|  |  |
| --- | --- |
| 文档编号 | EayunCloud-T04 |
| 文档版本 | 1.0.0 |
| 拟制人 | 徐光柳 |
| 日期 | 2016-8-1 |

EayunCloud Java开发编码规范

目录

[变更说明 3](#_Toc461978925)

[1. 简介 4](#_Toc461978926)

[1.1 概述 4](#_Toc461978927)

[1.2 目的 4](#_Toc461978928)

[1.3 参考资料 4](#_Toc461978929)

[2. 工程规范 4](#_Toc461978930)

[3. 程序MVC架构和规范 7](#_Toc461978931)

[3.1 包名和类名 8](#_Toc461978932)

[3.2 Controller 8](#_Toc461978933)

[3.3 Service 9](#_Toc461978934)

[3.4 Dao 9](#_Toc461978935)

[3.5 Model 9](#_Toc461978936)

[3.6 Job 9](#_Toc461978937)

[3.7 Thread 10](#_Toc461978938)

[3.8 常量 10](#_Toc461978939)

[3.9 变量和方法名 10](#_Toc461978940)

[3.10 字符串拼接 10](#_Toc461978941)

[4. Dao规范 10](#_Toc461978942)

[4.1 整体规范 10](#_Toc461978943)

[4.2 HQL:增，改 10](#_Toc461978944)

[4.3 HQL:删 12](#_Toc461978945)

[4.4 HQL:查-不分页 12](#_Toc461978946)

[4.5 HQL:查-分页 14](#_Toc461978947)

[4.6 SQL：增删改 15](#_Toc461978948)

[4.7 SQL:查，不分页 15](#_Toc461978949)

[5. 工程日志 16](#_Toc461978950)

[6. 异常处理 16](#_Toc461978951)

[7. 方法名 16](#_Toc461978952)

[8. 代码组织与风格 17](#_Toc461978953)

[8.1 导入eayun formatter模板 17](#_Toc461978954)

[8.2 导入code template模板 17](#_Toc461978955)

[8.3 缩进，间隔，风格等 18](#_Toc461978956)

[9. 注释 18](#_Toc461978957)

[9.1 基本原则 18](#_Toc461978958)

[10. 声明 19](#_Toc461978959)

[11. RedisKey 19](#_Toc461978960)

[12. MongoColletionName 19](#_Toc461978961)

变更说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **变更位置** | **变更说明** | **作者** |
| 2015-8-15 | 1.0.0 |  | **新建** | 陈浩 |
| 2016-7-29 | 1.0.0 |  | **评审修订** | 陈浩 |
| 2016-08-01 | 1.0.0 |  | **梳理修订** | 徐光柳 |
| 2016-08-19 | 1.0.0 |  | **明确job和thread不是spring管理的bean** | 陈浩 |
| 2016-09-01 | 1.0.0 |  | **增加对Redis , Mongo的要求** | 陈浩 |
| 2016-09-18 | 1.0.0 |  | **完善对异常处理的要求** | 陈浩 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 简介

## 概述

本文提供一整套编写高效可靠的代码的标准、约定和规范。它们以安全可靠的软件工程原则为基础，使代码易于理解、维护和增强。

最根本的原则：运用常识。当找不到任何规则或指导方针，当规则明显不能适用，当所有的方法都失效的时侯，运用常识并核实这些基本原则。这条规则比其它所有规则都重要。常识是必不可少的。

## 目的

本规范是使本项目能以标准的、规范的方式设计和编码，在使自己养成了良好的编码风格的同时，增强代码的可维护性。特别地，一致的代码注释风格，不仅有助于提高代码注释的可读性，同时为使用自动化工具进行开发文档的提取工作提供一致的基础数据。

## 参考资料

《Java 编程指南》见RUP（Rational Unified Process）中文版。

《Java技术手册》(Java in a Nutshell)

# 工程规范

ecsc和ecmc项目采用maven来管理，规范如下：

1. groupId：固定为com.eayun
2. artifactId：eayun-xxx。举例：eayun-common，eayun-mail，eayun-mail-intf
3. packaging：

|  |  |
| --- | --- |
| **packaging** | **说明** |
| jar | java业务工程  只包含java程序，其中eayun-common为公共的工程，eayun-eayunstack为和EayunStack底层交互的工程，其他业务工程按业务需要建 |
| war | 部署工程  只包含html，js，xml，配置文件，不含java程序，根据业务需要依赖一些jar工程 |
| pom | 构建工程  配置war工程和这些war工程依赖的jar工程 |

