



Java 代码规范

文件编号

IT-05-R01

版本号

2.2

Java 代码规范

编号	IT-05-R01
版本	2.2
状态	已发布
作者	李光明
日期	2022 年 12 月 2 日

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

修订历史

版本	状态	日期	修订人	审批人	修订内容
1.0	起草	2021/1/13	丁轶		初始版本。
2.0	已评审	2021/12/23	周易 李光明		第 6,7,8,9 章 代码规约调整。
2.1	已发布	2022/3/31	李光明 丁轶	肖伟、周易、 研发组长	第 9 章, Sonar 规则修改, 第 16 章, 单元测试约定修改。
2.2	已发布	2022/12/2	李光明	肖伟、李林 谢阳、石林清 胡晓晴、孙燕燕	第 10 章, 不合规报告与跟进修改

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

目录

1. 目的和范围.....	4
2. 适用对象.....	4
3. 参考文件.....	4
4. 术语和缩略语.....	4
5. 角色与职责.....	5
6. Java 项目工程结构规范（新建项目按照 DDD 4 层结构进行编码）	5
7. IDE 设置与代码统一格式化.....	6
8. 开发命名规范.....	7
9. Sonar 静态扫描重要规则清单	8
10. 不合规报告与跟进	9
11. SQL 编码	9
12. 代码安全.....	10
13. OOP 公约.....	11
14. 异常处理.....	12
15. 日志	13
15.1 日志框架	13
15.2 日志输出级别	14
15.3 需要打印日志场景.....	14
16. 单元测试.....	14

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

1. 目的和范围

制订孚创 Java 项目的代码规范，明确 Sonar 的静态代码扫描规则，给代码评审（动态评审规则与评审模板）提供标准参考。

2. 适用对象

适用于孚创信息技术部 Java 项目的代码开发与代码质量控制相关管理活动。

3. 参考文件

无。

4. 术语和缩略语

术语或缩略语	解释
Sonar	一个用于代码质量管理的开源平台，可以从七个维度检测代码质量。
UT: Unit Test	单元测试，由研发人员编写测试类及测试方法对关键逻辑测试。
SIT: System Integration Test	系统集成测试。
UAT: User Acceptance Test	用户接受测试，俗称:验收测试。
Jenkins	基于 Java 开发的一种持续集成工具。
Jacoco	一个开源的，通过接口测试实现代码覆盖率的工具。代码覆盖（Code coverage）是软件测试中的一种度量，描述源代码被测试的比例和程度。

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

5. 角色与职责

角色	职责
架构师	Java 代码规范的制订
研发负责人	Java 代码规范的评审, 监督执行
研发模块负责人	Java 代码规范的实施, KPI 考核, 意见改进
研发工程师	Java 代码规范的执行
质量管理工程师	Java 代码规范的监督, KPI 考核
测试工程师	提供接口测试覆盖率报告

6. Java 项目工程结构规范 (新建项目按照 DDD 4 层结构进行编码)

基于微服务 DDD 工程最佳实践分层, 父工程下创建子工程, 需要保证最基本 4 类子项目工程目录:

6.1 Api:

提供对外接口, 包括前端, 其它微服务调用接口, 传输对象转换。

Controller , Feign API , 前后端 DTO 及 与 DO 的转换。

6.2 Biz:

Application Service , 对 Domain 领域层方法集成, 编排, 及调用其它微服务接口, 这时也需要放 DO 与 DTO 的转换对象。

Remote , 对其它微服务 feign 接口引用 与 DTO。

Event , 包含 2 个目录: publish 和 subscribe, 事件发布与订阅, 建议存放所有订阅相关代码, 事件核心处理逻辑放在领域层的服务类里面。

6.3 Domain:

AggregateXX: 按照聚合的方式组织如下领域对象。

Domain Object: 领域对象: Entity , VO 实体 与 值对象。

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

Domain Service: 领域服务类。

Event, 事件实体对象 及 事件活动相关的简单业务逻辑, 提供给上层 Repository, 包含接口与接口实现, 简单 CRUD, DO 到 PO 的转换, 持久化, 类似 DAO。


6.4 Infra:

Config: 配置相关, 枚举, 静态类。

Utils: 平台, 开发框架, DataSource, File, 消息底层, 通用算法, 缓存, 安全, 总线, 网关 ...

