Simba框架使用文档

目录

[1.版本号 5](#_Toc501639135)

[2.文档说明 5](#_Toc501639136)

[2.1功能描述 5](#_Toc501639137)

[2.2阅读对象 5](#_Toc501639138)

[3.Simba框架下后台开发流程 5](#_Toc501639139)

[3.1搭建环境 5](#_Toc501639140)

[3.2新建开发项目 7](#_Toc501639141)

[3.3配置开发环境 7](#_Toc501639142)

[3.4进行业务开发 7](#_Toc501639143)

[3.5项目调试和启动 7](#_Toc501639144)

[4.框架使用说明 8](#_Toc501639145)

[4.1登录管理后台 8](#_Toc501639146)

[4.2使用代码生成器开发一个增删改查基础模块 9](#_Toc501639147)

[4.2.1建数据表 9](#_Toc501639148)

[4.2.2建类 9](#_Toc501639149)

[4.2.3自动生成 10](#_Toc501639150)

[4.2.4新建菜单和权限 11](#_Toc501639151)

[4.3操作管理菜单 12](#_Toc501639152)

[4.3.1权限管理操作 12](#_Toc501639153)

[4.3.2角色管理操作 13](#_Toc501639154)

[4.3.3机构管理操作 13](#_Toc501639155)

[4.3.4用户管理操作 14](#_Toc501639156)

[4.4拦截例外 14](#_Toc501639157)

[4.4.1新增拦截器 14](#_Toc501639158)

[4.5自定义页面 15](#_Toc501639159)

[4.6导出权限脚本 15](#_Toc501639160)

[4.7获取用户登录信息 15](#_Toc501639161)

[4.8菜单管理 16](#_Toc501639162)

[4.9机构扩展属性管理 16](#_Toc501639163)

[4.10用户扩展属性管理 16](#_Toc501639164)

[4.11注册表管理 17](#_Toc501639165)

[4.12定时器管理 17](#_Toc501639166)

[4.12.1创建定时器 17](#_Toc501639167)

[4.12.2查看定时器 17](#_Toc501639168)

[4.13上传文件 18](#_Toc501639169)

[4.14框架版本升级 20](#_Toc501639170)

[4.14.1快照版本升级 20](#_Toc501639171)

[4.14.2稳定版本升级 20](#_Toc501639172)

[4.15配置文件内容获取 20](#_Toc501639173)

[4.16常用工具类 21](#_Toc501639174)

[4.17异常邮件提示 21](#_Toc501639175)

[4.18数据库模型及打印sql语句 22](#_Toc501639176)

[4.19微服务注册中心 22](#_Toc501639177)

[4.19.1微服务服务端开发注意 23](#_Toc501639178)

[4.19.2微服务客户端使用注意 24](#_Toc501639179)

[4.20 Redis的使用 25](#_Toc501639180)

[4.21 数据库连接池Druid 27](#_Toc501639181)

[4.22 Json格式转换FastJson 27](#_Toc501639182)

[4.22微信相关开发 27](#_Toc501639183)

[4.22.1微信公众号 27](#_Toc501639184)

[4.22.2微信硬件平台 31](#_Toc501639185)

[4.23WebSocket相关开发 32](#_Toc501639186)

[4.23.1WebSocket使用Demo 32](#_Toc501639187)

[4.23.2WebSocket的应用场景 33](#_Toc501639188)

[4.23.3WebSocket的使用说明 33](#_Toc501639189)

[4.24集成Ueditor在线编辑器 35](#_Toc501639190)

[4.25 快递查询 35](#_Toc501639191)

[4.26 FTP工具类 36](#_Toc501639192)

[4.27 Groovy 36](#_Toc501639193)

[4.28 Spring Cache 38](#_Toc501639194)

[4.29 地址联动 39](#_Toc501639195)

[4.30 截图 40](#_Toc501639196)

[4.31 Thirft 40](#_Toc501639197)

[4.32 集群执行功能 40](#_Toc501639198)

[4.33 Spring常用注解 41](#_Toc501639199)

[4.34分布式session 42](#_Toc501639200)

[4.35 自定义注解 42](#_Toc501639201)

[4.36 前端使用说明 43](#_Toc501639202)

1.版本号

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 完成日期 | 完成者 | 联系方式 | 说明 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

2.文档说明

2.1功能描述

本文档主要描述Simba框架的使用方法。

2.2阅读对象

本文档主要为在Simba框架下开发的后台开发人员编写。目的是为后台开发人员能尽快熟悉框架的使用而编写。

3.Simba框架下后台开发流程

3.1搭建环境

搭建环境按照如下的步骤进行：

1. 准备安装包 Jdk 1.8 maven3 mysql5.7 STS redis3 Hbuilder

2. 安装好以上安装包

3. 配置环境变量MAVEN\_HOME ,PATH ,JAVA\_HOME

4. 将doc目录下的setting.xml文件覆盖到MAVEN\_HOME/conf目录下

5. 配置环境变量spring.profiles.active（linux下使用SPRING\_ PROFILES\_ ACTIVE）,开发环境设置成dev，真实环境设置成prod，不同环境会加载不同的配置文件