1. 一些固定的工程

|  |  |
| --- | --- |
| **工程名** | **说明** |
| eayun-common | 工具类  异常  过滤器  dao封装 |
| eayun-xxx-intf | 业务接口工程，只包含这个工程的Service接口和Model |
| eayun-xxx | 业务实现工程，包含Service实现和Controller |
| eayun-build | 构建工程，作为业务工程的父工程，所有的第三方工程依赖在该工程中体现 |

1. 工程的依赖

 一个业务实现工程（eayun-xxx）可以依赖多个业务接口工程（eayun-xxx-intf）。

业务接口工程也可以依赖其他业务接口工程。

业务实现工程不可以依赖其他业务实现工程。

工程依赖要避免循环依赖。

# 程序MVC架构和规范



|  |  |
| --- | --- |
| **层** | **作用** |
| Controller | 前后台的业务交互 |
| Service | 业务的实现，事务控制 |
| Dao | 数据库增删改查操作 |
| Model | 业务模型 |
| Job | 计划任务Job |
| Thread | 计划任务中的线程 |

## 包名和类名

|  |  |  |
| --- | --- | --- |
| **类型** | **规范** | **举例** |
| Controller | **包名：**com.eayun.模块名.controller  **类名：**XXController | **包名：**com.eayun.system.controller  **类名：**LoginController |
| Service接口 | **包名：**com.eayun.模块名.service  **接口名：**XXService | **接口包名：**com.eayun.system.service  **接口名：**LoginService |
| Service实现 | **实现类包名：**com.eayun.模块名.service.impl  **实现类名：**XXServiceImpl | **实现类包名：**com.eayun.system.service.impl  **实现类名：**LoginServiceImpl |
| DAO | **包名：**com.eayun.模块名.dao  **接口名：**XXDao | **包名：**com.eayun.system.dao  **接口名：**UserDao |
| Model | **包名：**com.eayun.模块名.model  **实体类名：**BaseXX  **普通类名：**XX | **包名：**com.eayun.system.model  **实体类名：**BaseUser  **普通类名：**User |
| Job | **包名：**com.eayun.模块名.job  **类名：**XXJob | **包名：**com.eayun.system.job  **类名：**UserSyncJob |
| Thread | **包名：**com.eayun.模块名.thread  **类名：**XXThread | **包名：**com.eayun.obs.thread  **类名：**ObsDetailGatherThread |
| MQ消息队列消费者 | **包名：**com.eayun.模块名.mq  **类名：**XXConsumer | **包名：**com.eayun.virtualization.mq  **类名：**com.eayun.virtualization.mq. RenewalConsumer |

## Controller

1. 一组业务操作一个类
2. 多例模式 @Scope("prototype")
3. 每个public方法匹配一个.do跳转，并返回一个Json对象
4. 只负责组织request传入的数据，并调用一个Service方法并返回结果（记录日志操作除外）

## Service

1. 一组业务操作一个类
2. 单例模式
3. 涉及数据库操作的一个Service方法是一个事务，Service方法需要用@Transactional注解。如果该service类所有的接口实现方法都是涉及数据库事务的，则该类用@Transactional注解。
4. 只能调用自己工程下的DAO，也允许调用其他Service，但不允许调用其他工程下的DAO
5. Service的方法不允许对异常进行处理，如有try-catch块，异常必须往上抛出
6. 可以在方法中组织SQL，调用到dao中的方法执行。
7. 所有可变条件用预编译方式传参数

## Dao

1. 一张数据表一个类，也可以额外根据业务要求建类
2. 接口类，继承com.eayun.common.dao.IRepository
3. 可以用注解的方式扩展HQL
4. 所有可变条件用预编译方式传参数

## Model

1. BaseModel中，类名和表名对应，每个属性对应一个字段（类名和属性名遵从驼峰命名法）
2. 普通类继承对应的实体类，并扩展自己的属性
3. 使用BeanUtils.copyPropertiesByModel实现BaseXXX和XXX之间的转换

## Job

1. 继承QuartzJobBean的计划任务
2. 非Spring管理的Bean
3. 配置文件命名为applicationContext-quartz-xxx.xml，统一放在schedule工程中。

