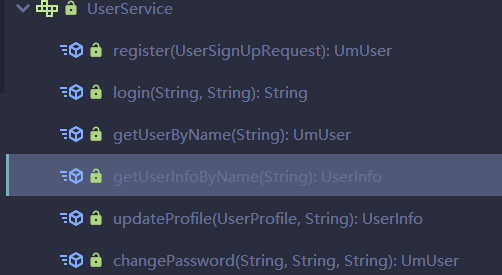
3. Redis cacheable 缓存实现

方法列表：

**3.2.3.2 UserService**

描述：用户注册，用户认证成功后返回token，获取用户信息，更新用户信息

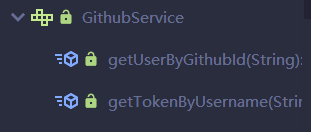
方法列表：



**3.2.3.3 GithubService**

描述： github授权服务处理

方法列表：



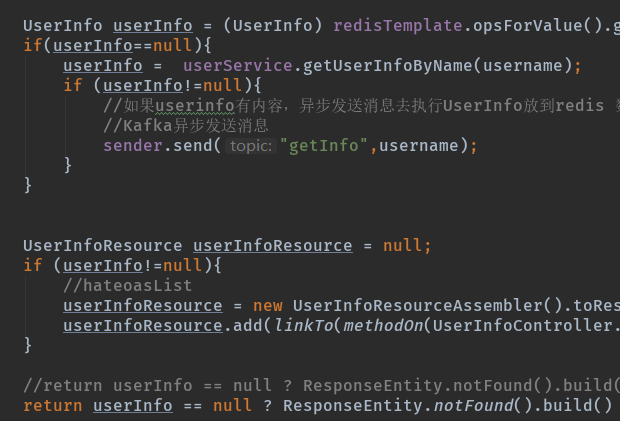
**3.2.3.3 KafkaConsumer**

描述： Kafka 消息消费者，监听器方法，进行异步业务处理

分别描述：

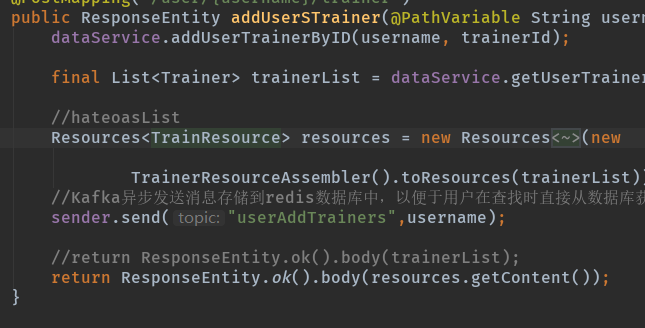
1. 用户在查询个人信息时，发送getinfo topic ，传入username message，在后台异步进行查询，然后放到redis数据库中，方便下次查询时，直接从redis数据库中读取。

对应controller 端异步发送代码逻辑，先从redis数据库中读取，如果内容为空，则进行异步发送，进行后台处理，如果不空，则直接从redis数据库中读取。



1. 用户在添加自己的trainer时，顺便发送userAddTrainers topic ， 传入username message ，即可在后台异步进行查询，然后把查询结果放入redis 数据库，方便下次用户添加教练时直接从redis 数据库中进行读取，加快处理效率。