6. 将项目导入STS 一个已经存在的maven项目

7. 业务代码全部在SimbaRoot中，其他的模块都是框架源码

8. 如果只是开发业务项目，只导入SimbaRoot即可

9. 创建数据库执行doc/mysql目录下的初始化sql脚本

10. 修改配置文件启动服务即可

如果将所有框架代码全部导入，可以使用working sets管理，如图3.1，项目结构会比较清晰。

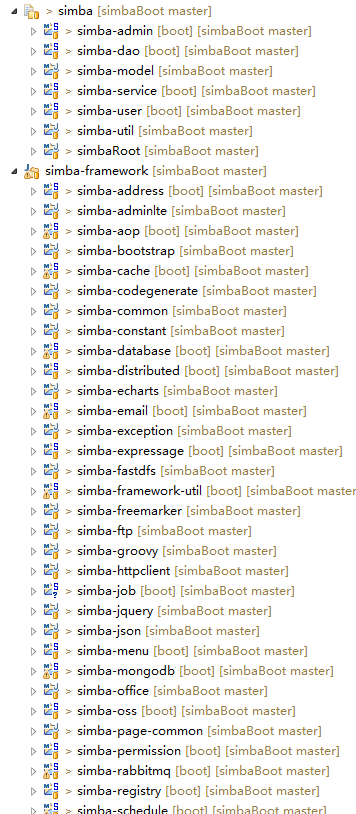


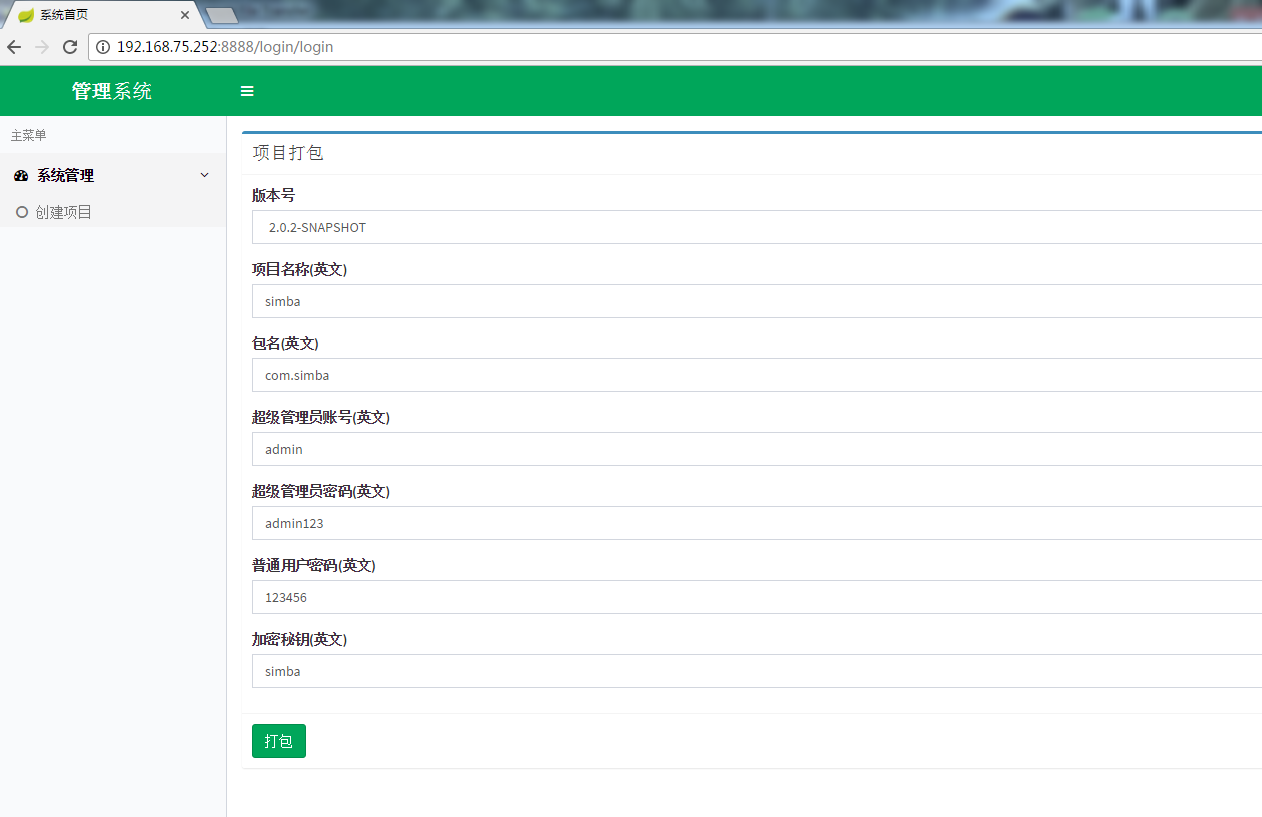
图3.1

3.2新建开发项目

访问地址   <http://192.168.75.252:8888/>

账号密码   a/1

登录系统之后，选择框架版本，设置相关参数（以前提供的框架都只能按照默认的配置，修改起来比较麻烦），目前系统提供项目名称，默认包名等自定义选项，打包完成之后，直接解压文件，将解压后的目录导入Eclipse/STS等IDE中，即可开始开发业务项目（框架相关说明文档在doc目录下）



