项目技术文档（framework 模块）

> 运行环境：Windows，Java 21（JDK 21），Spring Boot 4.0.3

概述

- 平台型后端服务，采用 `Spring Boot 4.0.3`，区分 `dev` / `server` 两种运行配置。
- 数据访问：`MyBatis-Plus 3.5.16` + `MyBatis 3.5.16`；连接池：`Druid 1.2.20`。
- 任务调度：`Quartz 2.3.2`；API 文档：`springdoc-openapi`（Swagger UI）。
- 缓存与基础设施：`Redis`、`WebSocket`、`Actuator`、验证码（`easy-captcha`）、加密（`jasypt`）、`ip2region`、`hutool` 等。
- 打包方式：可执行 `JAR`（已启用 `spring-boot-maven-plugin` 的 `repackage`）。默认开发端口 `8093`（`server` 可使用 `8090`）。

目录结构

- 仓库根（当前工作目录）：`D:\Trae_space\WholeProcessPlatform`
- 主要结构：

```
WholeProcessPlatform/
├── frontend/                 # 前端源码（Vite + TypeScript + Tailwind）
└── backend/                  # 后端服务（Spring Boot 4.0.3）
    ├── pom.xml               # Maven 构建管理
    └── src/
        ├── main/
        |   ├── java/         # 业务代码（入口类在 com.yfd.platform.*）
        |   └── resources/    # 配置与静态资源（static、配置 YAML、日志）
        └── test/
            └── java/         # 测试代码
```

快速开始

- 前置要求：
  - 安装 `JDK 21`、`Maven 3.6.3+`（建议 `3.9+`）、`Git`、`Node.js 18+`（前端构建）。
  - Windows 终端建议执行 `chcp 65001`，确保 UTF-8 编码输出。
- 构建后端：
  - `cd backend && mvn clean package -DskipTests`
- 本地运行（dev）：
  - `cd backend && java -jar target/platform-1.0.jar --spring.profiles.active=dev`
- 运行（server）：
  - `cd backend && java -jar target/platform-1.0.jar --spring.profiles.active=server`
- API 文档：
  - 访问 `http://localhost:8093/swagger-ui.html`（以实际配置为准）

前端集成

- 自动构建：Maven 在 `generate-resources` 阶段调用 `npm install` 与 `npm run build:mvn`，产物输出到 `frontend/dist`。
- 静态资源复制：可通过 `maven-resources-plugin` 将 `frontend/dist` 复制至 `src/main/resources/static`，如需开启请将 POM 中该插件的 `<skip>` 设置为 `false` 或按需配置 Profile。
- 单独构建：在 `framework/frontend` 目录手动执行 `npm install && npm run build`，然后复制 `dist/` 到后端静态目录。

配置说明

- Profile 切换：通过 `--spring.profiles.active=<dev|server>` 激活环境。
- 关键属性：
  - `file-space.system`：文件根路径，需在激活的 profile 中配置。
  - `spring.datasource.druid.*`：数据库连接与池化参数。
  - `server.port`：端口（`dev` 默认 8093，`server` 可使用 8090）。
- 编码与 JVM 参数：
  - 统一 JVM 编码：`-Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8`（POM 已配置）。
  - PowerShell 输出建议设为 UTF-8：`chcp 65001`。
- Maven 本地仓库：
  - 为避免某些 IDE/沙箱环境写入受限，后端已配置 Maven 本地仓库为项目内目录（`backend/.m2repo`），配置文件：`backend/.mvn/maven.config`。

依赖与版本

- Spring：`spring-boot-starter-web`、`security`、`cache`、`websocket`、`actuator`（Spring Boot 4.0.3 / Spring Framework 7）。
- 数据访问：`mybatis-spring-boot-starter 4.0.1`、`mybatis-plus-spring-boot4-starter 3.5.16`、`druid 1.2.20`、`mysql-connector-j`、`sqlite-jdbc`。
- 工具与增强：`hutool-all`、`poi` / `poi-ooxml`、`fastjson`、`freemarker`、`jsoup`、`jasypt`、`ip2region`、`easy-captcha`、`UserAgentUtils`、`nashorn-core`。
- 文档：`springdoc-openapi-starter-webmvc-ui 3.0.2`（Swagger UI）。
- 构建管控：`maven-enforcer-plugin`（JDK≥17、Maven≥3.6.3）。

