Blog API 复刻指南

TODOlist1: 任务规划

1. [ ] 初始化 Spring Boot 项目结构
2. [ ] 配置数据源 (Druid, MySQL)
3. [ ] 配置 JPA
4. [ ] 配置 Redis 缓存
5. [ ] 配置 Shiro 安全框架
6. [ ] 配置 Log4j2 日志
7. [ ] 配置邮件服务
8. [ ] 实现用户模块 (Entity, Repository, Service, Controller, VO)
9. [ ] 实现文章模块 (Entity, Repository, Service, Controller, VO)
10. [ ] 实现标签模块 (Entity, Repository, Service, Controller, VO)
11. [ ] 实现分类模块 (Entity, Repository, Service, Controller, VO)
12. [ ] 实现评论模块 (Entity, Repository, Service, Controller, VO)
13. [ ] 实现日志记录 (AOP)
14. [ ] 实现全局异常处理
15. [ ] 实现 OAuth2 (如果需要)
16. [ ] 编写单元测试和集成测试
17. [ ] 完善接口文档

filetodolist: 文件实现顺序及目标

- `pom.xml`：定义项目依赖 (Spring Boot Starter Web, JPA, MySQL, Redis, Shiro, Log4j2, Mail, Druid, Fastjson, Commons Lang3/Collections) 和构建配置。
  - **意义**：项目构建和依赖管理的核心配置文件。
  - **主要内容**：项目坐标、父项目、依赖项、构建插件、仓库配置。
- `application.properties`：配置服务器端口、数据库连接、Redis 连接、Shiro 配置、日志级别、邮件服务器等。
  - **意义**：Spring Boot 核心配置文件，配置应用行为和集成组件。
  - **主要内容**：服务端口、文件上传、静态资源、数据库(Druid)、JPA/Hibernate、Redis、日志配置路径。
  - **注意**：邮件服务和 Shiro 的配置似乎不在此文件中，可能在代码中配置或需要补充。
- `src/main/java/com/shimh/common/cache/RedisManager.java`：封装 Redis 常用操作。
  - **意义**：提供便捷的 Redis 读写删接口，统一管理默认过期时间。
  - **主要内容**：`set` (带/不带过期时间)，`get` (带类型转换)，`delete` 方法。
- `src/main/java/com/shimh/config/RedisConfig.java`：配置 Redis 客户端和缓存管理器。
  - **意义**：配置 Spring 与 Redis 集成所需的 Bean (`RedisTemplate`，`RedisManager`)。
  - **主要内容**：配置 `RedisTemplate` (设置连接工厂和 Key 序列化器)，创建自定义的 `RedisManager` Bean。
  - **依赖**：`RedisManager.java`。

    - **注意**：Value 序列化器被注释，建议启用 JSON 序列化。
- `src/main/java/com/shimh/oauth/OAuthSessionDAO.java`：Redis Session 数据访问对象。
    - **意义**：实现 Session 的 Redis 持久化。
    - **主要内容**：重写 Shiro `SessionDAO` 方法，调用 `RedisManager` 进行 Session 存取。
    - **依赖**：`RedisManager.java`。
- `src/main/java/com/shimh/oauth/OAuthSessionManager.java`：自定义 Session 管理器。
    - **意义**：从请求头获取 Token 作为 Session ID。
    - **主要内容**：重写 `getSessionId` 方法，从 `Oauth-Token` 请求头读取 Session ID。
- `src/main/java/com/shimh/oauth/OAuthRealm.java`：自定义 Shiro Realm。
    - **意义**：负责用户认证和授权的具体逻辑。
    - **主要内容**：实现 `doGetAuthenticationInfo`（通过 `UserService` 获取用户密码和盐值进行认证）和 `doGetAuthorizationInfo`（通过 `UserService` 获取用户角色进行授权，目前仅简单处理 admin 角色）。
    - **依赖**：`UserService.java`，`User.java`，`Base.java`，`UserStatus.java`。

