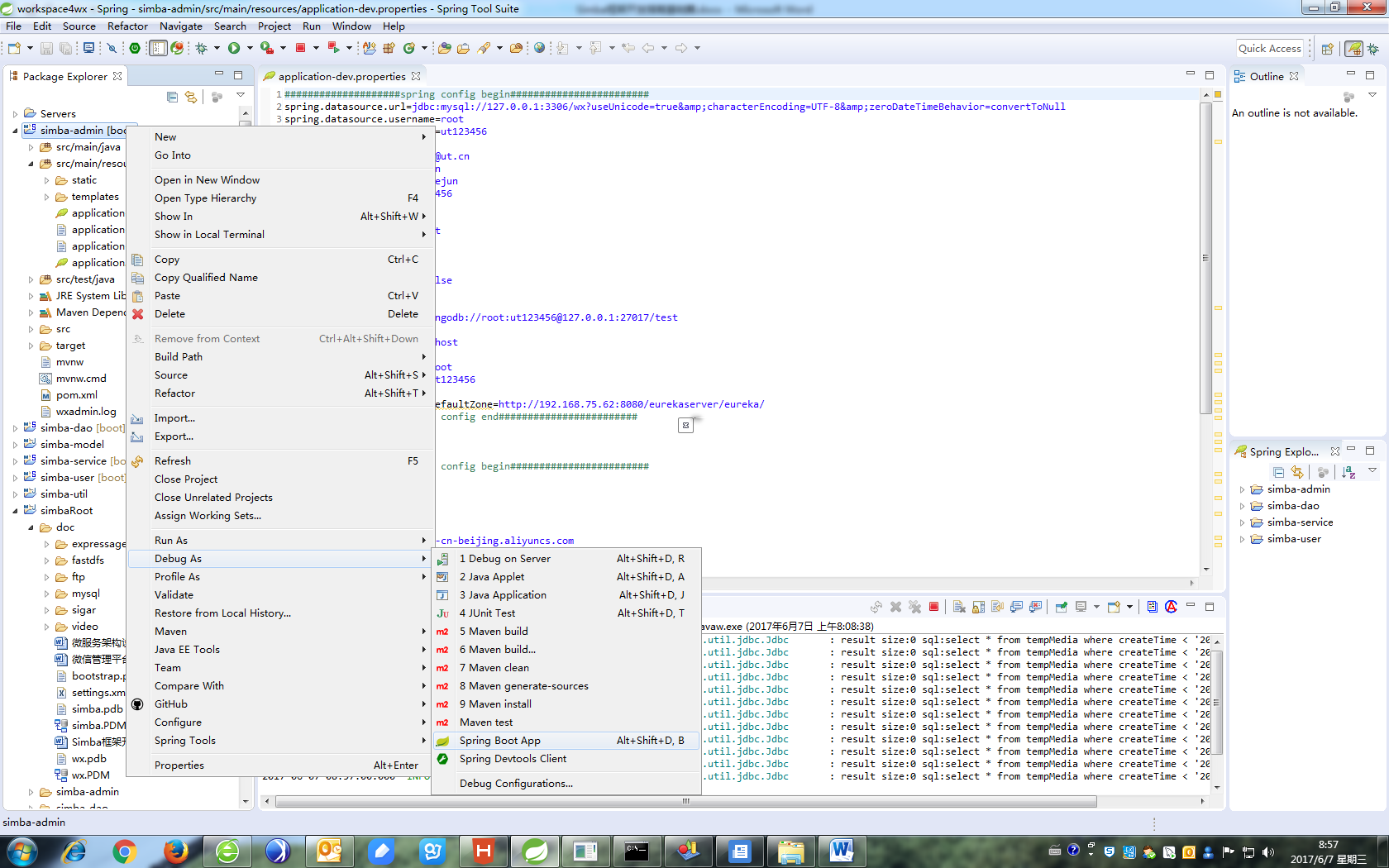
# 安装开发环境

按照《Simba框架开发环境搭建说明》文档搭建开发环境，打开配置文件application-dev.properties

修改数据库和redis服务器连接地址

# 启动服务

在需要启动的项目上，点击右键运行Spring Boot App启动服务



# 登录管理后台

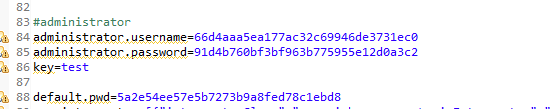
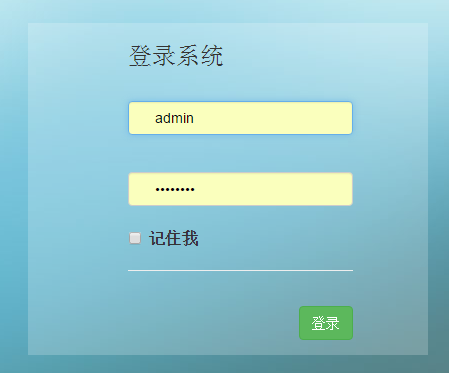
<http://127.0.0.1:8888/admin>

登录账号密码admin/admin123

如果想修改账号密码，可以打开配置文件application.properties，

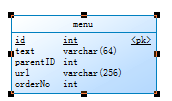
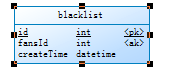
账号密码都保存的是密文，加密算法为md5(明文+key)