对应controller 代码逻辑，直接异步发送给后端，方便在获取自己的trainer时，直接从redis 数据库中读取，加快查询的效率。



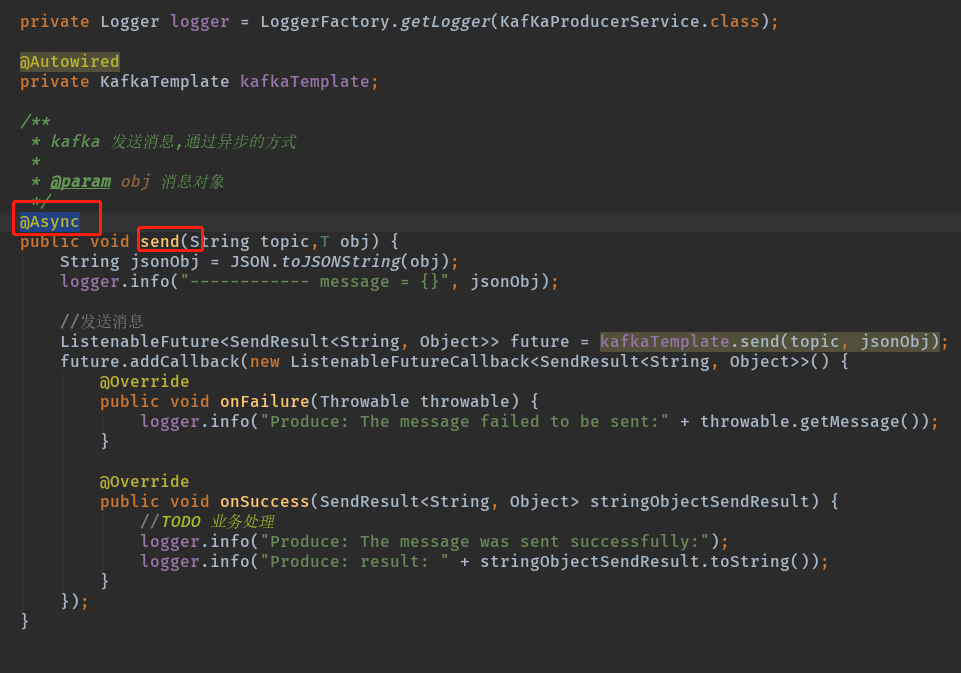
1. 用户在获取自己的教练时，同样进行上述操作，异步存入reids数据库中，方便下次直接从redis 数据库中读取。

对应controller 逻辑：先查询redis 数据库，如果redis数据库为空，则异步发送用户名，交给后台异步处理，存入redis 数据库中，为下次数据查询提升效率。



**3.2.3.3 KafKaProducerService**

描述： Kafka 消息生产者，异步发送端，@Async 标记

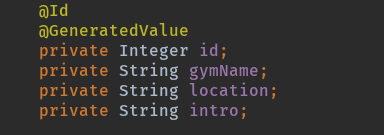
具体实现：

**3.2.4 Entity 层**

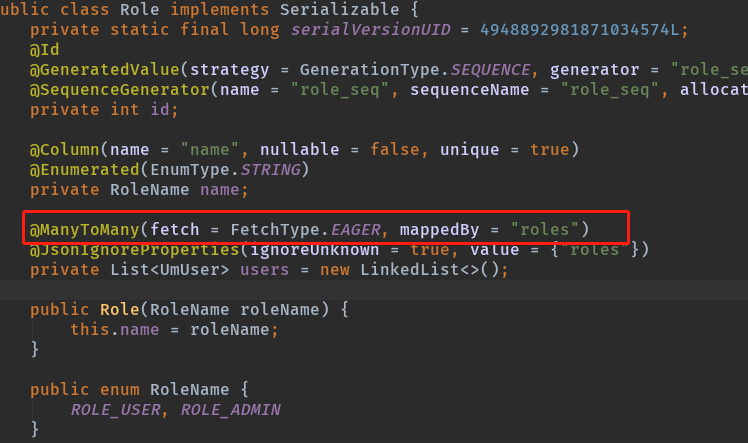
**3.2.4.1 Primary（主数据源）：**

**3.2.4.1.1 Gym:**

描述：Gym表相关属性及主键定义



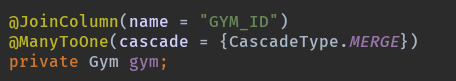
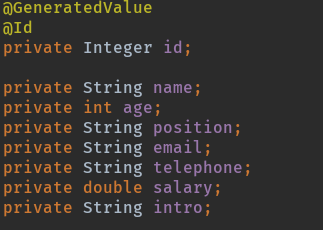
**3.2.4.1.2 Role:**