TODO

- `src/main/java/com/shimh/config/ShiroConfig.java`：配置 Shiro 的 Realm, SessionManager, SecurityManager 等安全相关组件。
    - **意义**：Shiro 框架核心配置。
    - **主要内容**：配置 `SecurityManager`，`ShiroFilterFactoryBean`（URL过滤规则，目前为 all anon），`HashedCredentialsMatcher`，自定义 `OAuthRealm`，`OAuthSessionManager`，`OAuthSessionDAO` 等 Bean。
    - **依赖**：`OAuthRealm.java`，`OAuthSessionManager.java`，`OAuthSessionDAO.java`。
    - **注意**：URL 过滤规则当前配置为全部允许匿名访问，Shiro 访问控制未生效。
- `src/main/java/com/shimh/common/interceptor/ClearTokenInteceptor.java`：Session 超时检查拦截器。
    - **意义**：检查请求 Token 对应的 Session 是否有效，若无效则通知前端清除 Token。
    - **主要内容**：在 `preHandle` 中通过 `RedisManager` 检查 Session，若无效则在 Response Header 中添加超时标记。
    - **依赖**：`RedisManager.java`，`OAuthSessionManager.java`（常量）。
- `src/main/java/com/shimh/config/WebMvcConfig.java`：配置 MVC 相关，如跨域、拦截器等（如果需要）。
    - **意义**：配置 Spring MVC，集成 Fastjson，注册拦截器。
    - **主要内容**：配置 `FastJsonHttpMessageConverter` 进行 JSON 序列化，注册 `ClearTokenInteceptor` 拦截器。
    - **依赖**：`ClearTokenInteceptor.java`。
- `src/main/resources/log4j2-spring.xml`：Log4j2 配置文件。
    - **意义**：配置日志输出格式、级别和目的地。
    - **主要内容**：定义了控制台（`Console`）和多种文件（`Log`，`RollingRandomAccessFileWarn`，`RollingFileError`）Appender，但根 Logger（`root`）当前只配置了输出到 `Console`，级别为 `info`。

    - **依赖：** `application.properties` （指定配置文件路径）。
    - **注意：** 文件日志 Appender 未启用。
- `src/main/java/com/shimh/common/entity/BaseEntity.java` ：实体基类。
    - **意义：** 提供所有实体的公共 `id` 主键及 `equals` / `hashCode` 实现。
    - **主要内容：** `@MappedSuperclass` ，包含自增的 `id` 字段。
- `src/main/java/com/shimh/entity/UserStatus.java` ：用户状态枚举。
    - **意义：** 定义用户状态常量。
    - **主要内容：** `normal` ， `blocked` 两种状态。
- `src/main/java/com/shimh/entity/User.java` ：定义用户实体类，包含 id，账号，密码，昵称，状态，创建时间，头像等字段，并添加 JPA 注解。
    - **意义：** 映射数据库 `sys_user` 表，定义用户数据结构。
    - **主要内容：** 继承 `BaseEntity` ，包含用户各属性字段（ `account` ， `password` ， `salt` ， `nickname` ， `email` ， `status` 等），使用 JPA 和校验注解。
    - **依赖：** `BaseEntity.java` ， `UserStatus.java` 。
- `src/main/java/com/shimh/common/constant/Base.java` ：系统常量。
    - **意义：** 定义全局常量。
    - **主要内容：** `ROLE_ADMIN` ， `CURRENT_USER` 等常量。
- `src/main/java/com/shimh/repository/UserRepository.java` ：用户数据访问接口。
    - **意义：** 提供用户数据的数据库操作方法（基于 Spring Data JPA）。
    - **主要内容：** 继承 `JpaRepository` ，定义 `findByAccount` 查询方法。
    - **依赖：** `User.java` 。
- `src/main/java/com/shimh/vo/UserVO.java` ：定义用户视图对象，用于接口返回，可能包含部分用户信息和权限信息。
- `src/main/java/com/shimh/service/UserService.java` ：定义用户服务接口。
    - **意义：** 声明用户相关业务操作方法。
    - **主要内容：** `findAll` ， `getUserByAccount` ， `getUserById` ， `saveUser` ， `updateUser` ， `deleteUserById` 等接口方法。
    - **依赖：** `User.java` 。
- `src/main/java/com/shimh/common/util/PasswordHelper.java` ：密码加密工具。
    - **意义：** 提供密码加密和盐值生成功能。
    - **主要内容：** 使用 Shiro 的 `SimpleHash` （MD5，迭代2次）和 `SecureRandomNumberGenerator` 对用户密码进行加盐加密。
    - **依赖：** `User.java` 。
- `src/main/java/com/shimh/service/impl/UserServiceImpl.java` ：实现 `UserService` 接口，处理用户相关的业务逻辑，如注册、登录、信息修改、权限管理等。
    - **意义：** 用户服务具体实现，处理业务逻辑。
    - **主要内容：** 实现用户 CRUD 操作，调用 `PasswordHelper` 加密密码，随机分配头像。更新逻辑不完整。
    - **依赖：** `UserService.java` ， `UserRepository.java` ， `User.java` ， `PasswordHelper.java` 。
- `src/main/java/com/shimh/common/constant/ResultCode.java` ：API 结果状态码枚举。
    - **意义：** 定义统一的 API 返回状态码及其消息。
    - **主要内容：** 包含成功、各种错误类型（参数、用户、业务、系统、数据、接口、权限等）的状态码。

