**成都羿明数字技术有限公司**

**产品开发手册**

目录

[文档历史 2](#_Toc461543673)

[1:概述 2](#_Toc461543674)

[2:准则 3](#_Toc461543675)

[3:结构 4](#_Toc461543676)

[1) 框架结构 4](#_Toc461543677)

[2) 项目结构： 4](#_Toc461543678)

[3) 代码结构： 4](#_Toc461543679)

[4:规范 16](#_Toc461543680)

# 文档历史

|  |  |  |  |
| --- | --- | --- | --- |
| 作者 | 时间 | 版本 | 备注 |
| Jim.lee(李星月) | 2016/9/13 | V0.1 | 初稿，完成文档架构：概述、准则，结构及Service，GUI项目部分规范 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 1:概述

本文用于为产品开发人员制定开发准则及相关公用组件、示例代码

# 2:准则

1. 所有项目基于JAVA7，如果是GUI项目使用Tomcat8.0
2. 所有项目开发使用Eclipse 4.5及以上版本
3. 产品开发分为 Dubbo Service 及 前端WEB GUI，在开发过程中两者代码分离，由Dubbo Service 向 前端GUI 提供接口及接口版本
4. 前端GUI开发不会涉及Dubbo Service 代码
5. Service开发人员必须在接口上注释描述 示例：

/\*\*

\* 保存用户信息

\* **@author** jim

\* **@since** 2016-09-01

\* **@param** person

\* **@see** Person

\* **@return** Long (用户ID)

\*

\*/

**public** Long saveOrUpdatePerson(Person person);

接口注解至少包含：

1. 接口功能描述
2. 接口提供者（具体到开发人员）
3. 接口提供日期
4. 参数列表 （若参数为domain实体类需要提供链接查看 @see）
5. 返回类型及描述

# 3:结构

## 3.1框架结构

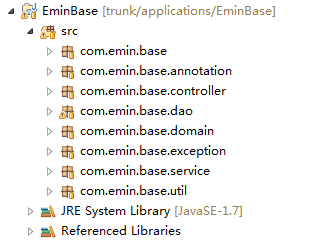
所有的Dubbo Service向zookeeper中心注册服务，所有的前端GUI从zookeeper获取服务；所有的 Dubbo Service 可以通过 zookeeper 调用其他Service

## 3.2项目结构：

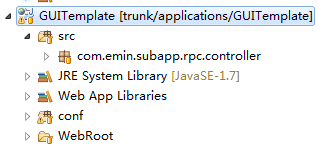
每一个项目分为Service及GUI。Service采用Spring4+hibernate4；GUI采用SrpingMVC4+FreeMarker2.3.18+Ace Admin 3.x

## 3.3代码结构：

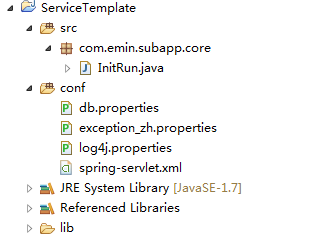
产品代码结构图：



EminBase：所有项目都会依赖EminBase，EminBase会已Jar包形式存在于各个项目中。



GUI项目：图中所示有GUI模板项目，具体命名按照应用命名例如：产品管理（prdmGUI）、物流管理（lmsGUI）等



Service项目：图中所示为Service模板项目，具体命名按照应用命名例如：产品管理（prdm）、物流管理（lms）等

目录说明：src 项目源码文件目录，conf 项目配置文件目录

对于conf中必须包含如下配置文件

|  |  |
| --- | --- |
| db.properties | 数据库配置文件（GUI项目不需要） |
| log4j.properties | 日志配置文件 |
| spring-servlet.xml | Spring配置文件 |
| exception\_zh.properties | 错误信息国际化配置文件，如增加语言需增加配置文件，例如英文版 需增加  exception\_en.properties |
| <appname>\_config.properties | 应用其他配置文件<appname>替换为具体应用名，例：prdm\_config.properties |

