**数据化运营平台系统架构**

|  |  |
| --- | --- |
| 版本 | 1.0.0 |
| 作者 | 张海 |
| 最后更新日期 | 2017.05.01 |

# 目录

[目录 1](#_Toc481588278)

[1.项目介绍 2](#_Toc481588279)

[1.1.项目说明 2](#_Toc481588280)

[1.2.系统架构 2](#_Toc481588281)

[1.3.组织结构 3](#_Toc481588282)

[1.4.依赖关系 4](#_Toc481588283)

[1.5.技术选型 4](#_Toc481588284)

[2.开发指南 5](#_Toc481588285)

[2.1.开发环境 5](#_Toc481588286)

[2.2.开发工具 5](#_Toc481588287)

[2.3.部署方式 5](#_Toc481588288)

[2.4.拓扑图 6](#_Toc481588289)

[3.框架规范约定 6](#_Toc481588290)

[应用分层 7](#_Toc481588291)

[约定优于配置 8](#_Toc481588292)

[3.1.命名规约 8](#_Toc481588293)

[3.2.常量定义 10](#_Toc481588294)

[3.3.格式规约 10](#_Toc481588295)

[3.4. ORM 规约 11](#_Toc481588296)

[4.Dubbo在线文档 11](#_Toc481588297)

