**微服务基础框架开发规范**

版本：V1.0

文档创建时间：2017年3月15日



**北京雅邦网络技术发展有限公司 YAPPAM.LTD.CO**

**文档变更记录**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 修改日期 | 修改的章节 | 修改类型 | 修改描述 | 修改人 | 审核人 | 版本号 |
| 2017/03/15 |  | A | 初始版本 | 杨辉 |  | 1.0 |
|  |  |  |  |  |  |  |

* 修改类型分为 **A** – ADDED（增加） **M** – MODIFIED（修改） **D** – DELETED（删除）

**目录**

[1 前言 5](#_Toc477507985)

[1.1 背景 5](#_Toc477507986)

[1.2 文档范围 5](#_Toc477507987)

[1.3 目标读者 5](#_Toc477507988)

[2 服务命名 5](#_Toc477507989)

[2.1 总体规范 5](#_Toc477507990)

[2.2 服务命名 5](#_Toc477507991)

[2.3 服务类型命名 5](#_Toc477507992)

[2.4 包名、类名 6](#_Toc477507993)

[3 微服务（SVN）结构 6](#_Toc477507994)

[3.1 SVN微服务地址 6](#_Toc477507995)

[3.2 SVN子结构命名 6](#_Toc477507996)

[3.2.1 基础服务根目录名 6](#_Toc477507997)

[3.2.2 后端服务根目录名 6](#_Toc477507998)

[3.2.3 前端服务根目录名 6](#_Toc477507999)

[3.2.4 公共服务根目录名 6](#_Toc477508000)

[3.2.5 产品根目录名 7](#_Toc477508001)

[3.2.6 项目根目录 7](#_Toc477508002)

[3.3 服务子目录 7](#_Toc477508003)

[4 服务设计 8](#_Toc477508004)

[4.1 数据库设计 8](#_Toc477508005)

[4.1.1 注意事项 8](#_Toc477508006)

[4.1.2 输出文件 8](#_Toc477508007)

[4.1.3 文件要求 8](#_Toc477508008)

[4.1.3.1 文件名 8](#_Toc477508009)

[4.1.3.2 注意事项 9](#_Toc477508010)

[4.2 服务概要设计 9](#_Toc477508011)

[4.2.1 目的 9](#_Toc477508012)

[4.2.2 注意事项 9](#_Toc477508013)

[4.2.3 输出文件 9](#_Toc477508014)

[4.2.4 文件模板 9](#_Toc477508015)

[4.2.5 文件要求 9](#_Toc477508016)

[4.2.5.1 文件名 9](#_Toc477508017)

[4.2.5.2 注意事项 10](#_Toc477508018)

[4.3 服务详细设计 10](#_Toc477508019)

[4.3.1 目的 10](#_Toc477508020)

[4.3.2 注意事项 10](#_Toc477508021)

[4.3.3 输出文件 10](#_Toc477508022)

[4.3.4 文件模板 10](#_Toc477508023)

[4.3.5 文件要求 10](#_Toc477508024)

[4.3.5.1 文件名 10](#_Toc477508025)

[4.3.5.2 注意事项 11](#_Toc477508026)

[4.4 服务划分 11](#_Toc477508027)

[4.4.1 服务划分原则 11](#_Toc477508028)

[4.4.2 基础服务 11](#_Toc477508029)

[4.4.3 后端服务 11](#_Toc477508030)

[4.4.4 前端服务 11](#_Toc477508031)

[4.4.5 公共服务 11](#_Toc477508032)

[4.5 服务依赖原则 12](#_Toc477508033)

[4.6 服务接口 12](#_Toc477508034)

[4.6.1 本地接口 12](#_Toc477508035)

[4.6.2 远程接口 12](#_Toc477508036)

[4.6.3 远程接口客户端（桩代码生成） 12](#_Toc477508037)

[4.7 异常类设计 12](#_Toc477508038)

[4.8 定时任务设计 12](#_Toc477508039)

[4.8.1 定时任务命名 12](#_Toc477508040)

[4.8.2 定时任务存放位置 13](#_Toc477508041)

[5 服务开发 13](#_Toc477508042)

[5.1 格式 13](#_Toc477508043)

[5.2 数据传递 13](#_Toc477508044)

[5.3 XML解析 13](#_Toc477508045)

[5.4 泛型 13](#_Toc477508046)

[5.5 警告信息 13](#_Toc477508047)

[5.6 注释信息 13](#_Toc477508048)

[5.7 文档-JAVADOC 14](#_Toc477508049)

[5.7.1 文件头样例 14](#_Toc477508050)

[5.7.2 package-info样例 15](#_Toc477508051)

[5.7.3 方法样例 15](#_Toc477508052)

[5.8 测试用例 15](#_Toc477508053)

[5.9 日志信息 15](#_Toc477508054)