7. IDE 设置与代码统一格式化

1、IDE 编码统一设定: utf-8。

2、推荐 IDEA 导入并使用统一孚创格式化文件  myFormatter.xml 至少各研发组内统一。

3、IDEA 工具统一设置, 本地开发环境添加 sonar 检查插件 SonarLint。

步骤一:

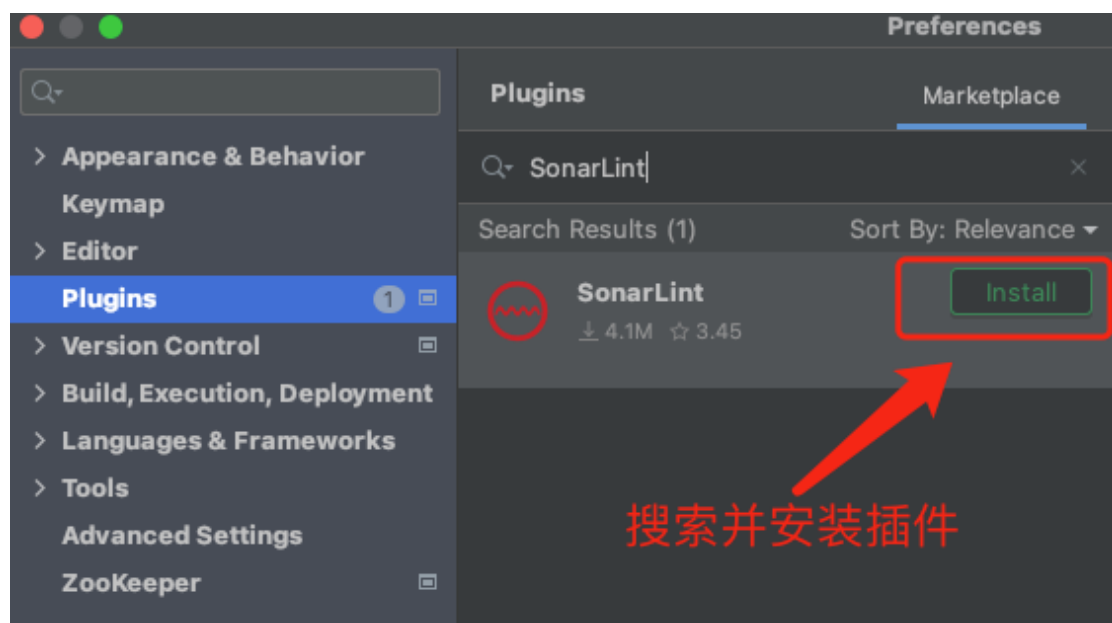


图 1 IDE 设置图 (一)