3.3配置开发环境

配置开发环境主要是配置数据库和redis服务器，在simba-admin|simba-user/src/main/resource目录下，打开文件application-\*.properties，按照图3.2指示配置数据库和redis服务器连接地址。

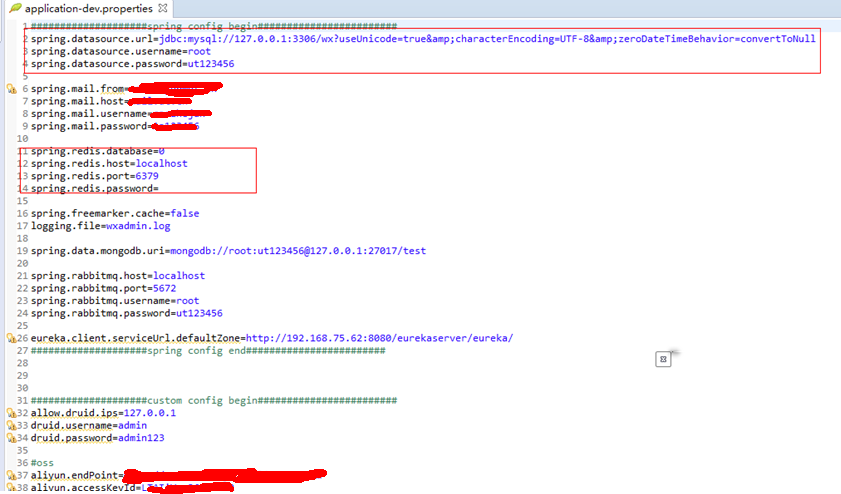


图3.2

3.4进行业务开发

在simba-admin|simba-user/src/main/java/com/simba/controller目录下新建java文件，进行开发。

3.5项目调试和启动

首先启动redis（自行下载即可）服务，然后如图3.3，在需要启动的项目上，点击右键运行Spring Boot App启动服务即可。

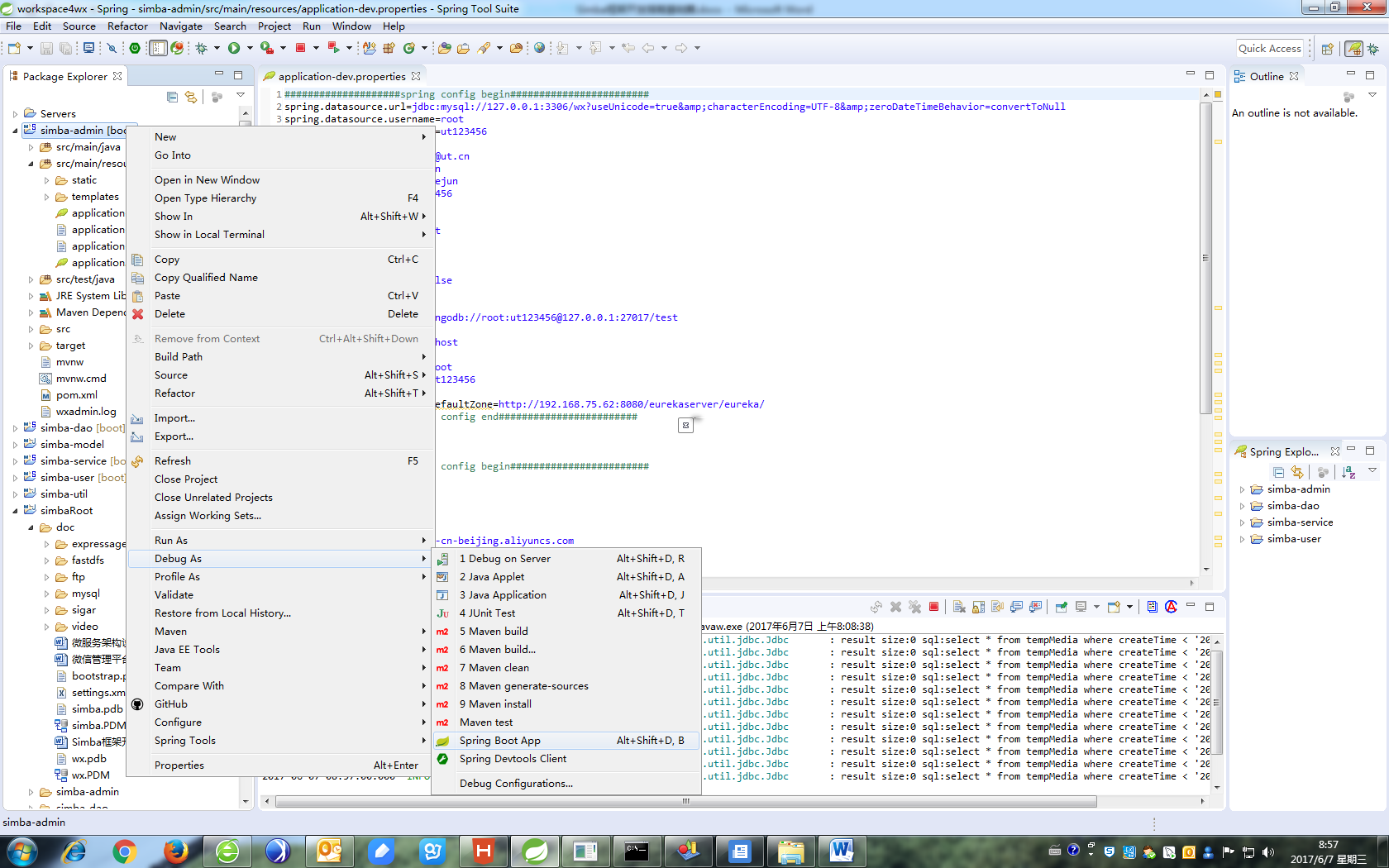


图3.3

4.框架使用说明

使用Simba框架开发新项目的过程中，需要注意的部分和使用方法。

4.1登录管理后台

登录后台的网址为<http://127.0.0.1:XXXX>，其中XXXX表示该服务启动后的端口号，可以simba-admin/src/main/resource目录下application.properties文件中自行配置，例如：server.port=8400。登录账号密码admin/admin123，登录成功后界面显示如图4.1。