数据库配置示例

- MySQL 连接：

```yaml
spring:
  datasource:
    druid:
      master:
        url: jdbc:mysql://<host>:3306/<db>?
useUnicode=true&characterEncoding=UTF8&rewriteBatchedStatements=true&useSSL=false&allowPubli
cKeyRetrieval=true
        username: <user>
        password: <password>
```

- 授权建议：

```sql
CREATE USER 'appuser'@'%' IDENTIFIED BY 'StrongPassword!';
GRANT ALL PRIVILEGES ON <db>.* TO 'appuser'@'%';
FLUSH PRIVILEGES;
```

- Druid 健壮性：

```yaml
spring:
  datasource:
    druid:
      initial-size: 0
      test-on-borrow: false
      test-while-idle: true
      validation-query: SELECT 1
```

日志与监控

- 日志：Logback（UTF-8 输出），推荐格式：`%d [%thread] %-5level %logger{50} - %msg%n`。
- 监控：启用 Actuator，健康检查路径：`/actuator/health`。
- 建议接入集中日志与指标采集（ELK / Vector），监控数据库与线程池（Druid / Quartz）。

定时任务

- 默认 `RAMJobStore`（非集群、内存存储）。
- 如需持久化与集群，改用 `JdbcJobStore` 并配置数据源与表结构。

安全与鉴权

- 采用 JWT 进行鉴权（如 `jwtAuthenticationTokenFilter`，按实际代码启用）。
- 敏感配置（`jwt.secret`、数据库密码）通过环境变量或外部密钥管理。
- 生产环境建议启用 HTTPS 并使用强密钥。

Docker 部署

- 端口：默认暴露 `8093`（dev）；可在 `server` profile 使用 `8090`。
- 基本流程：

    - 构建镜像：`docker build -t projectframework-app:latest .`
    - 运行（开发环境）：`docker run -d --name platform-app -p 8093:8093 -e SPRING_PROFILES_ACTIVE=dev projectframework-app:latest`
    - 运行（服务器环境）：`docker run -d --name platform-app -p 8090:8090 -e SPRING_PROFILES_ACTIVE=server projectframework-app:latest`
- 文件空间挂载示例：`-v D:/data/file-space:/data/file-space -e FILE_SPACE_SYSTEM=/data/file-space`

Git 使用（框架分支）

- 远程仓库：根据实际仓库地址配置（示例：`http://121.37.111.42:3000/ThbTech/JavaProjectRepo.git`）。
- 分支建议：`main-framework`（稳定分支）、`develop-framework`（开发分支）。
- 常用命令：`git pull`、`git add .`、`git commit -m "<message>"`、`git push`。
- 提交署名：`git config user.name "<Your Name>"`，`git config user.email "<your@email>"`。

CI/CD 建议

- CI 阶段：`mvn -B -DskipTests clean package`，配合单元测试与安全扫描（依赖检查、代码质量）。
- CD 阶段：推送镜像到私有仓库，使用环境变量注入敏感信息。

常见问题

- 中文乱码：执行 `chcp 65001`，确保 `logback-spring.xml` 使用 UTF-8。
- `NoSuchMethodError`（MyBatis）：升级到 `MyBatis 3.5.16+` 并与 MP 版本匹配。
- 数据库 `Access denied`：校验账户密码与远程授权；必要时新建业务账户并放开 `3306`。
- 端口占用/启动失败：检查 `server.port` 与冲突端口；查看应用日志定位根因。
- 前端构建问题：如遇 npm 依赖警告或复制冲突，可先独立构建并手动复制 `dist/`。

变更日志（模板）

- `feat:` 新增功能说明
- `fix:` 缺陷修复说明
- `docs:` 文档更新说明
- `refactor:` 重构说明
- `perf:` 性能优化说明

如需扩展专题文档（接口规范、部署拓扑、参数字典等），请在 `docs/` 目录继续维护并与版本管理同步。