default.pwd为新增一个用户时的默认密码，默认为123456

# 代码生成器开发一个增删改查基础模块

1. 在建模工具中创建模型表并执行sql创建表，如下图：



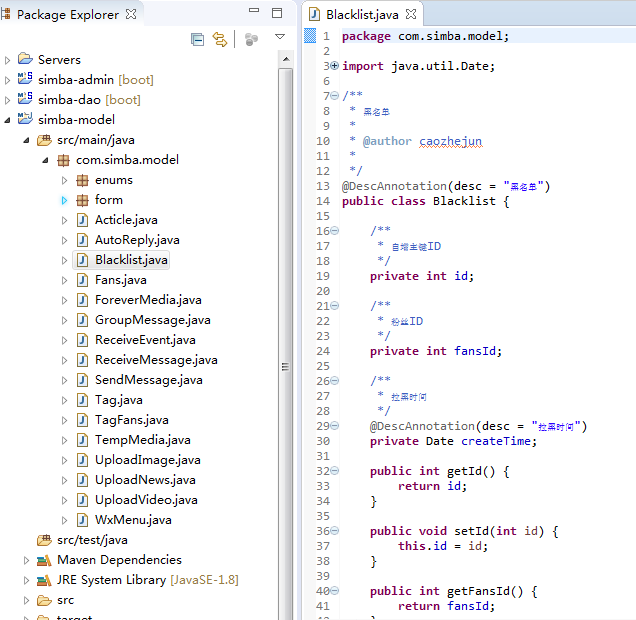
如果是普通对象，必须有一个id字段自增主键，

如果是树型结构，必须有id ,parentID,text三个字段

数据库的表名为类名首字母小写

列名与类的属性名相同

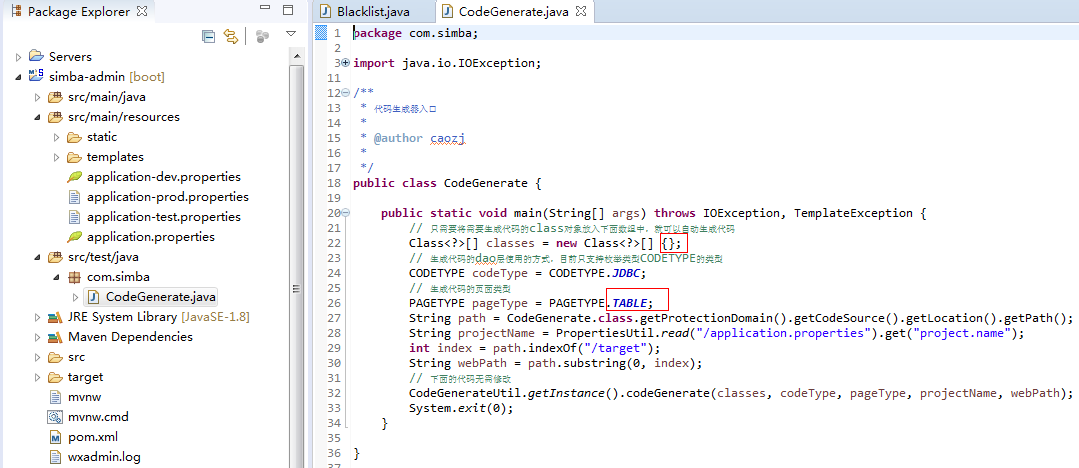
1. 在simba-model下创建对应的类，如下图：



id支持三种类型int long String

@DescAnnotation 是为了页面实现使用