[5.10 代码版本控制 16](#_Toc477508055)

[5.11 参考资料 16](#_Toc477508056)

# 前言

## 背景

本规范的主要目的是规范微服务设计开发阶段的设计、开发人员执行准则，并最终进入公共服务范畴和更好的服务扩展。最终目标是我们把共性的服务进行组装，进化为产品或平台。

## 文档范围

所有服务开发设计范畴。

## 目标读者

研发全员。

# 服务命名

## 总体规范

如无特殊说明，各部分命名应遵循公司基础《语言编码规范》。

## 服务命名

服务的中英文命名应简洁直白，明确表达服务的功能，避免使用非智能光轨的缩略语或缩写。英文名应该是按照“驼峰式”书写的几个单词，避免过于冗长。

## 服务类型命名

基础服务：base

后端服务：service

前端服务：ui

公共服务：common

## 包名、类名

基础服务的包命名应为:com.yappam. <服务类型名>.<全小写的服务英文名>

后端服务的包命名：com.yappam.<服务类型名>.<全小写的服务英文名>

前端服务的包命名：com.yappam.<服务类型名>.<全小写的服务英文名>

公共服务的包命名应为:com.yappam. <服务类型名>.<全小写的服务英文名>

公共服务（启动容器）部分包命名在顶层包路径com.yappam下

备注:启动服务因为spring boot需要进行包路径扫描，所以放在顶层

# 微服务（SVN）结构

## SVN微服务地址

svn://192.168.0.176/MyChevrolet/trunk/codes/microservices

## SVN子结构命名

### 基础服务根目录名

base/版本号/doc；

base/版本号/yappam-base-<全小写的服务英文名>；

### 后端服务根目录名

service/版本号/doc；

service /版本号/yappam-<全小写的服务英文名>-service；

### 前端服务根目录名

ui/版本号/doc；

ui /版本号/yappam-<全小写的服务英文名>-ui；

### 公共服务根目录名

common/版本号/doc；

common/版本号/yappam- common-<全小写的服务英文名>-server；

### 产品根目录名

product/产品名称/版本号/doc；

product/产品名称/版本号/code；

### 项目根目录

project/项目名称/版本号/doc；

project /项目名称/版本号/code；

## 服务子目录

demo（应用场景举例）

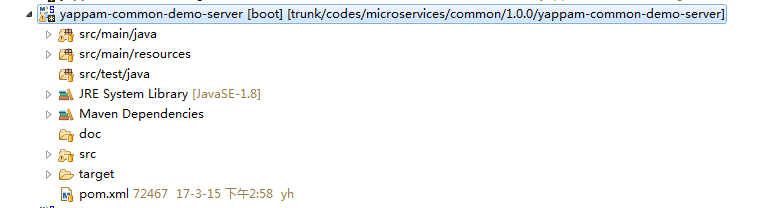
doc（描述文档doc\What’sNew\数据库变更记录\数据库设计PDM文件\UML等）