## Thread

1. 一般来说都在Job中调用
2. 非Spring管理的Bean

## 常量

1. 全大写，单词间用下划线分隔，比如：MY\_NAME

## 变量和方法名

1. 变量和方法名必须有业务含义，禁用比如flag，list1等业务含义模糊的命名。
2. 第一个字母小写，遵循驼峰命名法则，比如：myName

## 字符串拼接

1. SQL，HQL的拼接禁止使用String类，要用StringBuffer类实现拼接。

# Dao规范

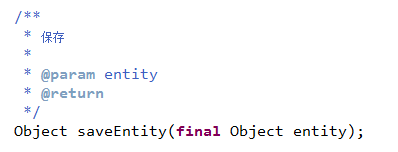
## 整体规范

1. 接口，继承Irepository
2. 如需扩展方法，用注解的方式写SQL

## HQL:增，改

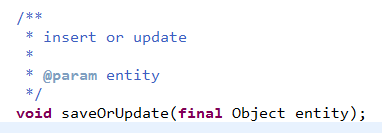
1. 新增记录：

Object saveEntity(final Object entity)：



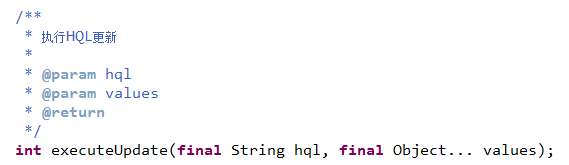
1. 修改记录，当bean是新new出来，所有的属性是手工set进去：

void saveOrUpdate(final Object entity)



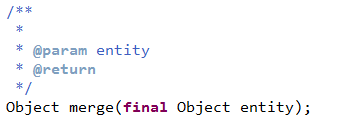
1. 通过hql只更新一部分属性，或新插入一条记录：

int executeUpdate(final String hql, final Object... values);



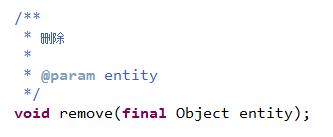
1. 修改记录，当对象是通过hibernate查询出来时：

Object merge(final Object entity);

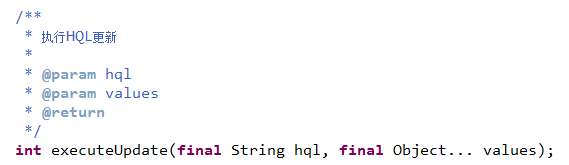


## HQL:删

1. void remove(final Object entity)



1. int executeUpdate(final String hql, final Object... values);



## HQL:查-不分页

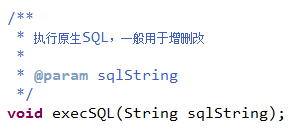
|  |
| --- |
| 1. List find(final String hql, final Object... values); |
| 1. List findBy(final String hql, Map<String, Object> map); |
| 1. Query createQuery(String hql, Object... values);      1. Query createSQLQuery(String hql); |
| 1. T findOne(ID id)   根据主键查询对象 |

## HQL:查-分页

|  |
| --- |
| 1. Page pagedQuery(final String hql, final QueryMap qm, final Object... values) |
| 1. List query(String hql, QueryMap qm, Object... values); |
| 1. Query createQuery(String hql, Object... values); |
| 1. Query createSQLQuery(String hql); |

## SQL：增删改

1. void execSQL(String sqlString)



## SQL:查，不分页

|  |
| --- |
| 1. Query createSQLNativeQuery(String sql) |
| 1. Query createSQLQuery(String hql); |

# 工程日志

1. 规范1：如何定义log对象

每个Service，Controller需要定义一个私有，静态log对象，比如：

private static final Logger log = LoggerFactory.getLogger(TestController.class);

1. 规范2：什么时候调用log方法

1.Controller层方法和Service层中接口的实现方法开始时记录：log.info（“XX业务开始”）

2.所有的Catch Exception记录：log.error(e.toString(), e);

# 异常处理

* 以下的方式是符合规范的：

1. 一般情况下，业务层不需要处理异常（无需try/catch），异常往上抛，由框架处理；
2. 如果业务需求上或代码上有必要在catch块中处理异常，处理完毕后必须在log中输出。同时根据是否需要事务回滚决定是否往上抛出（Spring框架只有截取到RunTimeException性质的Exception才会事务回滚），但是一般情况下直接继续往上抛出异常即可；
3. 一般情况下，不要在catch块中人为新建一个Exception往上抛出异常，除非明确这段逻辑需要catch异常且不作事务回滚（因为人为新建的纯Exception抛出不会引起事务回滚）。