描述：Role表相关属性以及主键定义，此外，作为多对多关系的被维护端。

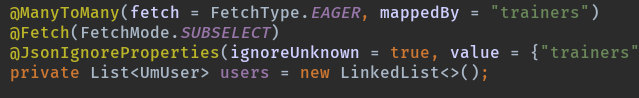
**3.2.4.1.3 Trainer:**

描述：

1. Trainer 表相关属性以及主键定义

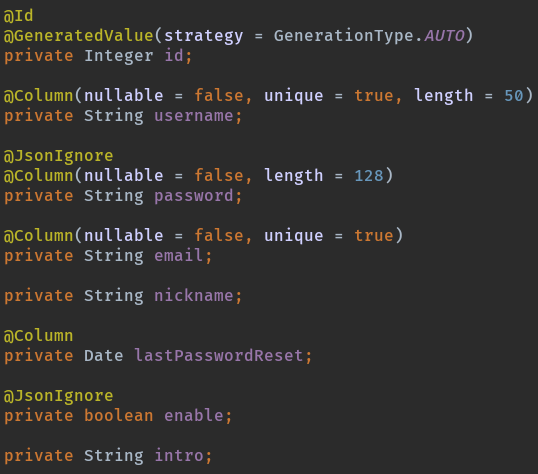
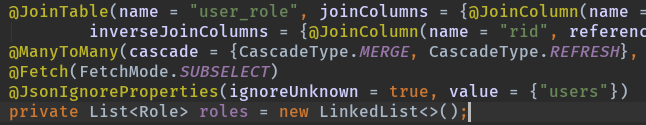
2. 多对一关系（相对于体育馆）的维护端