1. 打开代码生成器类



将刚刚新建的类，放到第一个红色框内，支持同时生成多个对象

第二个红色框代表的是页面类型，目前支持3种，无页面、列表页、树型页

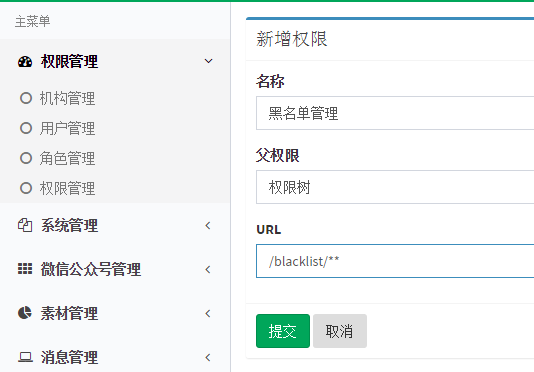


点击右键运行，自动生成好前后台代码，刷新整个项目，启动服务

1. 登录系统，新建菜单



1. 新建权限

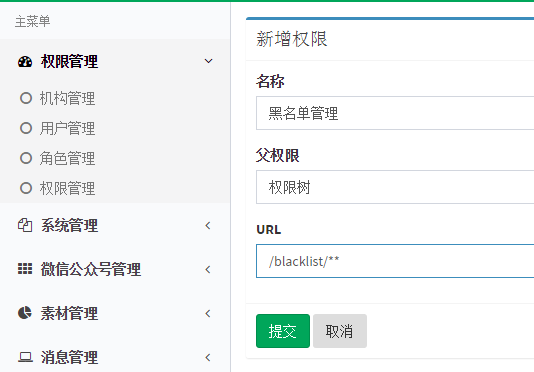
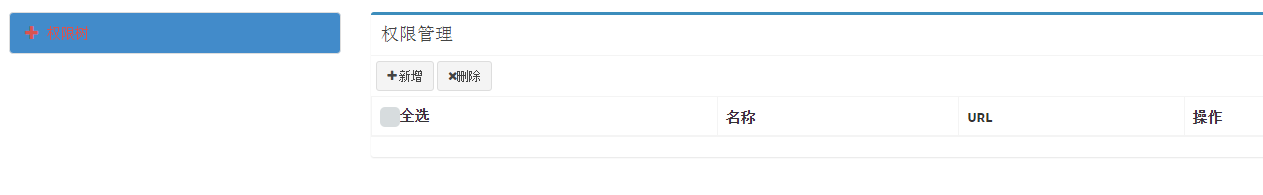


1. 将新建的权限分配给角色，拥有这个角色的用户就可以查看并操作菜单内容
2. 进入菜单，已经存在新增、修改、删除、分页功能，个性化需求，可以自己在生成的代码上修改