图4.1

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（如图4.2），默认为123456。

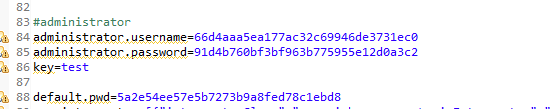


图4.2

4.2使用代码生成器开发一个增删改查基础模块

按照如下的步骤操作。

4.2.1建数据表

在建模工具中创建模型表并执行sql创建表。如图4.3和图4.4

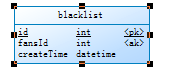


图4.3

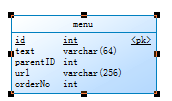


图4.4

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。

4.2.2建类

在simba-model下创建对应的类，如图4.5

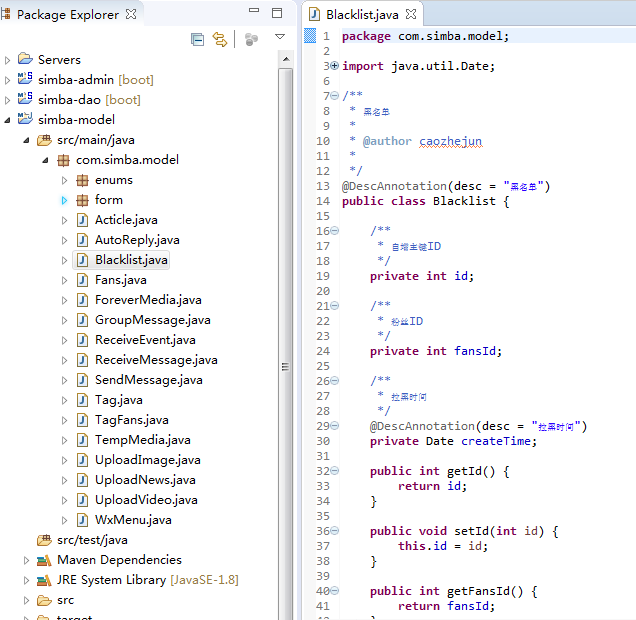


图4.5

其中id支持三种类型int long String，@DescAnnotation 是为了页面实现使用。

4.2.3自动生成

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，如图4.6。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，如图4.7。