配置文件说明：

db.properties 内容

# Database config

db.driverClass=org.postgresql.Driver

#DB default user and password, this will be used as all the database

db.default.user=postgres

db.default.password=1

db.plt.baseUrl=jdbc\:postgresql\://192.168.0.200\:5432

db.plt.name=meris

db.plt.user=postgres

db.plt.password=1

#Configuratin for the pool

db.poolClass=com.mchange.v2.c3p0.ComboPooledDataSource

db.minPoolSize=5

db.maxPoolSize=30

db.initialPoolSize=10

db.maxIdleTime=60

db.acquireIncrement=5

db.maxStatements=0

db.idleConnectionTestPeriod=60

db.acquireRetryAttempts=30

db.breakAfterAcquireFailure=true

db.testConnectionOnCheckout=false

# Hibernate config

hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

hibernate.show\_sql=true

hibernate.format\_sql=true

use\_sql\_comments=true

hibernate.generate\_statistics=false

hibernate.jdbc.batch\_size=1000

hibernate.cache.provider\_class=org.hibernate.cache.EhCacheProvider

hibernate.hbm2ddl.auto=create-drop

其中应包含：

数据库连接驱动、连接url、数据库名称、账号&密码、c3p0连接池配置

若没有特殊情况仅需改动 《连接url、数据库名称、账号&密码》三项配置即可

在调试过程中

hibernate.hbm2ddl.auto=create-drop

如果在发布时请将配置改为

hibernate.hbm2ddl.auto=update

log4j.properties 内容：

# Configure logging for testing: optionally with log file

log4j.rootLogger=INFO, stdout

# log4j.rootLogger=WARN, stdout, logfile

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.appender.logfile=org.apache.log4j.FileAppender

log4j.appender.logfile.File=target/eminview.log

log4j.appender.logfile.append=false

log4j.appender.logfile.layout=org.apache.log4j.PatternLayout

log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.logger.com.emin=INFO

log4j.logger.org.hibernate=WARN

rootlogger 在开发过程中

log4j.rootLogger=INFO, stdout

发布时

log4j.rootLogger=INFO, logfile

log4j.appender.logfile.File=target/eminview.log

日志文件目录需根据操作系统配置

exception\_zh.properties 内容：

exception.code.1.14=查询参数格式不对

exception.code.1.15={0}不存在

exception.code.2.1.5=非法参数,请检查必填项

exception.code.2.1.6=<{0}>已被使用

exception.code.2.3.2=密码错误

exception.code.2.3.3=密码为空

exception.code.2.3.4=没有登录权限

exception.code.2.3.5=用户名错误

exception.code.2.3.12=用户不存在

exception配置文件必须在GUI项目中存在，否则无法找到对应的异常信息

spring配置文件：

Service项目Spring配置文件：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*

xmlns:util=*"http://www.springframework.org/schema/util"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-4.0.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*

*http://code.alibabatech.com/schema/dubbo*

*http://code.alibabatech.com/schema/dubbo/dubbo.xsd*

*"*>

<!-- 启用spring mvc 注解 -->

<context:annotation-config />

<!-- 设置使用注解的类所在的jar包 -->

<context:component-scan base-package=*"com.emin"* ></context:component-scan>

<aop:aspectj-autoproxy></aop:aspectj-autoproxy>

<aop:config proxy-target-class=*"false"*></aop:config>

<bean id=*"springContextHolder"* class=*"com.emin.base.util.SpringContextHolder"*></bean>

<bean id=*"placeholderConfig"* class=*"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"*>

<property name=*"location"* value=*"classpath:/db.properties"* />

</bean>

<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*

>

<property name=*"driverClass"* value=*"${db.driverClass}"* />

<property name=*"jdbcUrl"* value=*"${db.plt.baseUrl}/${db.plt.name}"* />

<property name=*"user"* value=*"${db.plt.user}"* />

<property name=*"password"* value=*"${db.plt.password}"* />

<property name=*"minPoolSize"* value=*"${db.minPoolSize}"* />

<property name=*"maxPoolSize"* value=*"${db.maxPoolSize}"* />

<property name=*"initialPoolSize"* value=*"${db.initialPoolSize}"* />

<property name=*"maxIdleTime"* value=*"${db.maxIdleTime}"* />

<property name=*"acquireIncrement"* value=*"${db.acquireIncrement}"* />

<property name=*"maxStatements"* value=*"${db.maxStatements}"* />

<property name=*"idleConnectionTestPeriod"* value=*"${db.idleConnectionTestPeriod}"* />