- `src/main/java/com/shimh/common/result/Result.java` : API 结果封装类。
  - **意义**：统一 API 接口的返回格式。
  - **主要内容**：包含 `code` , `msg` , `data` 字段，提供 `success()` , `error()` 等静态工厂方法。
  - **依赖**： `ResultCode.java` 。
- `src/main/java/com/shimh/common/annotation/LogAnnotation.java` : 日志注解。
  - **意义**：用于标记需要记录日志的方法及其模块和操作。
  - **主要内容**：定义了 `@LogAnnotation` 注解，包含 `module` 和 `operation` 属性。
- `src/main/java/com/shimh/common/util/UserUtils.java` : 用户工具类。
  - **意义**：提供获取当前登录用户的便捷方法。
  - **主要内容**： `getCurrentUser()` 方法从 Shiro Session 中获取用户信息。
  - **依赖**： `Base.java` , `User.java` 。
  - **注意**：依赖于登录时将 User 对象存入 Session (key: "current_user")。
- `src/main/java/com/shimh/controller/UserController.java` : 定义用户相关的 RESTful API 接口，处理 HTTP 请求，调用 `UserService` 完成操作。
  - **意义**：处理用户管理的 API 请求。
  - **主要内容**：实现用户 CRUD 的 API 接口，使用了 `@RequiresRoles` 进行权限控制， `@LogAnnotation` 记录日志， `@FastJsonView` 控制输出。
  - **依赖**： `UserService.java` , `User.java` , `Base.java` , `Result.java` , `ResultCode.java` , `UserUtils.java` , `LogAnnotation.java` 。
  - **注意**： `getCurrentUser` 接口未显式添加认证注解。
- `src/main/java/com/shimh/controller/LoginController.java` : 登录/注册/登出控制器。
  - **意义**：处理认证相关的 API 请求。
  - **主要内容**：实现 `/login` , `/register` , `/logout` 接口。调用 Shiro `subject.login()` 和 `subject.logout()` 。登录成功后将 User 存入 Session 并返回 Session ID (Token)。 `/handleLogin` 用于处理 Shiro 的登录跳转 (返回 Session 超时)。
  - **依赖**： `UserService.java` , `User.java` , `Base.java` , `Result.java` , `ResultCode.java` , `OAuthSessionManager.java` , `LogAnnotation.java` 。
- `src/main/java/com/shimh/entity/Tag.java` : 文章标签实体。
  - **意义**：映射数据库 `me_tag` 表，定义标签数据结构。
  - **主要内容**：继承 `BaseEntity<Integer>` ，包含 `tagname` , `avatar` 字段。
  - **依赖**： `BaseEntity.java` 。
- `src/main/java/com/shimh/vo/TagVO.java` : 标签视图对象。
  - **意义**：用于 API 返回，在 Tag 基础上增加文章数字段。
  - **主要内容**：继承 `Tag` ，增加 `articles` 字段。
  - **依赖**： `Tag.java` 。
- `src/main/java/com/shimh/repository/wrapper/TagWrapper.java` : 标签数据包装接口。
  - **意义**：定义获取标签详细信息（含文章数）的方法。
  - **主要内容**： `findAllDetail()` , `getTagDetail()` 方法，返回 `TagVO` 。
  - **依赖**： `TagVO.java` 。
- `src/main/java/com/shimh/repository/TagRepository.java` : 标签数据访问接口。
  - **意义**：提供标签数据的数据库操作，包括获取热门标签。