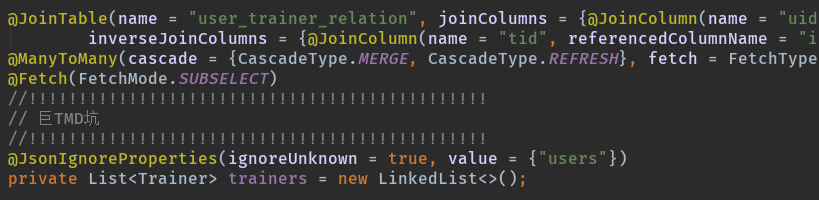
3. 多对多关系的被维护端

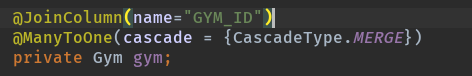


**3.2.4.1.4 UmUser:**

描述：

1. UmUser表相关属性以及主键定义
2. 多对多关系（相对于角色表）的主维护端
3. 多对多关系（相对于教练表）的主维护端

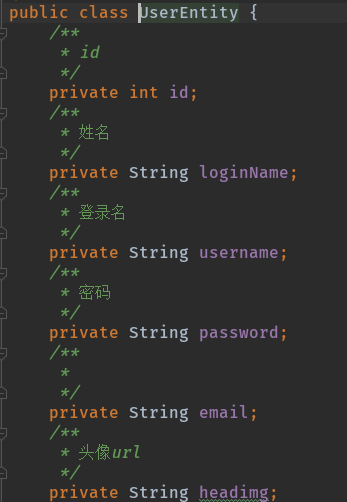


1. 多对一关系（相对于体育馆）的主维护端

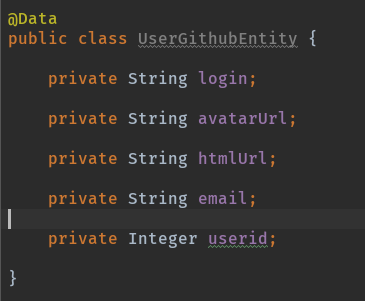
**3.2.4.1.5 UserAuth:**

描述：用户认证关系实体

**3.2.4.1.6 UserEntity:**

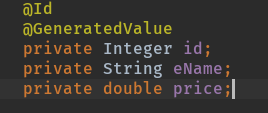
描述：用户实体

**3.2.4.1.7 UserGithubEntity**

描述：github用户实体

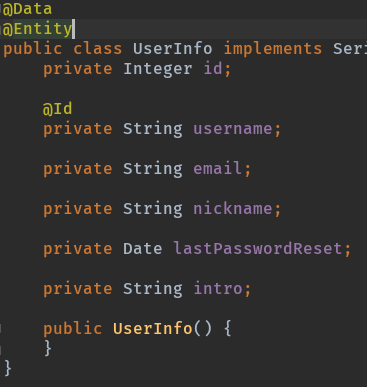
**3.2.4.2 Secondary（从数据源）:**

**3.2.4.2.1 Equipment:**

描述：Equipment表相关信息以及主键定义；

**3.2.4.2.2 UserInfo:**

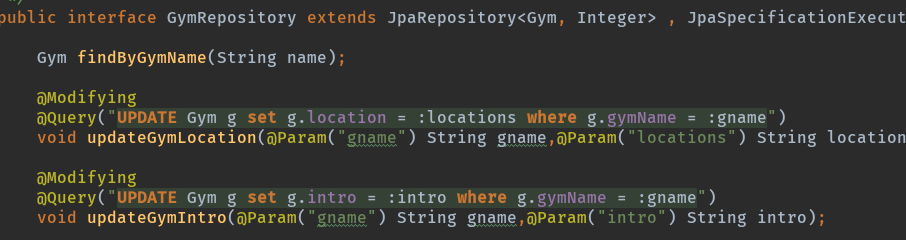
描述：UserInfo表相关信息以及主键定义, 此外，用作profile缓存。