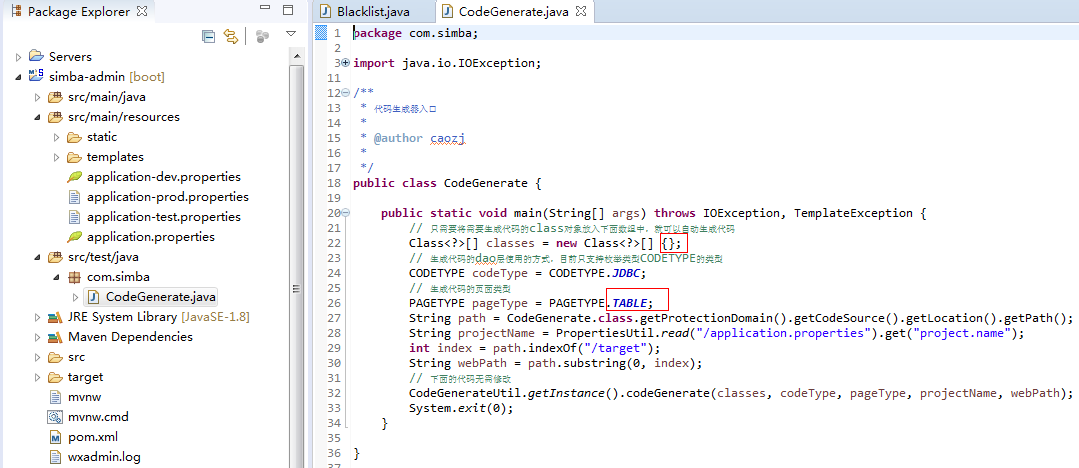


图4.6



图4.7

点击右键运行即可自动生成好前后台代码，刷新整个项目，启动服务，即可看到效果。

4.2.4新建菜单和权限

服务启动之后即可打开首页，登录系统，新建菜单和新建权限操作分别如图4.8和图4.9



图4.8

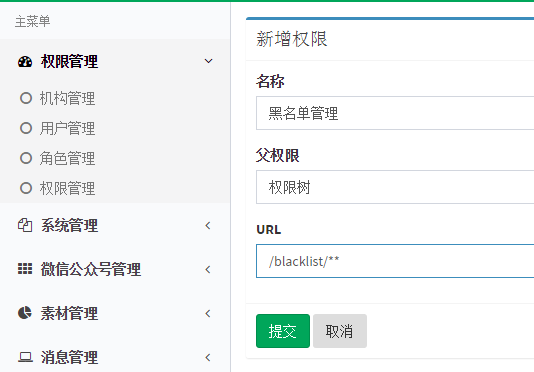


图4.9

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，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。