    - **主要内容**：继承 `JpaRepository<Tag, Integer>` 和 `TagWrapper`，定义原生 SQL 查询 `listHotTagsByArticleUse`。
    - **依赖**：`Tag.java`，`TagWrapper.java`。
- `src/main/java/com/shimh/repository/impl/TagRepositoryImpl.java`：标签自定义查询实现。
    - **意义**：实现 `TagWrapper` 接口，提供获取标签详情（含文章数）的复杂查询逻辑。
    - **主要内容**：使用原生 SQL 和 Hibernate `SQLQuery` 实现 `findAllDetail` 和 `getTagDetail` 方法，结果映射到 `TagVO`。
    - **依赖**：`TagWrapper.java`，`TagVO.java`。
- `src/main/java/com/shimh/service/TagService.java`：标签服务接口。
    - **意义**：声明标签相关的业务操作方法。
    - **主要内容**：定义 CRUD、获取热门标签、获取标签详情（返回 Tag 或 TagVO）的接口方法。
    - **依赖**：`Tag.java`，`TagVO.java`。
- `src/main/java/com/shimh/service/impl/TagServiceImpl.java`：标签服务实现。
    - **意义**：实现标签业务逻辑。
    - **主要内容**：实现 `TagService` 接口方法，主要调用 `TagRepository` 完成操作。
    - **依赖**：`TagService.java`，`TagRepository.java`，`Tag.java`，`TagVO.java`。
- `src/main/java/com/shimh/controller/TagController.java`：标签控制器。
    - **意义**：处理标签管理的 API 请求。
    - **主要内容**：实现标签 CRUD、获取热门标签、获取标签详情（区分 Tag 和 TagVO）的 API 接口，使用 `@RequiresRoles` 进行权限控制。
    - **依赖**：`TagService.java`，`Tag.java`，`TagVO.java`，`Result.java`，`ResultCode.java`，`Base.java`，`LogAnnotation.java`。
- `src/main/java/com/shimh/entity/Category.java`：文章分类实体。
    - **意义**：映射数据库 `me_category` 表，定义分类数据结构。
    - **主要内容**：继承 `BaseEntity<Integer>`，包含 `categoryname`，`description`，`avatar` 字段。
    - **依赖**：`BaseEntity.java`。
- `src/main/java/com/shimh/repository/wrapper/CategoryWrapper.java`：分类数据包装接口。
    - **意义**：定义获取分类详细信息（含文章数）的方法。
    - **主要内容**：`findAllDetail()`，`getCategoryDetail()` 方法，返回 `CategoryVO`。
    - **依赖**：`Category.java`，`CategoryWrapper.java`。
- `src/main/java/com/shimh/repository/impl/CategoryRepositoryImpl.java`：分类自定义查询实现。
    - **意义**：实现 `CategoryWrapper` 接口，提供获取分类详情（含文章数）的复杂查询逻辑。
    - **主要内容**：使用原生 SQL 和 Hibernate `SQLQuery` 实现 `findAllDetail` 和 `getCategoryDetail` 方法，结果映射到 `CategoryVO`。
    - **依赖**：`CategoryWrapper.java`，`CategoryVO.java`。
- `src/main/java/com/shimh/vo/CategoryVO.java`：定义分类视图对象。
- `src/main/java/com/shimh/service/CategoryService.java`：分类服务接口。
    - **意义**：声明分类相关的业务操作方法。
    - **主要内容**：定义 CRUD、获取分类详情（返回 Category 或 CategoryVO）的接口方法。
    - **依赖**：`Category.java`，`CategoryVO.java`。