**3.2.5 JPA Repository 层**

**3.2.5.1 Primary：**

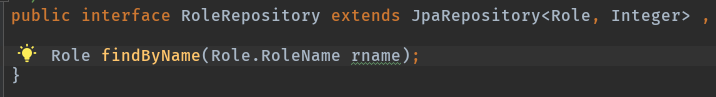
**3.2. 5.1.1 GymRepository:**

描述：

1. 使用JPA提供的JpaRepository 接口实现分页查询以及CRUD操作。
2. Query语句的使用来实现更新数据库字段的功能

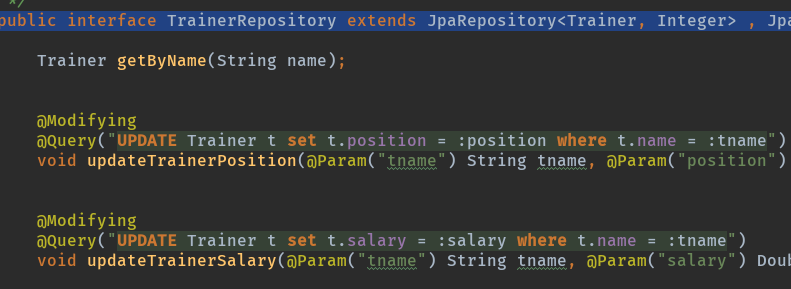
**3.2. 5.1.2 RoleRepository:**

描述：

1. 使用JPA提供的JpaRepository 接口实现CRUD操作。

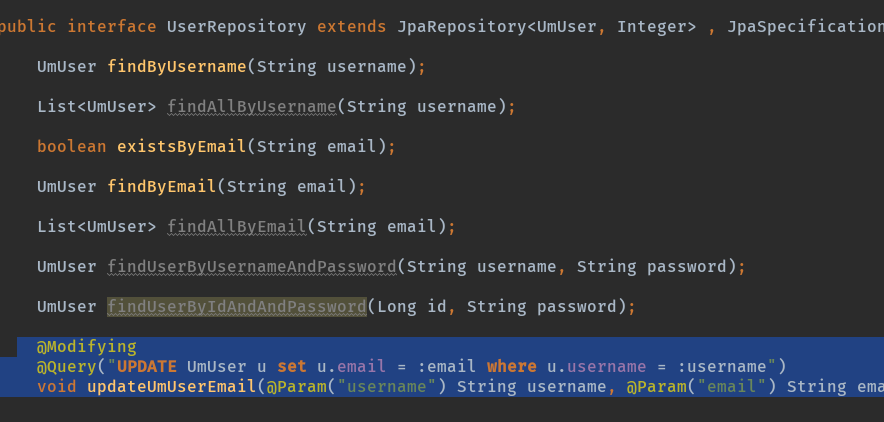
**3.2.5.1.3 TrainerRepository:**

描述：

1. 使用JPA提供的JpaRepository 接口实现分页查询以及CRUD操作
2. Query语句的使用来实现更新数据库字段的功能
3. 根据JPA 规范来书写查询方法

**3.2.5.1.4 UmUserRepository:**

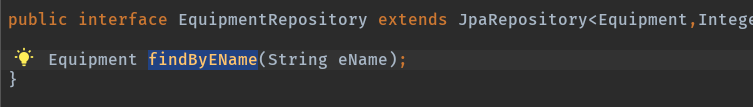
描述：

1. 使用JPA提供的JpaRepository 接口实现分页查询以及CRUD操作。
2. Query语句的使用来实现更新数据库字段的功能
3. 根据JPA 规范来书写查询方法