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

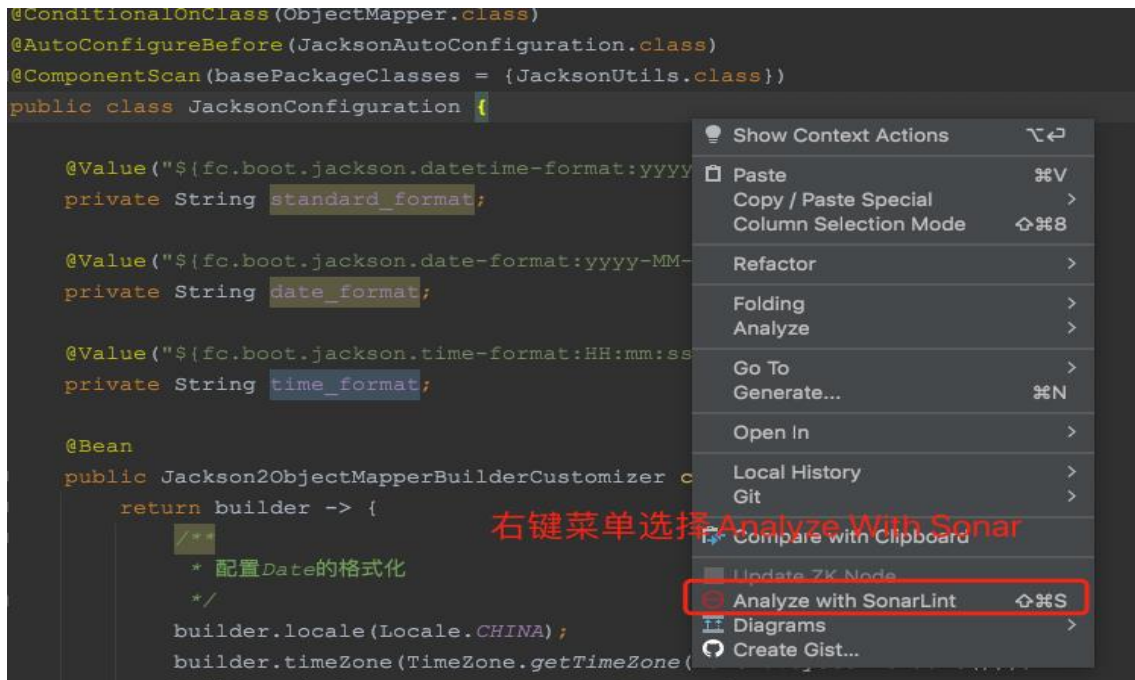


图 2 IDE 设置图 (二)

8. 开发命名规范

8.1 Gitlab 中对微服务名称约定:

全部小写英文字母，属于孚创业务后台应用以 fc 开头，非孚创后台应用直接以应用本身单词开头，如门店用 store 开头，经销商 dms 开头，单词间用“ - ”间隔符。

例：fc-mall-order，孚创后台应用，非孚创后台约定为 业务线 + 应用名。包名: 强制小写英文字母。

8.2 类名:

强制首字母大写，必须遵从驼峰形式（但以下情形例外：DO / BO / DTO / VO / AO）。

8.3 方法名、参数名、成员变量、局部变量:

首字母小写，必须遵从驼峰形式。

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

8.4 常量:

常量名称必须全部大写，单词间用下划线分隔。

8.5 枚举:

全部大写，必须有注释。

9. Sonar 静态扫描重要规则清单

规则编号	名称	说明	备注
1	https://sonar.fuchuang-auto.com/coding_rules?languages=java&open=java%3AS2095&severities=BLOCKER&types=BUG	Resources should be closed	
2	https://sonar.fuchuang-auto.com/coding_rules?languages=java&open=java%3AS2189&severities=BLOCKER&types=BUG	Loops should not be infinite	
3	https://sonar.fuchuang-auto.com/coding_rules?languages=java&open=java%3AS5547&severities=CRITICAL&types=VULNERABILITY	Cipher algorithms should be robust	
4	https://sonar.fuchuang-auto.com/coding_rules?languages=java&open=java%3AS4423&severities=CRITICAL&types=VULNERABILITY	Weak SSL/TLS protocols should not be used	
5	https://sonar.fuchuang-auto.com/coding_rules?languages=java&open=java%3AS1143&severities=CRITICAL&types=BUG	Jump statements should not occur in "finally" blocks	
6	https://sonar.fuchuang-auto.com/coding_rules?languages=java&open=java%3AS4602	"@SpringBootApplication" and "@ComponentScan" should not be	

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

		used in the default package	

以上是当前已调整的主要规则清单，后面不断完善添加规则项目。

主要是漏洞和 bug 类型，严重程度关注阻断和严重的问题。

10. 不合规报告与跟进

10.1 Sonar 统计报告

1、每次 lpt 和 stg 环境构建都触发扫描，并在 IT 管理平台的应用列表中展示相应的问题。有阻断和严重的问题，需尽快解决。

2、每日对所有后端应用扫描一次，Sonar 上配置应用归属邮件组，报告结果发送对应的开发负责人，督促当周内对阻断和严重问题进行整改。

3、每月汇总所有后端应用的扫描结果并生成月度报告，发送给开发负责人和 QA。

10.2 Code Review

在 WIKI 上搜索 IT-05-T03_CodeReviewChecklist 获取代码评审检查模板，代码评审会议参考代码评审检查表，进行检查并记录问题与跟进人，会议纪要需含 Check List 问题清单便于后续跟进。