# 1.项目介绍

## 1.1.项目说明

基于Spring+SpringMVC+Mybatis+Dubbo分布式开发系统架构。

## 1.2.系统架构

对外暴露统一规范的接口



## 1.3.组织结构

**portal**

│

├─**doc 项目文档**

│

├─**portal-common 公共模块**

│ ├─db 多数据源

│ ├─model 实体类

│ └─util 工具类

│

├─**portal-dao 数据持久层模块**

│ ├─mapper 持久化

│ └─mapper.xml mybatise 配置文件

│

├─**portal-api-service-parent RPC接口服务模块（暴露成RPC接口）**

│ ├─**portal-api-interface** rpc服务接口包**模块**

│ └─**portal-api-service** rpc 业务逻辑**模块**

│

├─**portal-admin-service-parent 后台管理服务模块（不暴露成RPC接口）**

│ ├─**portal-api-interface** 后台管理服务接口包**模块**

│ └─**portal-api-service** 后台管理业务逻辑**模块**

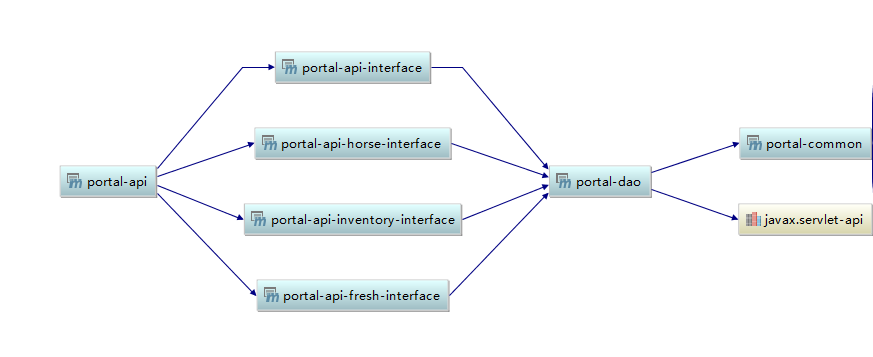
│

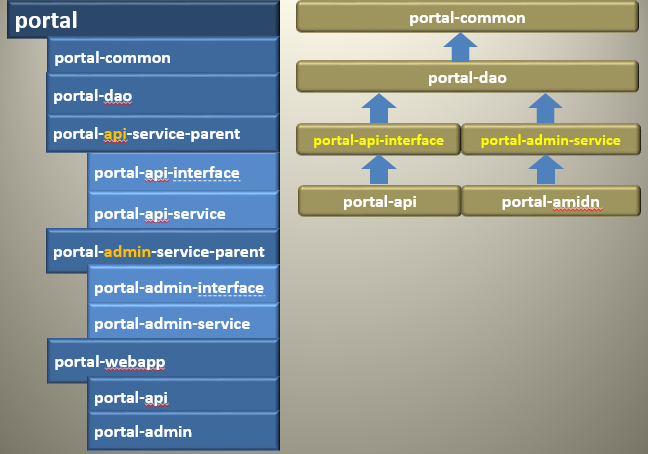
├─**portal-webapp webapp应用模块**

│ ├─**portal-api** api接口总线系统**模块**

│ └─**portal-admin** 后台管理系统**模块**

## 1.4.依赖关系





## 1.5.技术选型

|  |  |
| --- | --- |
| **技术** | **名称** |
| Spring Framework | 容器 |
| SpringMVC | MVC框架 |
| MyBatis | ORM框架 |
| Druid | 数据库连接池 |
| ZooKeeper | 分布式协调服务 |
| Dubbo | 分布式服务框架 |
| Redis | 分布式缓存数据库 |
| Log4J | 日志组件 |
| Swagger2 | 接口测试框架 |
| json | 数据序列化 |
| Jenkins | 持续集成工具 |
| Maven | 项目构建管理 |

# 2.开发指南

## 2.1.开发环境

Jdk7+

Mysql5.5+

Redis

Zookeeper

Dubbo-admin

## 2.2.开发工具

MySql: 数据库

Tomcat: 应用服务器

IntelliJ IDEA: 开发IDE

Navicat for MySQL: 数据库客户端

## 2.3.部署方式

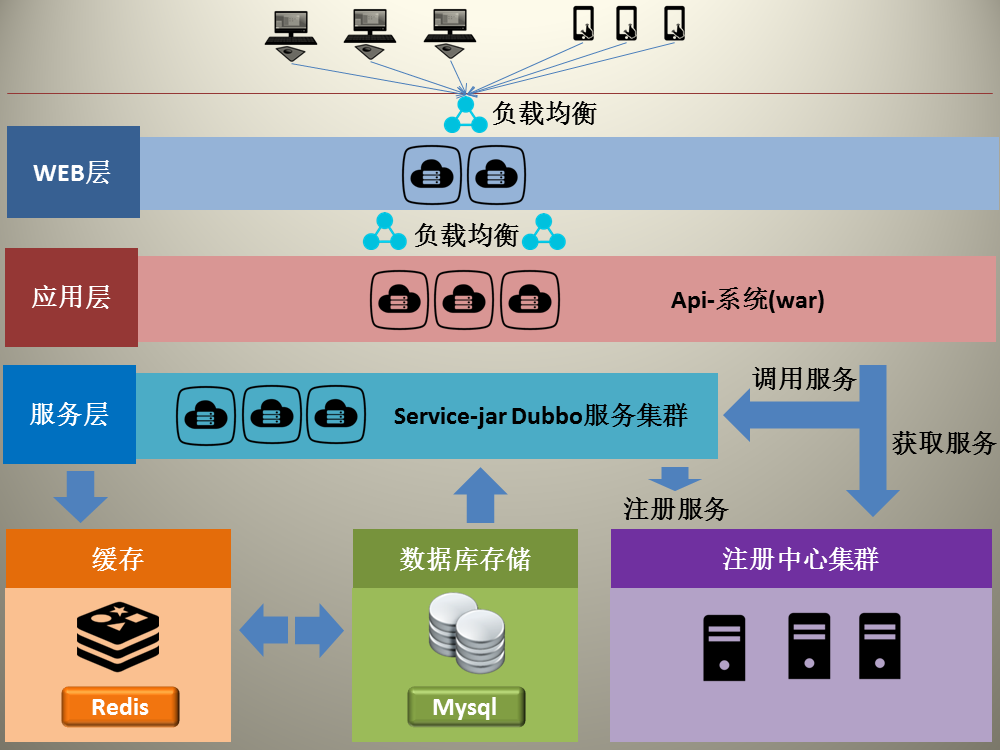
- 启动流程：优先api-service服务提供者，再启动其他webapp服务消费者