src（源码目录）：main（主源码目录）java（java源码）

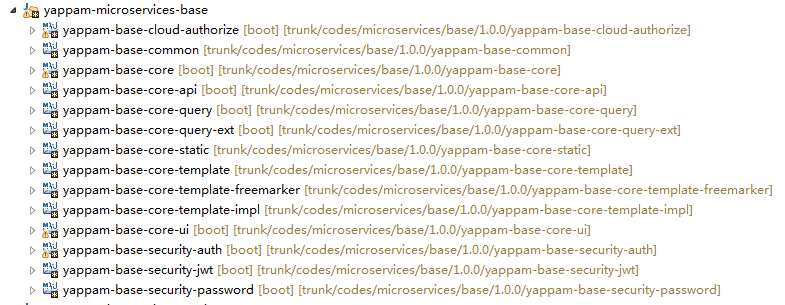
resources（主配置文件）

test（单元测试目录）java（java源码）

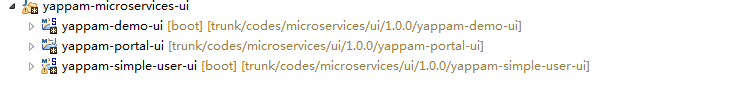
resources（单元测试配置文件）



**全景目录结构如下：**









# 服务设计

## 数据库设计

### 注意事项

服务的数据库设计遵循《数据库设计规范》。

### 输出文件

DDL文件

### 文件要求

#### 文件名

服务英文名\_<数据库名称>.ddl。

例如：Oracle 数据库的DDL:服务英文名\_oracle.ddl,Mysql同理。

#### 注意事项

格式整齐，有必要的注释说明。对以前发布的数据表结构进行修改时应注意兼容。

推荐根据实体类进行表结构反推定义，避免数据库变更引起的整体调整。

## 服务概要设计

### 目的

确定服务各项命名，包括服务中文名，英文名及jar包名等。明确服务的功能边界。明确与其他服务的调用关系等信息。

### 注意事项

使用PowerDesigner或StarUML作为设计建模工具，将设计内容以UML形式体现。

确定服务调用关系时，应以业务上依赖为原则，避免在业务中体现界面依赖及某项目或产品特有依赖等非正常业务的依赖关系。

### 输出文件

UML设计图

Word文档

PDM 模型图

### 文件模板

《XX项目概要设计文档模板》

### 文件要求

#### 文件名

<服务类型名>\_服务中文名(英文名)\_概要设计.docx

<服务类型名>\_服务中文名(英文名)\_概要设计.oom/.uml

<服务类型名>\_服务中文名(英文名)\_概要设计.pdm

#### 注意事项

设计内容阐述清楚完整。图形绘制清晰整洁。

图块大小一致，连线横平竖直。

在特殊时期，可在两种文件类型中二选一。

模板文档结构可伸缩选择。

## 服务详细设计

### 目的

说明每个业务方法的功能及复合场景。

说明每个业务方法的方法命名、参数及返回值含义。

说明每个业务方法的主要流程。

### 注意事项

使用PowerDesigner或StarUML作为设计建模工具，将设计内容以UML形式体现。

对各主要方法应以活动图，序列图，或者状态图等方式准确描述其业务流程，尤其是包含逻辑判断或者分支的流程

### 输出文件

UML设计图

Word文档

PDM 模型图

### 文件模板

《XX项目XX模块详细设计文档模板》

### 文件要求

#### 文件名

<服务类型名>\_服务中文名(英文名)\_详细设计.docx

<服务类型名>\_服务中文名(英文名)\_详细设计.oom/.uml

<服务类型名>\_服务中文名(英文名)\_详细设计.pdm

#### 注意事项

设计内容阐述清楚完整。图形绘制清晰整洁。

图块大小一致，连线横平竖直。

在特殊时期，可在两种文件类型中二选一。

模板文档结构可伸缩选择。

## 服务划分

### 服务划分原则

服务划分遵循功能完整、职责单一和粒度适中原则，初步划分为：

1. 基础服务
2. 后端服务
3. 前端服务
4. 公共服务

### 基础服务

基础服务严格意义说只是其他服务通过maven引用的sdk组件,提供常用的公共方法、数据查询公共类、数据封装与绑定、通知、过滤、前端模板、后端授权认证、统一静态资源等。

### 后端服务

后端服务类似业务逻辑层，涵盖业务操作、API调用、逻辑运算等

后端服务以业务功能的原子化为设计理念，保证业务流程内部自治、数据自治。

### 前端服务

前端服务类似视图层定义，负责前端展示和调用后端服务等

前端服务以后台功能菜单或首页页面展示版块为设计理念，保证功能流程内部自治、展示自治。

### 公共服务

公共用于启动服务、配置服务等

公共服务是在后端和前端服务基础上（一般情况下一个service对应一个ui）构建，微服务常用的功能也都涵盖在其中，例如监控、注册、发现、熔断、路由、边界、配置等。

公共服务是启动项目和产品的入口，通过不同的组合可以发布不同的产品或项目。

## 服务依赖原则

其他服务对本服务的调用只能基于本服务对外提供的服务接口。

同层服务间不能有依赖，否则可考虑合并服务。

不可跨层次服务依赖。

如特殊业务场景需要跨服务进行分布式事务，采用稳定事件模型和补偿机制，我们会针对特殊业务场景先保证单服务数据一致性，然后将事件信息推送至异步中间件例如（MQ）,通过关联服务进行消费服务保证最终的事务一致性。

如因业务原因必须实时保证事务一致性，我们推荐将服务进行聚合，保证强一致性。

## 服务接口

### 本地接口

本地直接调用服务API接口

### 远程接口

服务间远程调用以Rest等方式对外提供

### 远程接口客户端（桩代码生成）

服务需要作为客户端访问第三方提供的WebService接口时，应使用CXF/AXIS等或其他IDE生成的对应桩代码，并放入服务中，桩文件的包命名规则：com.yappam.<服务类型命名>.<全小写的服务英文名>.WS功能名+Stub。