4.3操作管理菜单

包括权限、角色、机构和用户管理操作。

4.3.1权限管理操作

URL配置支持通配符，多个用英文逗号隔开，具体操作如图4.10和图4.11。

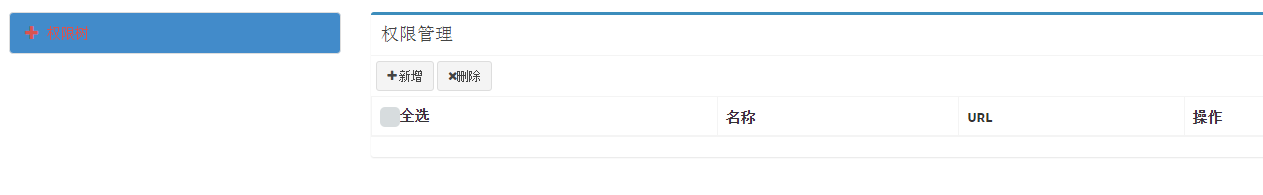


图4.10

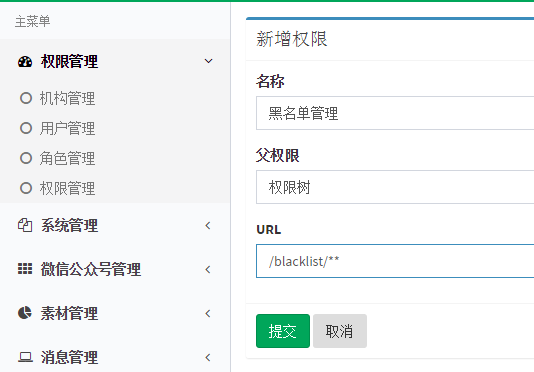


图4.11

4.3.2角色管理操作

新增角色，并分配权限，具体操作如图4.12和图4.13。



图4.12

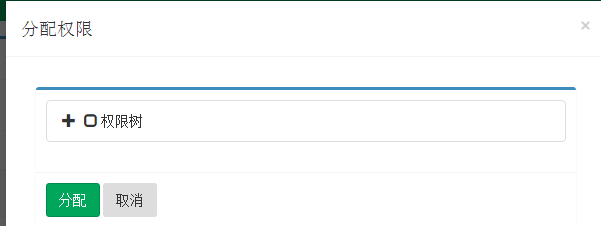


图4.13

4.3.3机构管理操作

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，具体操作如图4.14和图4.15。

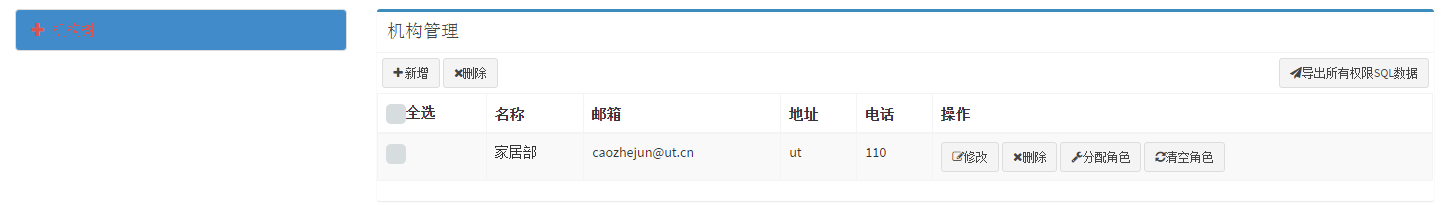


图4.14

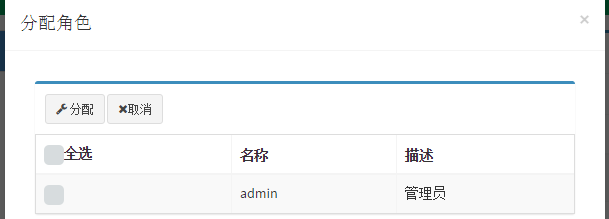


图4.15

4.3.4用户管理操作

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。具体操作如图4.16。



图4.16

4.4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。示例如图4.17



图4.17

4.4.1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，示例如图4.18。

图4.18

4.5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，如下图4.19。

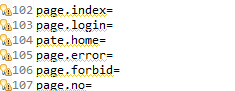


图4.19

其中:

page.login 登录页面，

page.index为登录成功之后的页面，

page.home为登录成功之后进入的页面中心的欢迎页面，

page.error为异常页面，

page.forbid为访问拒绝页面，

page.no为找不到资源页面。

4.6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，示例如图4.20。



图4.20

4.7获取用户登录信息

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。

4.8菜单管理

导出菜单SQL脚本操作如图4.21。



图4.21

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。

4.9机构扩展属性管理

通过配置文件configs.properties的org.ext动态扩展机构属性。如图4.21（1）。结构为key1:value1,key2:value2。key为数据库的字段名，value为页面显示的描述，key如果以\_r结尾，代表此字段必填，如果数据库中没有此字段，会在服务启动的时候，自动新增。

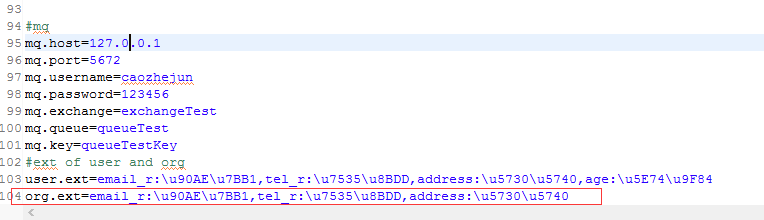


图4.21（1）

4.10用户扩展属性管理

通过配置文件configs.properties的user.ext动态扩展用户属性。如图4.21（2）。结构为key1:value1,key2:value2。key为数据库的字段名，value为页面显示的描述，key如果以\_r结尾，代表此字段必填。如果数据库中没有此字段，会在服务启动的时候，自动新增。