- 扩展流程：可扩展和拆分interface和service模块，可按微服务拆分或场景拆分

- war包项目：使用tomcat等web容器启动

- api-service服务提供者jar包：执行打包后的xxx-api-service.jar

## 2.4.拓扑图



# 3.框架规范约定

## 应用分层





## 约定优于配置

#不导入无用的包

#代码格式化

#业务逻辑块必须添加注释

# **service类**，需要在叫名`service`的包下，并以`Service`结尾，如`SysStartLogServiceImpl`

#**controller类**，需要在以`controller`结尾的包下，类名以Controller结尾，如`SysStartLogController.java`，并继承`BaseController`

#**mapper.xml**，需要在名叫`mapper`的包下，并以`Mapper.xml`结尾，如SysStartLogMapper.xml`

#**mapper接口**，需要在名叫`mapper`的包下，并以`Mapper`结尾，如`SysStartLogMapper.java`

#**model实体类**，需要在名叫`model`的包下，命名规则为数据表转驼峰规则，如`SysStartLog.java`

# **spring配置文件**，命名规则为`applicationContext-\*.xml`

**- 类名**：首字母大写驼峰规则；方法名：首字母小写驼峰规则；常量：全大写；变量：首字母小写驼峰规则，尽量非缩写

#**springmvc**配置加到对应模块的`springMVC-servlet.xml`文件里

#**配置文件**放到`src/main/resources`目录下

#**静态资源文件**放到`src/main/webapp/resources`目录下

# **jsp文件**，需要在`/WEB-INF/jsp`目录下

# `RequestMapping`和返回物理试图路径的url尽量写全路径，如：`@RequestMapping("/manage")`、`return "/manage/index"`

#`RequestMapping`指定method

# 模块命名为`项目`-`子项目`-`业务`，如`yonghui-portal-server`

# 数据表命名为：`子系统`\_`表`，如`partal\_user`

## 3.1.命名规约

1. 代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

反例： \_name / \_\_name / $Object / name\_ / name$ / Object$

2. 代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。 说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式 也要避免采用。

反例： DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3

正例： alibaba / taobao / youku / hangzhou 等国际通用的名称，可视同英文。

3. 类名使用 UpperCamelCase 风格，必须遵从驼峰形式，但以下情形例外：（领域模型 的相关命名）DO / BO / DTO / VO 等。 正例：MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion 反例：macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion

4. 方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从 驼峰形式。 正例： localValue / getHttpMessage() / inputUserId

5. 常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。 正例： MAX\_STOCK\_COUNT 反例： MAX\_COUNT

6. 抽象类命名使用 Abstract 或 Base 开头；异常类命名使用 Exception 结尾；测试类 命名以它要测试的类的名称开始，以 Test 结尾。

7. 中括号是数组类型的一部分，数组定义如下：String[] args;

反例：请勿使用 String args[]的方式来定义。

8. POJO 类中布尔类型的变量，都不要加 is，否则部分框架解析会引起序列化错误。

反例：定义为基本数据类型 boolean isSuccess；的属性，它的方法也是 isSuccess()，RPC 框架在反向解析的时候，“以为”对应的属性名称是 success，导致属性获取不到，进而抛出异 常。

9. 包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用 单数形式，但是类名如果有复数含义，类名可以使用复数形式。

正例： 应用工具类包名为 com.alibaba.open.util、类名为 MessageUtils（此规则参考 spring 的框架结构）

10. 杜绝完全不规范的缩写，避免望文不知义。

反例： AbstractClass“缩写”命名成 AbsClass；condition“缩写”命名成 condi，此类 随意缩写严重降低了代码的可阅读性。

11. 接口和实现类的命名有两套规则：

对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部 的实现类用 Impl 的后缀与接口区别。 正例：CacheServiceImpl 实现 CacheService 接口。

## 3.2.常量定义

1. 【强制】不允许出现任何魔法值（即未经定义的常量）直接出现在代码中。 反例： String key="Id#taobao\_"+tradeId； cache.put(key, value);

2. 【强制】long 或者 Long 初始赋值时，必须使用大写的 L，不能是小写的 l，小写容易跟数字 1 混淆，造成误解。 说明：Long a = 2l; 写的是数字的 21，还是 Long 型的 2?