**3.2.5.2 Secondary:**

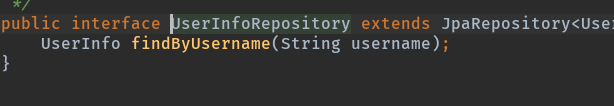
**3.2.5.2.1 EquipmentRepository:**

描述：

1. 使用JPA提供的JpaRepository 接口实现CRUD操作。
2. 根据JPA 规范来书写查询方法

**3.2.5.2.2 UserInfoRepository:**

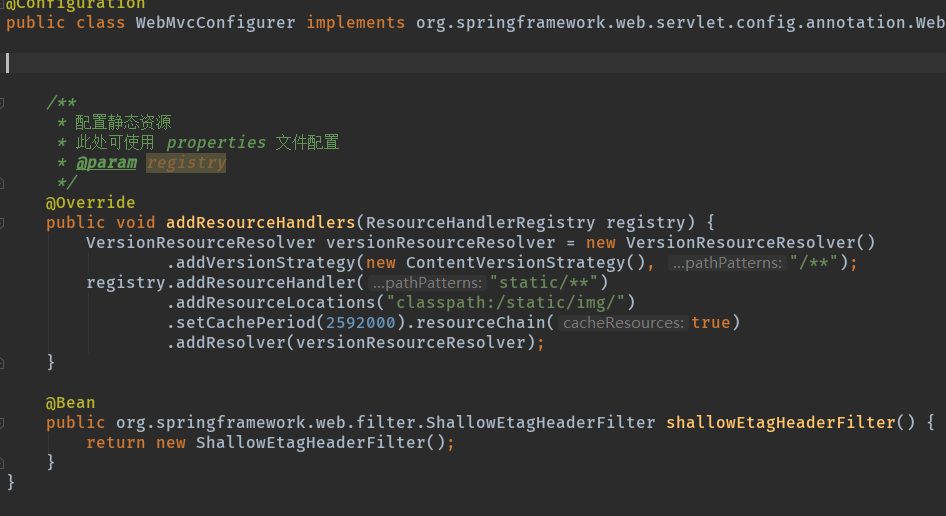
描述：

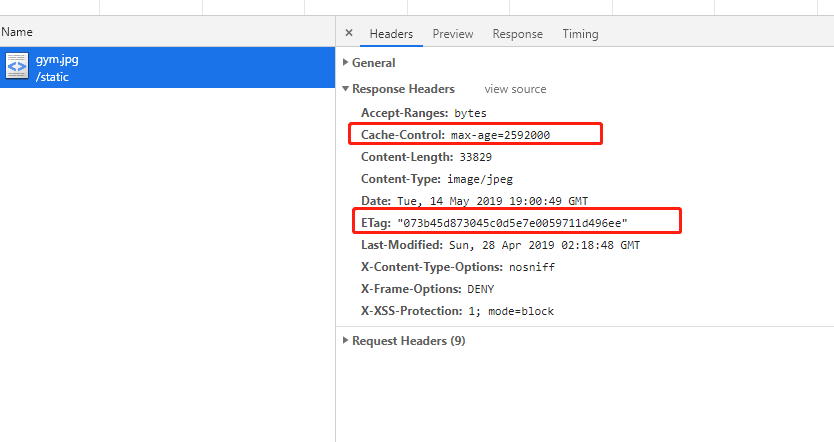
1. 使用JPA提供的JpaRepository 接口实现CRUD操作。
2. 根据JPA 规范来书写查询方法

**3.2.6 Relative config**

**3.2.6.1 Http caching config**

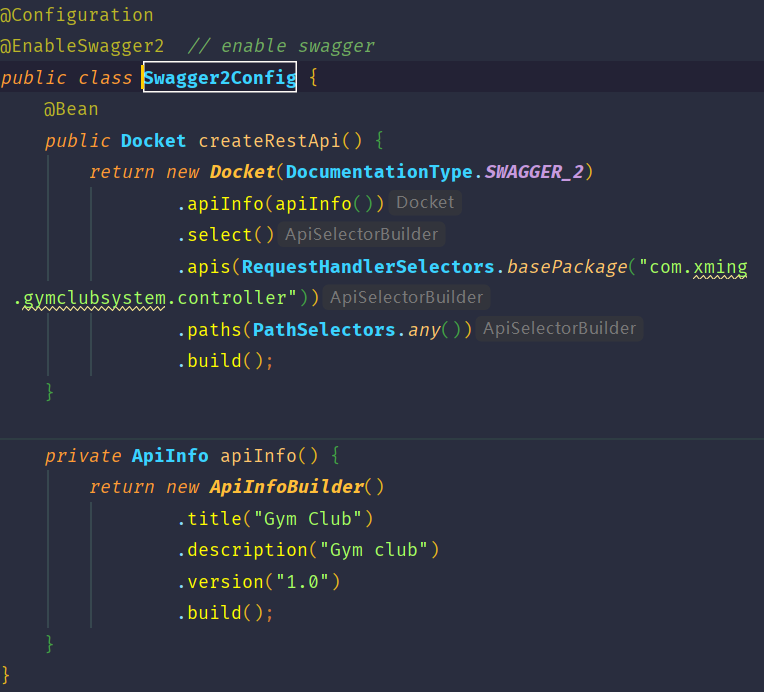
**描述：通过配置WebMvcConfigurer来实现Http静态资源缓存**