## 异常类设计

异常类命名：异常含义（控制范围或分层含义等）+Exception。

异常类的包命名：com.yappam.<服务类型命名>.<全小写的服务英文名>.exception

## 定时任务设计

### 定时任务命名

com.yappam.<服务类型命名>.<全小写的服务英文名>.功能名+Job+Service

### 定时任务存放位置

代码文件放置在业务接口实现的同级目录之下。

# 服务开发

## 格式

除非有特殊需求，所有代码，包括java、xml等，均为UTF-8 编码。

## 数据传递

服务内部使用POJO传递；服务间的数据传递使用POJO；

## XML解析

数据结构复杂的XML解析统一使用DOM4J进行解析。

键值对类型的数据可以使用DOM4J和commons-digester进行解析。

## 泛型

各种集合复合泛型声明元素的类型，例如：ArrayList<IPojo>而少用ArrayList。

对集合的遍历使用上，如无计数要求，请用for each语句。例如:for(JavaBean : beans)。

## 警告信息

编码时应注意处理IDE报出的警告信息，原则上不允许有警告信息。

对于POJO数据Bean实现Serializable接口所产生的警告，不许使用@SuppressWarnings(“serial”)注解来忽略这个警告，应提供serialVersionUID，而且不同类的serialVersionUID应该是不同的。

对于泛型的警告，应尽量使用合适的泛型方式来处理。

对于过期方法的警告，应尽量采用合适的替代方法。

## 注释信息

代码中应有“恰到好处”的注释信息，用以辅助其它开发者读懂你的代码。

应清除那些由某些生成工具或模板产生的无意义的注释，如：“generated by。。”之类的。

对于由于某种原因不再使用但希望在代码中保留的代码段使用段注释（/\*\* \*/）的方式将其注释掉，同时应在此注释段开头说明这样做的理由。

应为每个分支书写注释信息，表明该分支的含义。

## 文档-JAVADOC

为每个方法（除了bean属性的set/get方法）及字段，包括保护的以及私有的，编写javadoc信息，说明方法/字段的功能，每个参数的含义，返回值的含义，异常的说明等。对有约定格式的参数或者返回值应说明其格式。

如果是接口中定义的方法可以直接把JAVADOC书写的接口中，在实现类中可以省略。

推荐采用Swagger方式进行API自动生成

### 文件头样例

/\*\*

\* Copyright (c) 2013-10 YAPPAM, LTD.CO

\* All rights reserved.

\* This software is the confidential and proprietary information of

\* YAPPAM LTD.CO. ("Confidential Information"). You shall not

\* disclose such Confidential Information and shall use it

\* only in accordance with the terms of the license agreement you entered into

\* with YAPPAM.

\*/

**package** com.yappam.yappamframework.core.dao;

**import** java.io.Serializable;

**import** java.util.Collection;

**import** java.util.List;

**import** com.yappam.yappamframework.core.exception.DataAccessLayerException;

**import** com.yappam.yappamframework.core.support.page.Page;

/\*\*

\*

\* 数据对象DAO层公共接口，提供基本CRUD操作方法 <br>

\*

\* **@see** IDao

\* **@author** zzh

\* **@version** 1.0.0

\* **@since** JDK1.5

\* **@param** <D>

\* 序列化泛型对象

\*/

**public** **interface** IPojoDao<D **extends** Serializable> **extends** IDao {

### package-info样例

/\*\*

\*

\* 数据持久层公共调用接口包，包括所有持久层父接口及POJO父接口定义

\*/

**package** com.yappam.yappamframework.core.dao;

### 方法样例

/\*\*

\* 添加序列化对象。

\*

\* **@param** obj

\* 序列化对象

\* **@return** 返回主键编号

\* **@throws** DataAccessLayerException

\*/

String save(D obj) **throws** DataAccessLayerException;

## 测试用例

应为各主要业务方法编写单元测试用例，包括主用例，异常用例，边缘用例，当接口发生变化时，应同时对测试用例进行修正。

一个业务方法对应一个测试方法，测试方法中用注释注明各个测试用例。

单元测试方法中的测试用例应该保持稳定，保证回归单元测试的有效性。

可以对测试方法补充测试用例，也可以对新增业务方法增加测试方法。

测试用例应以junit测试用例用框架进行编写，避免使用在程序中编写main方法等方式。

## 日志信息

应避免使用System.out.println()或者exception.printStackTrace()之类的语句直接向控制台输出文字信息。

代码中复合合适的日志级别记录日志信息，禁止使用error级别记录调试性的信息。

记录日志信息时应使用可读的描述文字，避免使用无含义的大片字母或符号

## 代码版本控制

编码人员要按照不同应用建立相应的eclipse或idea 工程，以保证在不同项目之间切换时，对应项目代码的正确性。

编码人员按照要求从SVN服务库中下载最新代码。编码自测完成后，上传SVN。

## 参考资料

1、《java语言代码规范》，Sun Microsystems, Inc.，CodeConventions.pdf

2、《语言编码规范》，语言编码规范.docx

3、《数据库设计规范》,数据库设计规范.docx

4、《XX项目概要设计文档模板》，XX项目概要设计文档模板.docx

5、《XX项目XX模块详细设计文档模板》，XX项目XX模块详细设计文档模板.docx

6、《阿里巴巴java开发手册.pdf》