- `src/main/java/com/shimh/service/impl/CategoryServiceImpl.java`：分类服务实现。
  - **意义**：实现分类业务逻辑。
  - **主要内容**：实现 `CategoryService` 接口方法，主要调用 `CategoryRepository` 完成操作。
  - **依赖**：`CategoryService.java`，`CategoryRepository.java`，`Category.java`，`CategoryVO.java`。
- `src/main/java/com/shimh/controller/CategoryController.java`：分类控制器。
  - **意义**：处理分类管理的 API 请求。
  - **主要内容**：实现分类 CRUD、获取分类详情（区分 Category 和 CategoryVO）的 API 接口，使用 `@RequiresRoles` 进行权限控制。
  - **依赖**：`CategoryService.java`，`Category.java`，`CategoryVO.java`，`Result.java`，`ResultCode.java`，`Base.java`，`LogAnnotation.java`。
- `src/main/java/com/shimh/entity/ArticleBody.java`：文章内容实体。
  - **意义**：存储文章的大文本内容（Markdown 和 HTML），与文章主体分离。
  - **主要内容**：继承 `BaseEntity<Long>`，包含 `content` 和 `contentHtml` 字段（使用 `@Lob` 和懒加载）。
  - **依赖**：`BaseEntity.java`。
- `src/main/java/com/shimh/entity/Comment.java`：评论实体。
  - **意义**：映射数据库 `me_comment` 表，定义评论数据结构及关联。
  - **主要内容**：继承 `BaseEntity<Integer>`，包含 `content`，`createDate`，`level` 字段，以及与 `User`（author, toUser），`Article`，`Comment`（parent, childrens）的关联关系。
  - **依赖**：`BaseEntity.java`，`User.java`，`Article.java`。
- `src/main/java/com/shimh/entity/Article.java`：文章实体。
  - **意义**：映射数据库 `me_article` 表，定义文章核心数据及关联。
  - **主要内容**：继承 `BaseEntity<Integer>`，包含 `title`，`summary`，`commentCounts`，`viewCounts`，`weight`，`createDate` 字段，以及与 `User`（author），`ArticleBody`，`Category`，`Tag`（ManyToMany），`Comment`（OneToMany）的关联关系。
  - **依赖**：`BaseEntity.java`，`User.java`，`ArticleBody.java`，`Category.java`，`Tag.java`，`Comment.java`。
- `src/main/java/com/shimh/vo/PageVo.java`：分页参数视图对象。
  - **意义**：封装分页和排序请求参数。
  - **主要内容**：`pageNumber`，`pageSize`，`name`（排序字段?），`sort`（排序方向）。
- `src/main/java/com/shimh/vo/ArticleVo.java`：文章视图对象。
  - **意义**：用于 API 查询参数或结果封装。
  - **主要内容**：继承 `Article`，增加 `year`，`month`，`tagId`，`categoryId`，`count` 等字段，用于归档或条件查询。
  - **依赖**：`Article.java`。
- `src/main/java/com/shimh/repository/wrapper/ArticleWrapper.java`：文章数据包装接口。
  - **意义**：定义文章复杂查询接口（分页、条件查询、归档）。
  - **主要内容**：`listArticles`（分页/条件分页），`listArchives`（归档）方法签名。
  - **依赖**：`Article.java`，`ArticleVo.java`，`PageVo.java`。