<property name=*"acquireRetryAttempts"* value=*"${db.acquireRetryAttempts}"* />

<property name=*"breakAfterAcquireFailure"* value=*"${db.breakAfterAcquireFailure}"* />

<property name=*"testConnectionOnCheckout"* value=*"${db.testConnectionOnCheckout}"* />

</bean>

<bean id=*"sessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>

<!-- 依赖注入上面定义的数据源dataSource -->

<property name=*"dataSource"* ref=*"dataSource"* />

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>

${hibernate.dialect}

</prop>

<prop key=*"hibernate.show\_sql"*>

${hibernate.show\_sql}

</prop>

<prop key=*"use\_sql\_comments"*>${use\_sql\_comments}</prop>

<prop key=*"hibernate.format\_sql"*>

${hibernate.format\_sql}

</prop>

<prop key=*"hibernate.generate\_statistics"*>

${hibernate.generate\_statistics}

</prop>

<prop key=*"hibernate.jdbc.batch\_size"*>

${hibernate.jdbc.batch\_size}

</prop>

<prop key=*"hibernate.hbm2ddl.auto"*>

${hibernate.hbm2ddl.auto}

</prop>

<prop key=*"hibernate.cache.region.factory\_class"*>org.hibernate.cache.ehcache.EhCacheRegionFactory</prop>

<prop key=*"hibernate.cache.use\_query\_cache"*>true</prop>

</props>

</property>

<property name=*"packagesToScan"*>

<list>

<value>com.emin.platform.domain</value>

</list>

</property>

</bean>

<bean class=*"org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"* />

<!--定义Hibernate的事务管理器HibernateTransactionManager -->

<bean id=*"transactionManager"* class=*"org.springframework.orm.hibernate4.HibernateTransactionManager"*>

<!-- 依赖注入上面定义的sessionFactory -->

<property name=*"sessionFactory"* ref=*"sessionFactory"* />

</bean>

<tx:annotation-driven transaction-manager=*"transactionManager"* />

<dubbo:application name=*"meris"* />

<dubbo:registry address=*"zookeeper://192.168.0.222:2181"* />

<dubbo:protocol name=*"dubbo"* port=*"20080"*></dubbo:protocol>

<dubbo:consumer check=*"false"*/>

<dubbo:provider accesslog=*"true"*></dubbo:provider>

<dubbo:service interface=*"com.emin.platform.service.PersonService"* ref=*"personService"* version=*"0.0.1"* protocol=*"dubbo"*/>

<dubbo:service interface=*"com.emin.platform.service.CompanyService"* ref=*"companyService"* version=*"0.0.1"* protocol=*"dubbo"*/>

<dubbo:service interface=*"com.emin.platform.service.OrganizeService"* ref=*"organizeService"* version=*"0.0.1"* protocol=*"dubbo"*/>

</beans>

配置文件中的packagesToScan根据需要配置

要公布的Dubbo服务需要在配置文件中指明 接口、版本及协议

例如：

<dubbo:service interface=*"com.emin.platform.service.CompanyService"* ref=*"companyService"* version=*"0.0.1"* protocol=*"dubbo"*/>

GUI项目Spring配置文件：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*

xmlns:util=*"http://www.springframework.org/schema/util"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-4.0.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*

*http://code.alibabatech.com/schema/dubbo*

*http://code.alibabatech.com/schema/dubbo/dubbo.xsd*

*"*>

<!-- 启用spring mvc 注解 -->

<context:annotation-config />

<mvc:resources location=*"/ace/"* mapping=*"/ace/\*\*"*/>

<mvc:resources location=*"/css/"* mapping=*"/css/\*\*"*/>

<mvc:resources location=*"/resources/"* mapping=*"/resources/\*\*"*/>

<mvc:resources location=*"/js/"* mapping=*"/js/\*\*"*/>

<mvc:resources location=*"/"* mapping=*"/\*.html"*/>

<!-- 设置使用注解的类所在的jar包 -->

<context:component-scan base-package=*"com.emin"*></context:component-scan>

<aop:aspectj-autoproxy></aop:aspectj-autoproxy>

<aop:config proxy-target-class=*"false"*></aop:config>

<bean id=*"springContextHolder"* class=*"com.emin.base.util.SpringContextHolder"*></bean>