# 权限管理

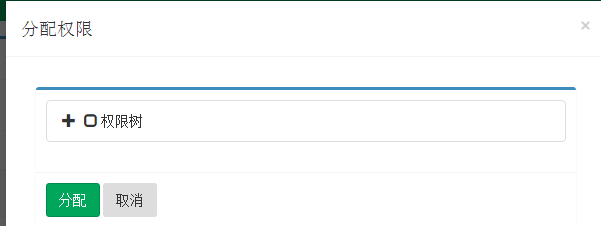
## 权限管理

URL配置支持通配符，多个用英文逗号隔开



## 角色管理

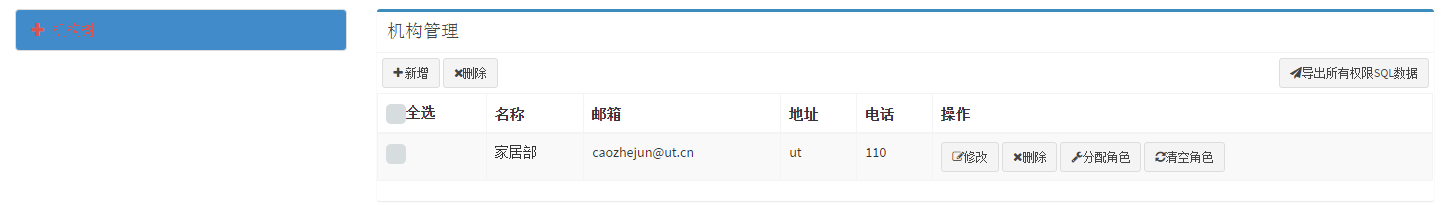
新增角色，并分配权限

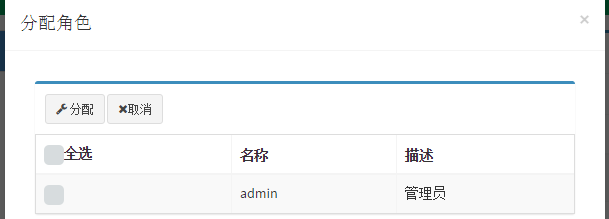


## 机构管理

新建机构，并分配角色

机构下的用户自动拥有机构所拥有的角色





## 用户管理

在机构下新建用户，并分配角色，用户拥有自己的角色+所在机构的角色，一个用户可以属于多个机构

用户拥有的所有角色对应的url，用户都可以访问，其他url都将被拒绝访问



# 拦截例外

系统目前有2个默认拦截器，登录及权限拦截器，可以通过配置文件配置哪些Url不需要被拦截



登录拦截器不拦截login.interceptor.exclude配置的url

权限拦截器不拦截login.interceptor.exclude+permission.interceptor.exclude配置的url

## 新增拦截器

新增一个类，实现接口org.springframework.web.servlet.HandlerInterceptor

配置到配置文件中



interceptorClass配置类的全路径

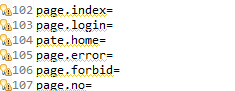
url配置拦截器要拦截的url

# 导出权限脚本

点击按钮可以导出所有权限的SQL数据，主要用于测试环境配置好的权限，导入到真实环境中

# 自定义页面

配置文件中提供了自定义页面配置方式，系统已经提供了一套默认的页面实现



page.login 登录页面

page.index 登录成功之后的页面

page.home 登录成功之后进入的页面中心的欢迎页面

page.error 异常页面

page.forbid 访问拒绝页面

page.no 找不到资源页面

# 获取登录用户信息

用户登录成功之后，会将用户相关信息保存到Session中，可以使用SessionUtil工具类获取登录用户相关信息

# 菜单管理

1. 菜单操作
2. 导出菜单SQL脚本



主要用于将开发环境配置的菜单信息导出之后，导入到真实环境中

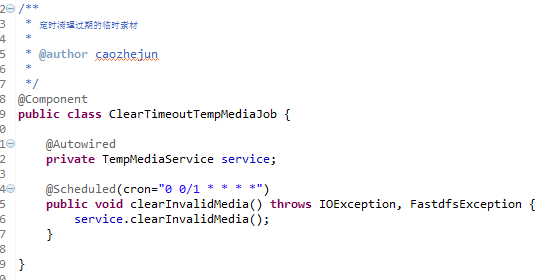
# 注册表管理

1. 创建注册类型，然后在注册类型下创建注册表，注册表的编码全局唯一

代码中通过RegistryTableData.get(key)获取对应的值，在界面上修改的值实时更新，并同步到集群中的所有服务器

# 定时器管理

1. 使用Spring Task注解方式创建定时器，如下图：



具体注解使用方式可以网上搜索

1. 使用任务管理界面配置



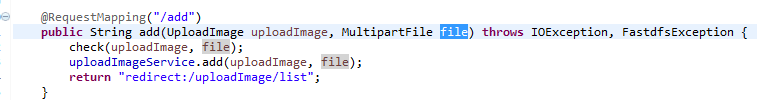
如果配置了cron表达式，优先按照cron表达式方式执行定时任务，可以在界面查看定时器执行情况并操作任务