- `src/main/java/com/shimh/repository/ArticleRepository.java`：文章数据访问接口。
  - **意义**：提供文章数据的数据库操作方法（基于 Spring Data JPA）。
  - **主要内容**：继承 `JpaRepository<Article, Integer>` 和 `ArticleWrapper`，定义按 Tag/Category 查询、按浏览/时间排序查询的方法。
  - **依赖**：`Article.java`，`Category.java`，`Tag.java`，`ArticleWrapper.java`。
- `src/main/java/com/shimh/repository/impl/ArticleRepositoryImpl.java`：文章自定义查询实现。
  - **意义**：实现 `ArticleWrapper` 接口，提供文章复杂查询逻辑（分页、条件、归档）。
  - **主要内容**：使用 HQL（分页/条件）和原生 SQL（归档）实现 `listArticles` 和 `listArchives` 方法，结果映射到 `Article` 或 `ArticleVo`。
  - **依赖**：`ArticleWrapper.java`，`Article.java`，`ArticleVo.java`，`PageVo.java`。
- `src/main/java/com/shimh/vo/ArticleVO.java`：定义文章列表和详情的视图对象。
- `src/main/java/com/shimh/service/ArticleService.java`：文章服务接口。
  - **意义**：声明文章相关的业务操作方法。
  - **主要内容**：定义文章的各种查询（分页、条件、ID、标签、分类、热门、最新、归档）和写操作（发布、保存、更新、删除）的接口方法。包含 `getArticleAndAddViews` 特殊方法。
  - **依赖**：`Article.java`，`ArticleVo.java`，`PageVo.java`，`Tag.java`。
- `src/main/java/com/shimh/service/impl/ArticleServiceImpl.java`：文章服务实现。
  - **意义**：实现文章业务逻辑。
  - **主要内容**：实现 `ArticleService` 接口方法，调用 `ArticleRepository` 完成数据库操作，处理作者、创建时间、权重、浏览量增加等业务细节。
  - **依赖**：`ArticleService.java`，`ArticleRepository.java`，`Article.java`，`ArticleVo.java`，`PageVo.java`，`Category.java`，`Tag.java`，`User.java`，`UserUtils.java`。
- `src/main/java/com/shimh/controller/ArticleController.java`：文章控制器。
  - **意义**：处理文章相关的 API 请求（CRUD、查询、归档等）。
  - **主要内容**：实现文章相关 API 接口，大量使用 `@FastJsonView/Filter` 控制 JSON 输出，使用 `@RequiresAuthentication` 或 `@RequiresRoles` 进行权限控制。
  - **依赖**：`ArticleService.java`，`TagService.java`，`Article.java`，`ArticleBody.java`，`ArticleVo.java`，`PageVo.java`，`Tag.java`，`User.java`，`Result.java`，`ResultCode.java`，`Base.java`，`LogAnnotation.java`。
- `src/main/java/com/shimh/repository/CommentRepository.java`：评论数据访问接口。
  - **意义**：提供评论数据的数据库操作方法（基于 Spring Data JPA）。
  - **主要内容**：继承 `JpaRepository<Comment, Integer>`，定义按文章和层级查询评论的方法。
  - **依赖**：`Comment.java`，`Article.java`。
- `src/main/java/com/shimh/service/CommentService.java`：评论服务接口。
  - **意义**：声明评论相关的业务操作方法。
  - **主要内容**：定义 CRUD、按文章查询评论的方法。包含特殊方法用于在增删评论时同步更新文章评论数。
  - **依赖**：`Comment.java`，`Article.java`。
- `src/main/java/com/shimh/service/impl/CommentServiceImpl.java`：评论服务实现。
  - **意义**：实现评论业务逻辑。