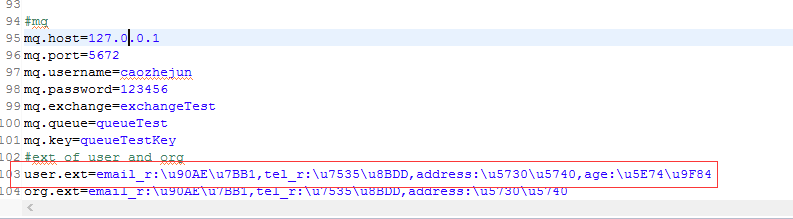


图4.21（2）

4.11注册表管理

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。

4.12定时器管理

4.12.1创建定时器

使用Spring Task注解方式创建定时器，操作如图4.22。

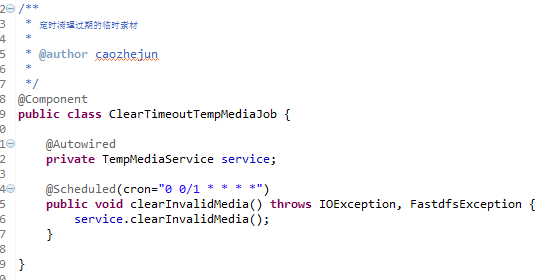


图4.22

具体注解使用方式可以网上搜索。

4.12.2查看定时器

任务管理界面配置。如图4.23。



图4.23

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。如图4.24。



图4.24

4.13上传文件

上传文件按照如下步骤实现。

1、在页面增加input file上传文件，如图4.25



图4.25

2、在后台接收文件，操作如图4.26。

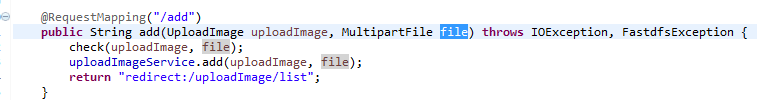


图4.26

参数同名就可以自动将文件解析放到对应的对象内

3、配置统一上传文件管理组件，操作如图4.27。

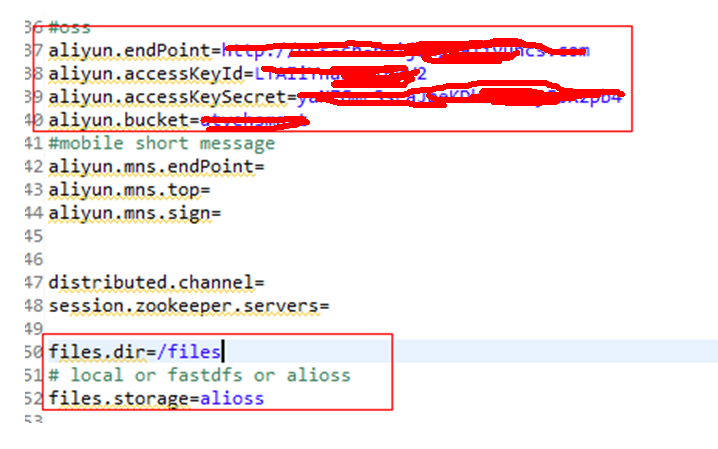


图4.27

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。



图4.28

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}。

4.14框架版本升级

分为快照版本升级和稳定版本升级。

4.14.1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。示例如图4.30。

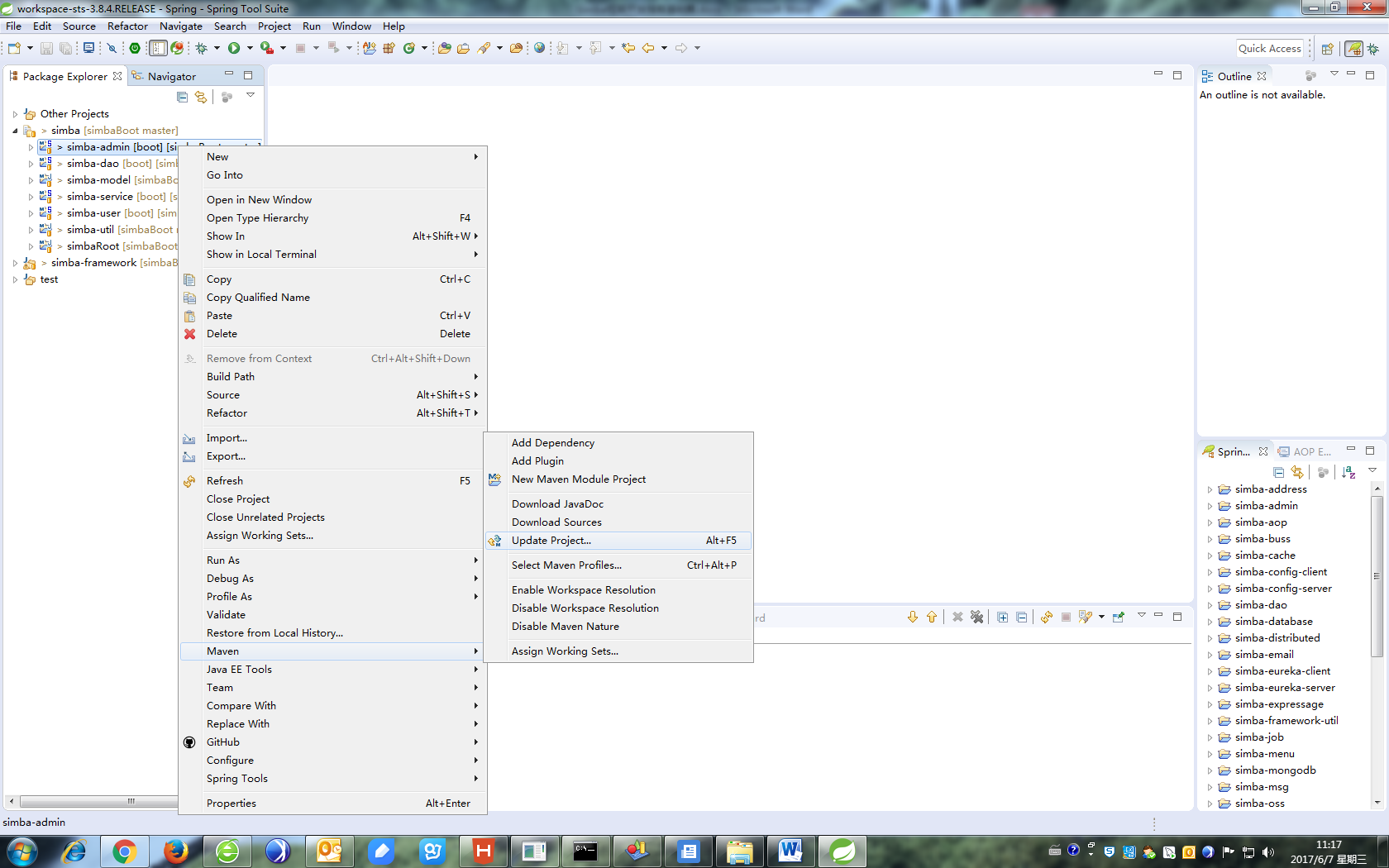


图4.30

4.14.2稳定版本升级

直接修改simbaRoot目录下的pom.xml文件，图4.31中红框里的内容修改成对应稳定版本的版本号，保存即可更新框架，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