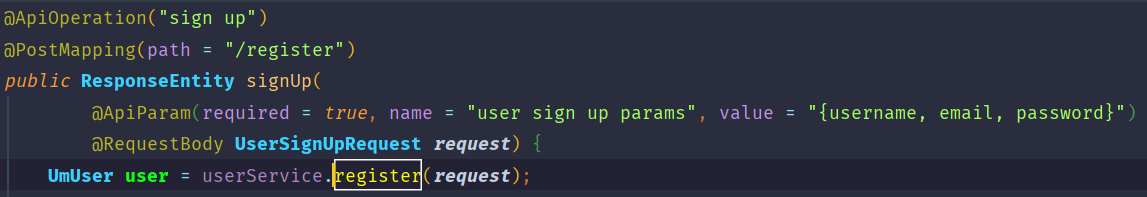
**实现：**

**客户端静态资源实现实例：**

* + - 1. **Swagger config**

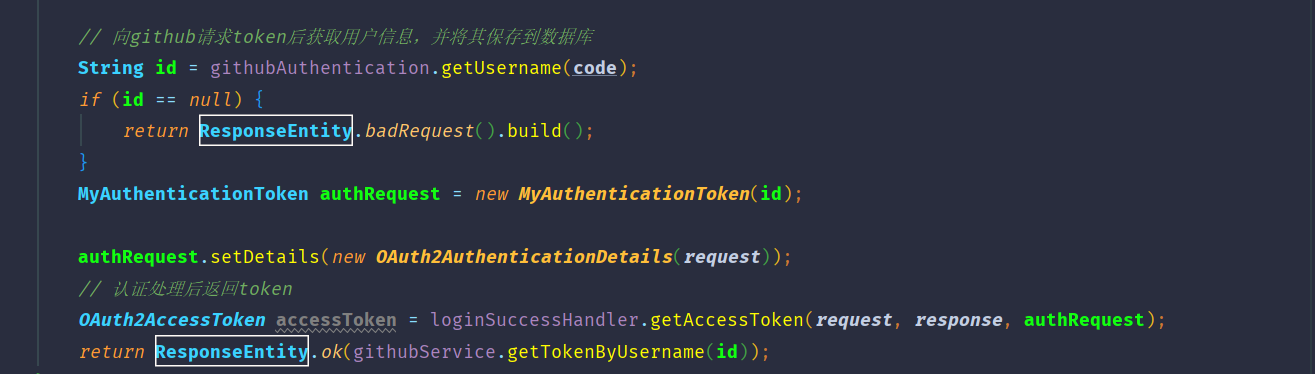
配置：



使用：

**3.2.7 Oauth2**

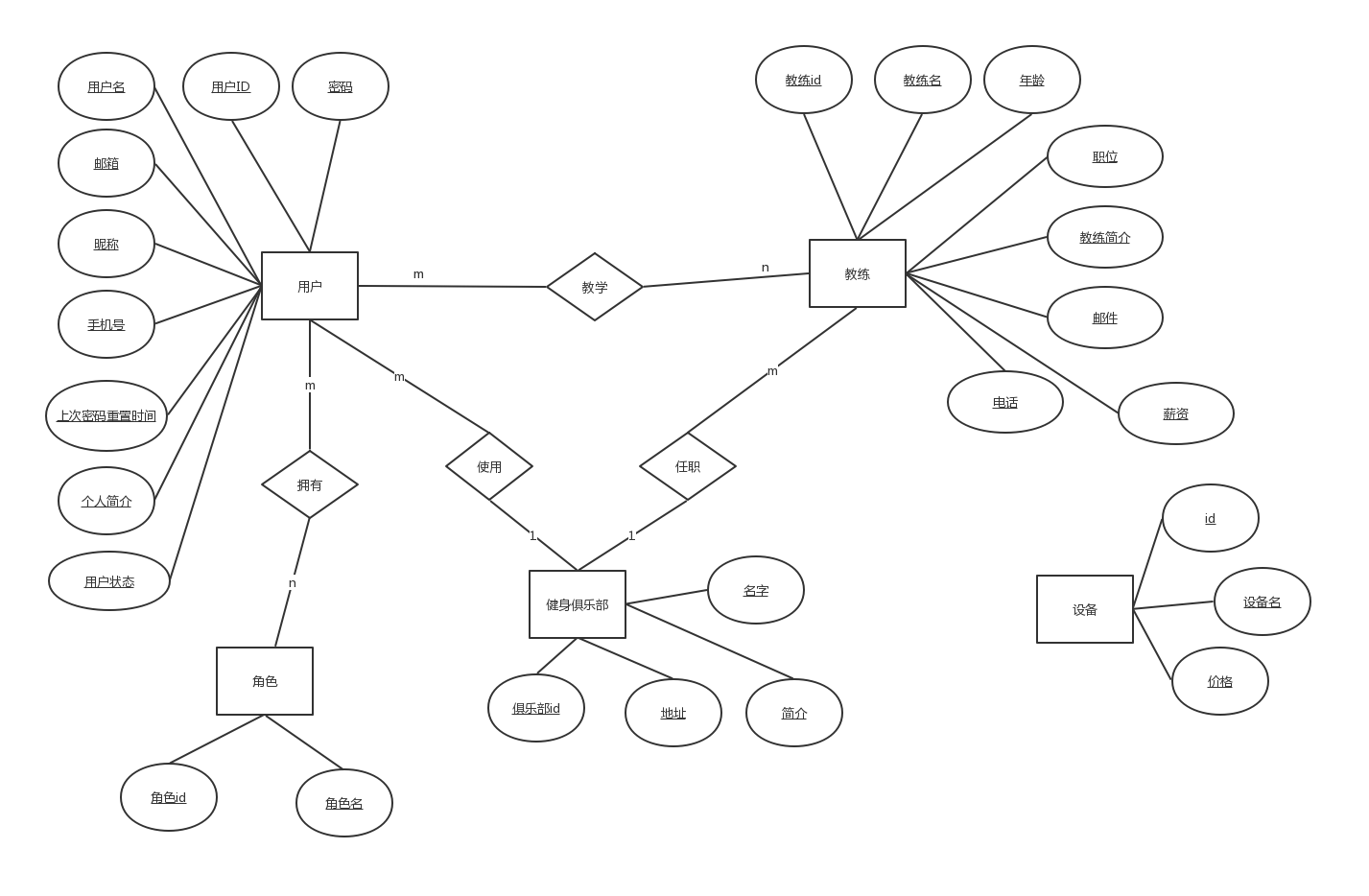
**oauth登录控制器认证流程**

****

**4. 数据库实现**

**4.1数据库整体设计**

数据库整体设计分为以下5个模块：用户、教练、健身俱乐部、角色、设备。其相互关系如下图：



数据库整体设计图

**4.2各模块设计**

**4.2.1用户表**

|  |  |  |
| --- | --- | --- |
| 字段说明 | 字段名称 | 类型 |
| 用户id | **id** | int(11) PK |
| 电子邮箱 | email | varchar(255) |
| 用户状态 | enable | bit(1) |
| 用户简介 | intro | varchar(255) |
| 上次重置密码日期 | last\_password\_reset | datetime |
| 昵称 | nickname | varchar(255) |
| 密码 | password | varchar(128) |
| 用户名 | **username** | varchar(50) |
| 俱乐部ID | **gym\_id** | int(11) |

**4.2.2用户-角色关系表**

|  |  |  |
| --- | --- | --- |
| 字段说明 | 字段名称 | 类型 |
| 用户id | **uid** | int(11) |
| 角色id | **rid** | int(11) |

**4.2.3角色表**

|  |  |  |
| --- | --- | --- |