11. SQL 编码

1、数据库表命名补充

功能设计期间明确包含物理删除功能的表，预见后期数据有维护需要（物理删除）的表，新建表的表名加后缀 下划线 ‘_’ + wisc。

2、研发人员在 Yearning 提交调整工单时一旦明确包含：

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

delete , alter table 操作, 需要勾选通知到数据团队的选项, 同时运维会增加判断逻辑 (单据内容含 delete , alter table) , 自动发送邮件通知到数据组此工单处理状态。

3、禁止使用 INSERT INTO t_xxx VALUES(xxx), 必须显式指定插入的列属性, 避免表结构变动导致数据出错。

4、禁止 select *。

5、已超过或将来可能超过 10 万行数据的表, 查询语句必须走到索引, 不允许全表扫描。

6、主库不允许出现超过 1 秒的慢 SQL, 从库不允许超过 30 秒。

7、主库上尽量不要有 JOIN 语句, 确实无法避免的, 禁止超过两个以上表的 JOIN 操作。

8、禁止不同类型的字段进行比较, 避免隐式转换, 必要时修改字段类型。

9、禁止单条 SQL 语句同时更新多张表。

10、禁止使用左模糊和全模糊, 必要时考虑用 ES。

11、禁止使用存储过程和视图。

12、禁止字符串直接拼接 SQL, 使用参数绑定, 防止 SQL 注入。

13、本地事务块中避免包含远程调用和磁盘 IO 等耗时较长的操作, 防止数据库事务和连接被长时间 hold 住。

12. 代码安全

1、避免硬编码敏感数据, 禁止将用户密码和手机号, 身份证明文输出。

2、禁止暴露不需要使用的接口到网关。

3、后台管理服务强制做用户权限校验。

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

4、使用自增长主键作为查询条件的情况必须做数据权限校验。

13.OOP 公约

1、【强制】避免通过一个类的对象引用访问此类的静态变量或静态方法，无谓增加编译器解析成本，直接用类名来访问即可。

2、【强制】所有静态方法直接调用，不要 new 对象.静态方法(). 甚至工具类就不应该写 public 的构造函数，或者写一个 private 的构造函数避免其它人通过构造函数 new 对象来调用工具方法。

3、【强制】所有的覆写方法，必须加@Override 注解。(原来实现类 不加注解编译也不会报错)。

反例：getObject()与 get0bject()的问题。一个是字母的 O，一个是数字的 0，加 @Override 可以准确判断是否覆盖成功。另外，如果在抽象类中对方法签名进行修改，其实实现类会马上编译报错。

4、【强制】不能使用过时的类或方法。

5、【强制】Object 的 equals 方法容易抛空指针异常，应使用常量或确定有值的对象来调用 equals。

6、【强制】关于基本数据类型与包装数据类型的使用标准如下：

1) 所有的 POJO 类属性必须使用包装数据类型。(防止数据库中返回 null 值而抛异常)。

2) RPC 方法的返回值和参数必须使用包装数据类型。(防止远程调用中返回 null 值而抛异常)。

3) 所有的局部变量【推荐】使用基本数据类型。

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

说明：POJO 类属性没有初始值时需提醒使用者在需要使用时，必须自己显式地进行赋值，任何 NPE 问题，或者入库检查，都由使用者来保证。

7、【强制】禁止在循环中执行耗时的操作，如在循环中执行 SQL 语句/调用外部服务等。

8、【强制】需要多次使用的可复用对象将对象单独定义，禁止链式方式调取不同属性，代码更简介，如：

```
String name = userService.getUser(id).getName();
```

```
Long deptId = userService.getUser(id).getDepId();
```

替换为：

```
User user = userService.getUser(id);
```

```
String name = user.getName(), ....
```