* 不符合规范的操作：

1. 业务上catch异常并手动处理，但没有log输出以及继续抛出异常。

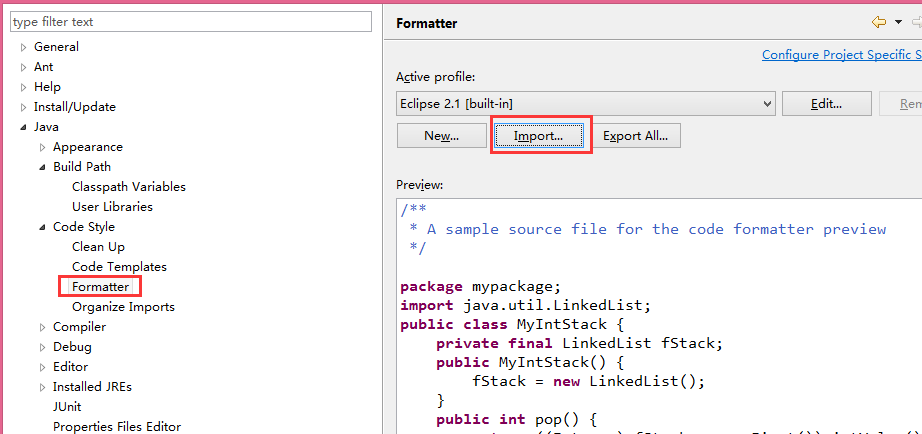
# 方法名

适用于Controller、Service和Dao

|  |  |  |
| --- | --- | --- |
| **方法类型** | **约定** | **举例** |
| 新增 | addXXX | addUser |
| 修改 | updateXXX | updateUser |
| 删除 | deleteXXX | deleteUser |
| 查询列表 | getXXXList | getUseList |
| getListByXXX | getListByCostomer |
| 查询单条记录 | findXXX | findUser |
| findXXXByXXX | findUserById |

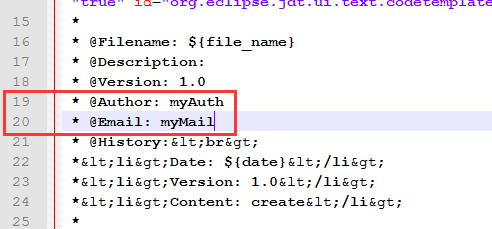
# 代码组织与风格

## 导入eayun formatter模板

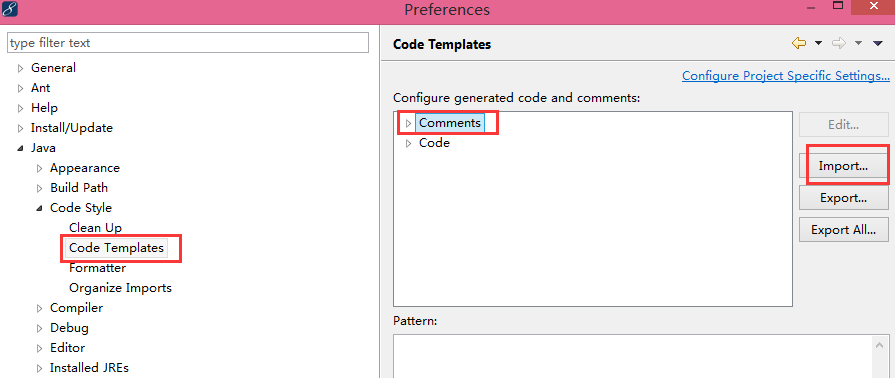


## 导入code template模板

1. 修改Auth和Email



1. 导入模板



## 缩进，间隔，风格等

Shift+Ctrl+F快捷键格式化代码即可。

# 注释

## 基本原则

1. 必须注释的有：
   1. 所有的类
   2. Controller层的public方法
   3. Service层的接口方法
   4. 逻辑复杂的private方法
   5. 方法体中逻辑复杂的部分
   6. 所有的全局静态变量
2. 注释的内容：
   1. 类上要注释这个类要做什么业务
   2. 方法要注释所有的入参和返回值的含义，业务逻辑
   3. 方法体中逻辑复杂的部分，注释其业务逻辑
   4. 静态变量要注释变脸含义
3. 错误的注释有：
   1. 注释显而易见的内容
   2. 注释“是什么”而不是“为什么”

# 声明

1. 在导入包时当完全限制代码所使用的类的名字,而少用通配符的方式，如：

import java.awt.Color;

import java.awt.Button;

1. 对于引入了但是没有用到的类或接口，必须从声明中移除。
2. 使用Shift+Ctrl+o快捷键处理声明

# RedisKey

RedisKey由固定的前缀和可变的后缀组成，其中固定的前缀在com.eayun.common.constant.RedisKey中维护，同时维护Redis设计文档。

# MongoColletionName

MongoColletionName由若干个单词组成，各单词之间用点（.）分隔，在com.eayun.common.constant.MongoCollectionName中维护，同时维护Mongo设计文档。