| 字段说明 | 字段名称 | 类型 |
| 角色id | **id** | int(11) PK |
| 角色名 | name | varchar(255) |

**4.2.4健身俱乐部表**

|  |  |  |
| --- | --- | --- |
| 字段说明 | 字段名称 | 类型 |
| 俱乐部id | **id** | int(11) PK |
| 名字 | gym\_name | varchar(255) |
| 简介 | intro | varchar(255) |
| 地点 | location | varchar(255) |

**4.2.5教练表**

|  |  |  |
| --- | --- | --- |
| 字段说明 | 字段名称 | 类型 |
| 教练id | **id** | int(11) PK |
| 年龄 | age | int(11) |
| 电子邮件 | email | varchar(255) |
| 简介 | intro | varchar(255) |
| 姓名 | name | varchar(255) |
| 职位 | position | varchar(255) |
| 月工资 | salary | double |
| 电话 | telephone | varchar(255) |
| 健身馆id | **gym\_id** | int(11) |

**4.2.6用户教练关系表**

|  |  |  |
| --- | --- | --- |
| 字段说明 | 字段名称 | 类型 |
| 用户id | **uid** | int(11) |
| 教练id | **tid** | int(11) |

**4.2.7设备表**

|  |  |  |
| --- | --- | --- |
| 字段说明 | 字段名称 | 类型 |
| 设备id | Id | int(10) |
| 设备名 | E\_name | varchar(255) |
| 设备价格 | price | double |