9、【推荐】能尽早返回的一定要尽早返回，先判断不成立的条件，尽早返回。

10、【推荐】不要出现大面积的注释代码，无用代码，有则删除。

11、【推荐】代码注释尽量详细，越详细越好，方便他人也方便自己。

12、【推荐】Lombok 简化代码臃肿。

14. 异常处理

1、【强制】调用外部服务等可能异常的代码块，用 try/catch 代码块捕获并在 catch 中记录异常跟踪日志及业务逻辑处理，记录详细的出入参。

2、【强制】禁止吞掉异常信息。

3、【强制】禁止 catch 里不做任何记录和处理，吞掉异常及其堆栈信息。

4、禁止：

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

`logger.error(“XXX 操作异常”)` 或 `logger.error(“XXX 操作异常” +
e.printStackTrace())`

正确: `logger.error("XXX 操作异常", e)`

15. 日志

15.1 日志框架

- 1、强制使用 slf4j 接口+logback 实现组件，禁止使用 log4j。
- 2、禁止 `system.out`, `excepton.printStackTrace()` 输出，这种会输出到控制台，不便于查询。
- 3、应用中统一使用 SLF4J 中的 API， 不要使用日志系统（Log4j、Logback）中的 API，否则各日志输出将无法统一。

- 4、日志输出使用占位符的方式， 避免字符串拼接， 提升性能。

//反例:

```
logger.info("接收到支付请求:" + request);
```

// 正例:

```
logger.info("接收到支付请求: {}", request);
```

- 5、打印异常堆栈不要用字符串拼接，字符串拼接方式输出的是异常对象的 `toString()` 方法返回的信息，一般为异常的 `message` 信息。

//正例

```
logger.error("支付请求发生系统异常", e);
```

//反例

```
logger.error("支付请求发生系统异常" + e);
```

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

15.2 日志输出级别

DEBUG: 线下环境用于调试目的的日志，现有调试和监控手段已比较多，建议少用，注意生产环境禁止输出 debug 日志。

INFO: 为了问题定位排查而输出的日志，避免同一信息在不同层次重复输出。

WARN: 发生的问题不影响系统正常运行，需要引起开发重视的可输出 warn 日志。

ERROR: 需要立即关注或解决的问题输出为 error 日志，如网络调用异常，目前 error 日志全部同步到 sentry 平台，并通过短信和邮件报警。

15.3 需要打印日志场景

系统边界处: 对外接口的请求和响应，调用外部接口的请求和响应。

发生业务异常时: 如发现外部请求非法。

捕获到系统异常时: 如网络异常等。

对于非预期的条件，尽量增加 else 记录跟踪日志。

禁止通过 `System.*.out()` 打印日志（单元测试例外）。

16. 单元测试

鼓励开发人员给核心、重要的功能添加 JUNIT 单元测试类。

鼓励开发人员给重要 Restful 接口提供 mock.http 文件。

编写单元测试的准入标准：

1. 发生线上问题，根本原因是代码实现逻辑缺陷。
2. DDD 改造后，处于核心域的应用的相关逻辑，定性为被重要依赖，意味着一旦发生错误将产生严重后果（待标准完善）。
3. 明确是接口自动化测试及回归测试很难覆盖的代码逻辑（待标准完善）。

	Java 代码规范	文件编号	IT-05-R01
		版本号	2.2

单元测试工具使用 IDEA 插件 JUnitGeneratorV2.0，UT 框架是 Junit 5。

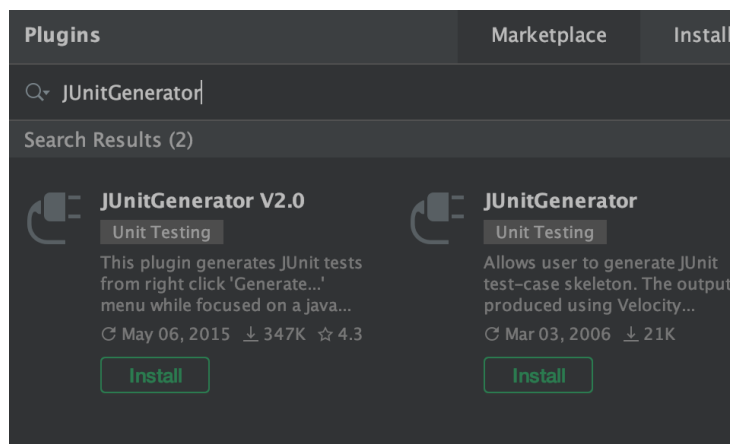


图 3 展示图