# 上传文件

1. 在页面增加input file上传文件，如下图：



1. 后台接收文件，如下图：



参数同名就可以自动将文件解析放到对应的对象内

1. 配置统一上传文件管理组件



配置红框中的内容，目前支持本地、fastdfs、阿里云oss三种方式，如果服务器部署在阿里云，推荐使用阿里云oss



使用红框内的代码上传文件，

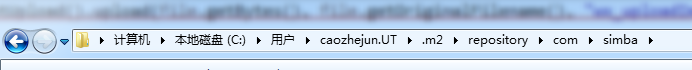
如果使用的是阿里云OSS方式，返回的url地址，可以直接访问，

如果是本地或者fastdfs，需要拼接下载文件url为/download/download?fileName=${url}

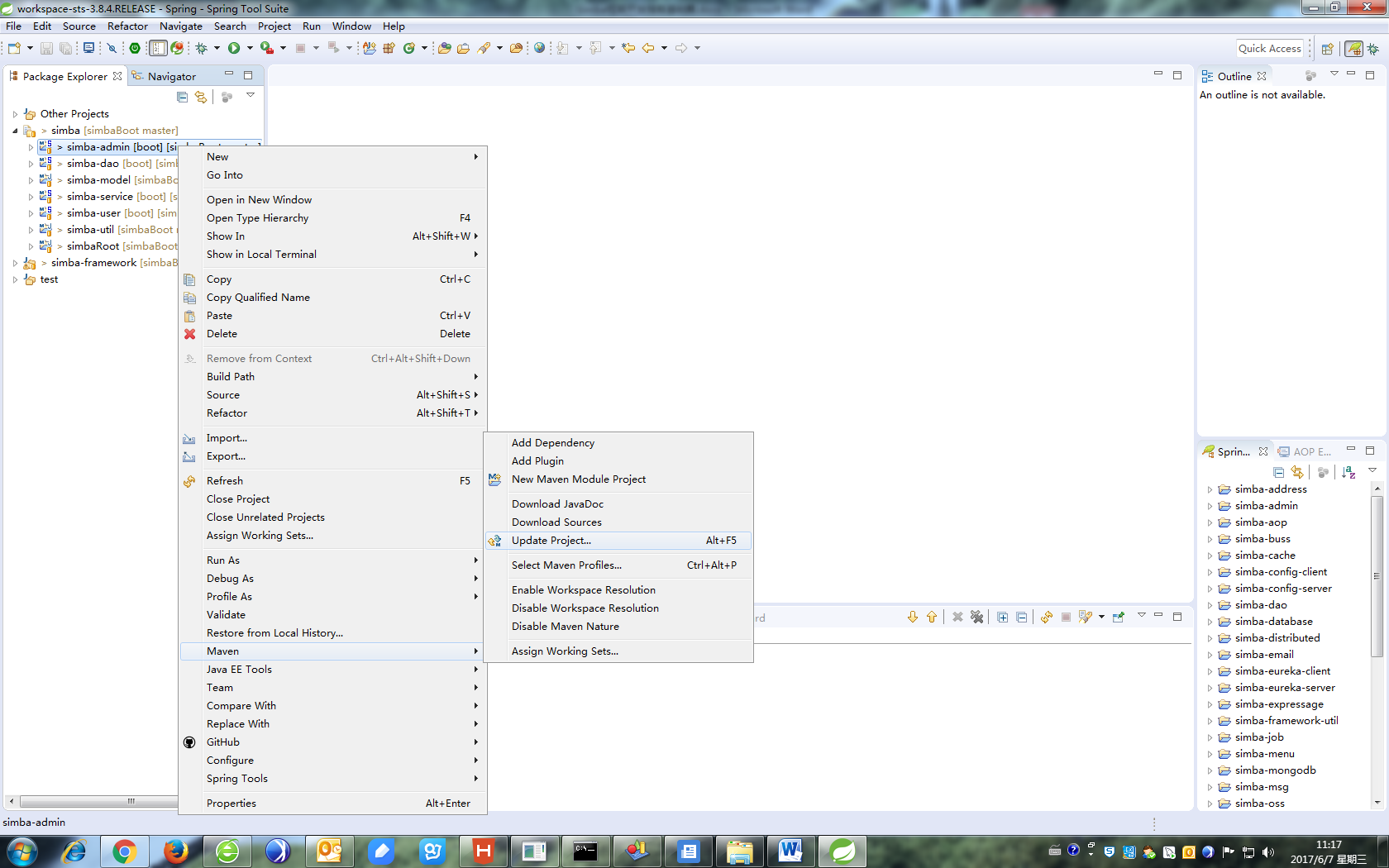
# 框架版本升级

1. 快照版本升级

找到用户目录下的.m2\repository\com\simba目录，删除下面所有的文件



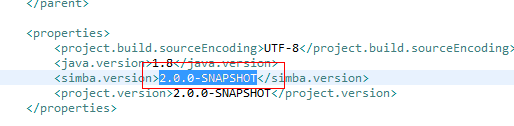
在项目点右键🡪Maven🡪Update Project



则框架代码自动更新成最新版本的快照代码

1. 稳定版本升级

直接修改simbaRoot目录下的pom.xml文件

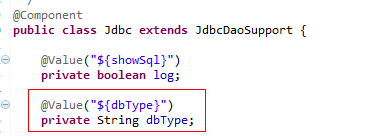


红框中内容修改成对应稳定版本的版本号，保存即可更新框架

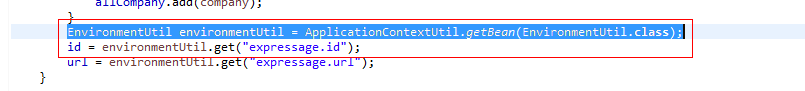
PS 如果除了导入了SimbaRoot，还导入了框架代码，需要在开发工具中删掉框架代码对应的项目才能更新框架版本，否则更新无效

# 配置文件内容获取

1. 在Bean中，使用@Value注解



1. 在非Bean中，使用工具类，如下图:



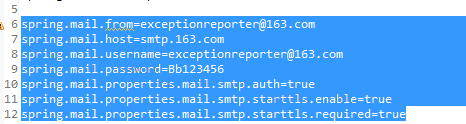
# 常用工具类

1. 操作http（com.simba.framework.util.http.HttpClientUtil）
2. 操作freemarker（com.simba.framework.util.freemarker.FreemarkerUtil）
3. 操作groovy（com.simba.framework.util.groovy.GroovyUtil）