    - **主要内容**：实现 `CommentService` 接口方法，处理评论 CRUD 和按文章查询，并在增删时同步更新文章评论数，自动设置作者、时间、层级。
    - **依赖**：`CommentService.java`，`CommentRepository.java`，`ArticleRepository.java`，`Comment.java`，`Article.java`，`User.java`，`UserUtils.java`。
- `src/main/java/com/shimh/controller/CommentController.java`：评论控制器。
    - **意义**：处理评论相关的 API 请求。
    - **主要内容**：实现评论 CRUD、按文章查询评论的 API 接口。区分了是否同步更新文章评论数的操作。使用 `@FastJsonView/Filter` 控制 JSON 输出，使用 `@RequiresAuthentication` 进行权限控制（部分接口权限可能需细化）。
    - **依赖**：`CommentService.java`，`Comment.java`，`Article.java`，`User.java`，`Result.java`，`ResultCode.java`，`LogAnnotation.java`。
- `src/main/java/com/shimh/common/util/StringUtils.java`：字符串工具类。
    - **意义**：提供字符串操作的辅助方法。
    - **主要内容**：`isEmpty` 方法。
- `src/main/java/com/shimh/common/util/HttpContextUtils.java`：HTTP 上下文工具类。
    - **意义**：获取当前 `HttpServletRequest` 对象。
    - **主要内容**：`getHttpServletRequest()` 方法。
- `src/main/java/com/shimh/common/util/IpUtils.java`：IP 地址工具类。
    - **意义**：获取客户端真实 IP 地址（考虑反向代理）。
    - **主要内容**：`getIpAddr` 方法。
    - **依赖**：`HttpContextUtils.java`，`StringUtils.java`。
- `src/main/java/com/shimh/entity/Log.java`：操作日志实体。
    - **意义**：映射数据库 `sys_log` 表，定义日志数据结构。
    - **主要内容**：继承 `BaseEntity<Integer>`，包含 `userId`，`nickname`，`module`，`operation`，`method`，`params`，`time`，`ip`，`createDate` 字段。
    - **依赖**：`BaseEntity.java`。
- `src/main/java/com/shimh/repository/LogRepository.java`：日志数据访问接口。
    - **意义**：提供日志数据的数据库操作方法（基于 Spring Data JPA）。
    - **主要内容**：继承 `JpaRepository<Log, Integer>`。
    - **依赖**：`Log.java`。
- `src/main/java/com/shimh/service/LogService.java`：日志服务接口。
    - **意义**：声明日志保存的业务方法。
    - **主要内容**：`saveLog` 接口方法。
    - **依赖**：`Log.java`。
- `src/main/java/com/shimh/service/impl/LogServiceImpl.java`：日志服务实现。
    - **意义**：实现日志保存逻辑。
    - **主要内容**：实现 `LogService` 接口，调用 `LogRepository` 保存日志。
    - **依赖**：`LogService.java`，`Log.java`，`LogRepository.java`。
- `src/main/java/com/shimh/common/aspect/LogAspect.java`：日志记录切面。
    - **意义**：使用 AOP 自动记录带有 `@LogAnnotation` 注解的方法的操作日志。
    - **主要内容**：定义切点拦截 `@LogAnnotation`，使用 `@Around` 通知记录方法执行时间、IP、模块、操作等信息，并调用 `LogService` 保存。

    - **依赖：** `LogService.java`，`Log.java`，`LogAnnotation.java`，`User.java`，`HttpContextUtils.java`，`IpUtils.java`，`UserUtils.java`。
    - **注意：** 请求参数和用户信息记录逻辑被注释。
- `src/main/java/com/shimh/BlogApiApplication.java`：Spring Boot 启动类。
    - **意义：** Spring Boot 应用入口。
    - **主要内容：** `main` 方法和 `@SpringBootApplication` 注解。
- `src/test/...`：编写单元测试和集成测试。