<!-- 完成请求和注解POJO的映射 -->

<bean class=*"org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"*>

<property name=*"messageConverters"*>

<list >

<ref bean=*"mappingJacksonHttpMessageConverter"* />

<bean class=*"org.springframework.http.converter.ByteArrayHttpMessageConverter"*/>

<bean class=*"org.springframework.http.converter.StringHttpMessageConverter"*/>

<bean class=*"org.springframework.http.converter.FormHttpMessageConverter"*/>

</list>

</property>

</bean>

<!-- 处理JSON数据转换的 -->

<bean class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"*/>

<bean class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"*>

<property name=*"messageConverters"*>

<list>

<ref bean=*"mappingJacksonHttpMessageConverter"* />

<bean class=*"org.springframework.http.converter.ByteArrayHttpMessageConverter"*/>

<bean class=*"org.springframework.http.converter.StringHttpMessageConverter"*/>

<bean class=*"org.springframework.http.converter.FormHttpMessageConverter"*/>

</list>

</property>

</bean>

<bean id=*"mappingJacksonHttpMessageConverter"*

class=*"org.springframework.http.converter.json.MappingJackson2HttpMessageConverter"*>

<!-- 为了处理返回的JSON数据的编码，默认是ISO-88859-1的，这里把它设置为UTF-8,解决有乱码的情况 -->

<property name=*"supportedMediaTypes"*>

<list>

<value>application/json; charset=UTF-8</value>

<value>text/html; charset=UTF-8</value>

<value>application/x-www-form-urlencoded; charset=UTF-8</value>

</list>

</property>

</bean>

<bean id=*"multipartResolver"* class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>

<!-- set the max upload size100MB -->

<property name=*"maxUploadSize"*>

<value>104857600</value>

</property>

<property name=*"maxInMemorySize"*>

<value>4096</value>

</property>

<property name=*"defaultEncoding"*>

<value>UTF-8</value>

</property>

</bean>

<dubbo:application name=*"platform-gui"* />

<dubbo:registry address=*"zookeeper://192.168.0.222:2181"* />

<dubbo:consumer check=*"false"*></dubbo:consumer><!-- 启动时不检查 -->

<!-- 扫描注解包路径，多个包用逗号分隔，不填pacakge表示扫描当前ApplicationContext中所有的类 -->

<dubbo:annotation />

</beans>

# 4:规范

## 4.1 Service项目规范

包名命名规范：com.emin.<appname>.xxx

appname 替换为相应的应用名称 例如：prdm、lms

xxx可以为service.、domain、dao、util等

其下可以有子包，例如：com.emin.<appname>.service.impl

所有的Domain必须继承BaseEntity

示例：

**public** **class** OrgMember **extends** BaseEntity

如果实例只需要做逻辑删除处理则必须实现UndeleteableEntity接口

示例：

**public** **class** Person **extends** BaseEntity **implements** UndeleteableEntity

映射数据库表的Domain都需要注解JPA的@Entity及@Table标签

示例：

@Entity

@Table(schema="platform",name="person")

每个field对应的get方法需要注解@Column及字段说明

每个Domain需有一个对应的Service接口及实现类

如果非逻辑删除处理的Domain 则其对应的Service接口需要继承CRUDSerice接口，其接口实现类需要继承CRUDServiceImpl类

如果是逻辑删除处理的Domain则其对应的Service接口需要继承UndeleteableService接口，其接口实现类需要继承UndeleteableServiceImpl类

示例：

**public** **interface** PersonService **extends** UndeleteableService<Person>

@Service("personService")

**public** **class** PersonServiceImpl **extends** UndeleteableServiceImpl<Person> **implements** PersonService

注意：Service实现类需要注解Spring @Service标签并指定Service名称，命名规则为接口对象引用格式：如PersonServiceImpl注解的则为PersonService接口的对象引用名称personService 首字母小写