## 3.3.格式规约

1. 【强制】大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；如果 是非空代码块则：

1） 左大括号前不换行。

2） 左大括号后换行。

3） 右大括号前换行。

4） 右大括号后还有 else 等代码则不换行；表示终止右大括号后必须换行。

2. 【强制】 左括号和后一个字符之间不出现空格；同样，右括号和前一个字符之间也不出现空 格。详见第 5 条下方正例提示。

3. 【强制】if/for/while/switch/do 等保留字与左右括号之间都必须加空格。

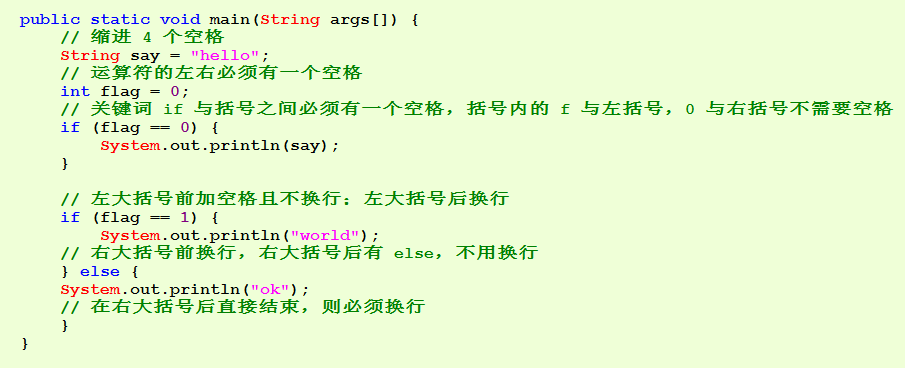
4. 【强制】任何运算符左右必须加一个空格。

说明：运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号、三目运行符等。

5. 【强制】缩进采用 4 个空格，禁止使用 tab 字符。

说明：如果使用 tab 缩进，必须设置 1 个 tab 为 4 个空格。IDEA 设置 tab 为 4 个空格时， 请勿勾选 Use tab character；而在 eclipse 中，必须勾选 insert spaces for tabs。

正例： （涉及 1-5 点）



6. 【强制】单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：

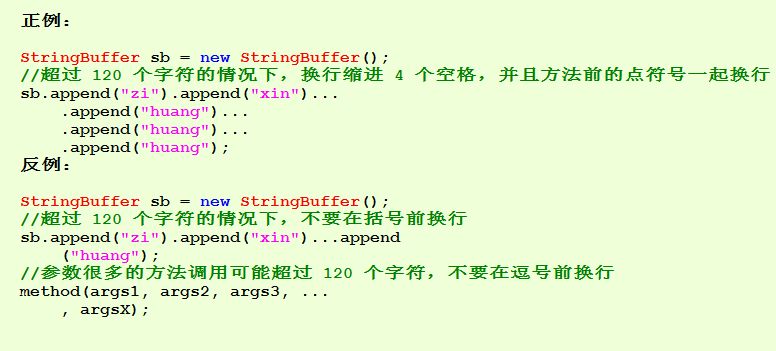
1） 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。

2） 运算符与下文一起换行。

3） 方法调用的点符号与下文一起换行。

4） 在多个参数超长，逗号后进行换行。

5） 在括号前不要换行，见反例。



7. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。

正例：下例中实参的"a",后边必须要有一个空格。 method("a", "b", "c");

8. 【强制】IDE 的 text file encoding 设置为 UTF-8;

IDE 中文件的换行符使用 Unix 格式， 不要使用 windows 格式。

## 3.4. ORM 规约

1. 【强制】在表查询中，一律不要使用 \* 作为查询的字段列表，需要哪些字段必须明确写明。

说明：

1）增加查询分析器解析成本。

2）增减字段容易与 resultMap 配置不一致。

2. 【强制】xml 配置中参数注意使用：#{}，#param# 不要使用${} 此种方式容易出现 SQL 注 入。

# 4.Dubbo在线文档

<http://dubbo.io/>