4. 操作redis（com.simba.cache.RedisUtil）
5. 操作zip（com.simba.framework.util.zip.ZipUtil）
6. 操作Spring Bean（com.simba.framework.util.applicationcontext.ApplicationContextUtil）
7. 操作拼音（com.simba.framework.util.common.PinYinUtil）
8. 操作反射（com.simba.framework.util.common.ReflectUtil）
9. 操作对象序列化（com.simba.framework.util.common.SerializeUtil）
10. 操作线程数据（com.simba.framework.util.data.ThreadDataUtil）
11. 操作properties文件（com.simba.framework.util.file.PropertiesUtil）
12. 操作json（com.simba.framework.util.json.FastJsonUtil）
13. 操作xml（com.simba.framework.util.common.XmlUtil）
14. 更多的工具类，请查看框架源码

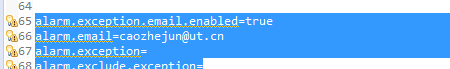
# 异常邮件提示

当系统由于用户web请求发生异常时，可以自动将异常信息堆栈发送到指定邮箱，用于异常监控及发现系统bug（系统后台线程或者定时任务发生的异常无法发送邮件）

在系统配置文件application-dev.properties中如下图配置



发送邮件的邮箱账号密码信息



alarm.exception.email.enabled: 是否发送邮件

alarm.email：接收邮件的地址，多个邮箱地址使用英文逗号隔开

alarm.exception：需要发送邮件的异常类，使用类的全路径，不配则拦截所有异常

alarm.exclude.exception：不需要发送邮件的异常类，使用类的全路径，不配则没有例外异常

# 打印sql语句

配置文件application.properties中配置是否打印出所有执行的sql