需要将查询字段暴露给GUI项目的都需要在Domain中声明常量

例如：查询名称

**public** **static** **final** String ***PROP\_NAME***= "name";

书写规范PROP\_字段名称大写

注意：常量值需要与Domain Field名称对应，并非数据库字段名称

EminBase中封装了大量CRUD方法，具体参考EminBase中CRUDService接口及UndeleteableService接口

Service内部使用查询时参考EminBase下的PreFilter及PreFilters类

示例：

**public** Person login(String mobilePhone, String password) {

**if** (**null** == mobilePhone || **null** == password) {

**throw** **new** RuntimeException("账号或密码不能为空");

}

PreFilter mobileFilter = PreFilters.*eq*(Person.***PROP\_MOBILE\_PHONE***,

mobilePhone);

PreFilter passwordFilter = PreFilters

.*eq*(Person.***PROP\_PASSWORD***, password);

PreFilter statusFilter = PreFilters.*ge*(Person.***PROP\_STATUS***, 0);

Person person = **this**.findUniqueByPreFilter(mobileFilter,

passwordFilter, statusFilter);

**if** (**null** == person) {

**throw** **new** RuntimeException("账号或密码错误");

}

**return** person;

}

Service抛出的业务处理异常统一为EminException，exceptionCode作为参数code规则参见exception\_<zh/en>.properties

示例：

**throw** **new** EminException("2.3.2");

若code对应的异常信息中有占位符则根据占位符数量添加变长参数即可

示例：

exception.code.2.1.6=<{0}>已被使用

**throw** **new** EminException("2.1.6",name);

占位符下标从0开始 {0}，{1}，{2}....

Service引用规则：

若在Service实现类中需要引用其他Service需要先申明对应Service接口，然后加上@AutoWrite、@Qualifier("serviceName")注解

serviceName对应Service实现类上@Service注解中命名的名称

示例：

@Autowired

@Qualifier("personService")

**private** PersonService personService;

## 4.3 Android项目规范

包名命名规范：com.emin.<appname>.mobile.xxx

appname 替换为相应的应用名称 例如：meris

xxx可以为plugin、util、等

插件开发说明: 每一个插件对应一个Java插件类和一个JavaScript文件

JAVA部分的插件代码规范

命名规范:PluginXxx,例如PluginBarcode,PluginDatabse

Plugin类中的功能方法,有且仅有一个参数,PluginParams

例如:PluginDatabase中的往数据表中插入数据方法:

**public void** insert(PluginParams params)

其中PluginParams封装了JS传递的参数,params.getArguments()返回了参数数组,通过该数组的index来获取相应的参数,

例如JS方法中传递了,insert的sql语句,则String sqlString = params.getArguments()[0];

JavaScript部分的插件代码规范

文件命名规范:pluginXxx.js,例如pluginBarcode.js,pluginDatabase.js

## 4.2 GUI项目规范

包名命名规范：com.emin.<appname>.xxx

appname 替换为相应的应用名称 例如：prdm、lms

xxx可以为controller、util、listener等

其下可以有子包，例如：com.emin.<appname>.controller.mobile

所有的Controller必须继承BaseController

示例：

**public** **class** PersonController **extends** BaseController

并且在Class上需要加上@Controller注解

所有的Controller方法前都要加上@RequestMapping注解指定请求路径，SpringMvc默认以当前项目根路径作为相对路径，

由于所有Controller方法都已JSON格式返回，无论方法返回什么对象最终都会解析为JSON，故方法前必须加上@ResponseBody标签

Controller中如何使用Dubbo服务

Controller若要使用Service，需要先由Service提供方的Jar包，并且在Controller中申明接口，然后加上@Reference注解，并填入version即可

示例：

@Reference(version="0.0.1")

**private** PersonService personService;

若在调用服务方法而服务会抛出异常时，必须用try-catch捕获并获得异常文本返回前端

注意必须首先捕获EminException

示例：

**try** {

Company company = **new** Company();

company.setId(1l);

PltThreadLocalUtil.*setCompany*(company);

Person person = personService.login("13800000000", "888888");

json.put("person", person);

success = **true**;

} **catch** (EminException e) {

message = e.getLocalizedMessage();//此处会根据当前环境自动匹配对应异常配置文件中的信息

e.printStackTrace();

}

**catch** (Exception e) {

message = "登录失败";

e.printStackTrace();

}