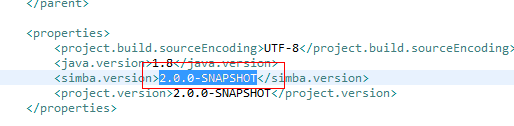


图4.31

4.15配置文件内容获取

在Bean中，使用@Value注解，操作如图4.32。

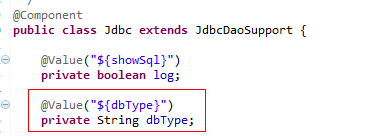


图4.32

在非Bean中，使用工具类，操作如图4.33。

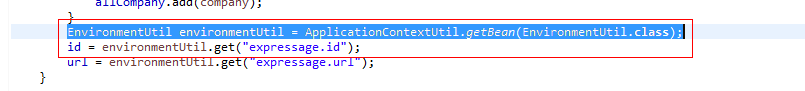


图4.33

4.16常用工具类

操作http（com.simba.framework.util.http.HttpClientUtil）

操作freemarker（com.simba.framework.util.freemarker.FreemarkerUtil）

操作groovy（com.simba.framework.util.groovy.GroovyUtil）

操作redis（com.simba.cache.RedisUtil）

操作zip（com.simba.framework.util.zip.ZipUtil）

操作Spring Bean（com.simba.framework.util.applicationcontext.ApplicationContextUtil）

操作拼音（com.simba.framework.util.common.PinYinUtil）

操作反射（com.simba.framework.util.common.ReflectUtil）

操作对象序列化（com.simba.framework.util.common.SerializeUtil）

操作线程数据（com.simba.framework.util.data.ThreadDataUtil）

操作properties文件（com.simba.framework.util.file.PropertiesUtil）

操作json（com.simba.framework.util.json.FastJsonUtil）

操作xml（com.simba.framework.util.common.XmlUtil）

更多的工具类，请查看框架源码

4.17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中如图4.34配置发送邮件的邮箱账号密码信息。

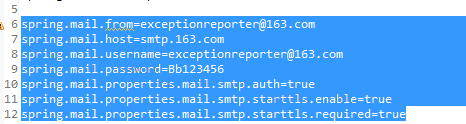


图4.34

更多配置信息如图4.35。

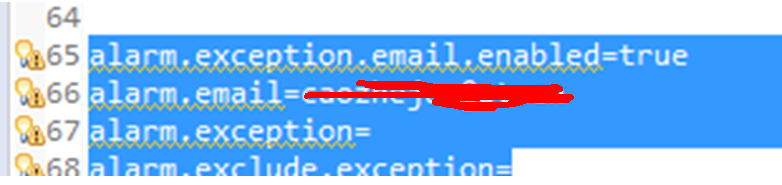


图4.35

其中

alarm.exception.email.enabled: 是否发送邮，

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。

4.18数据库模型及打印sql语句

数据库模型在安装powerdesigner之后，打开doc目录中的pdm文件即可看到

配置文件application.properties中配置是否打印出所有执行的sql，示例如图4.36。



图4.36

4.19微服务注册中心

微服务打包框架项目为simba-eureka-server，启动服务的方法为java –jar。

4.19.1微服务服务端开发注意

1、pom文件增加依赖

1）、simbaRoot下的pom.xml增加：

<dependency>

<groupId>com.simba</groupId>

<artifactId>simba-eureka-client</artifactId>

<version>${simba.version}</version>

</dependency>

2）、simba-user下的pom.xml增加：

<dependency>

<groupId>com.simba</groupId>

<artifactId>simba-eureka-client</artifactId>

</dependency>

2 、修改simba-user下的配置文件application-dev.properties，修改成自己的微服务注册中心地址，只需要修改ip地址即可。例如：

eureka.client.serviceUrl.defaultZone=http://127.0.0.1:9999/eureka/

3、服务端代码的实现和普通的springmvc的controller代码一样编写。如图4.37



图4.37

4.19.2微服务客户端使用注意

1、pom文件增加依赖

1）、simbaRoot下的pom.xml增加：

<dependency>

<groupId>com.simba</groupId>

<artifactId>simba-eureka-client</artifactId>

<version>${simba.version}</version>

</dependency>

2）、需要编写客户端代码的项目module下的pom.xml文件增加

<dependency>

<groupId>com.simba</groupId>

<artifactId>simba-eureka-client</artifactId>

</dependency>

2、修改web工程项目（simba-user或者simba-admin）下的配置文件application-dev.properties，修改成自己的微服务注册中心地址，只需要修改ip地址即可，例如：

eureka.client.serviceUrl.defaultZone=http://127.0.0.1:9999/eureka/

3、编写客户端调用代码

1）、使用Get方式调用，具体写法如图4.38.

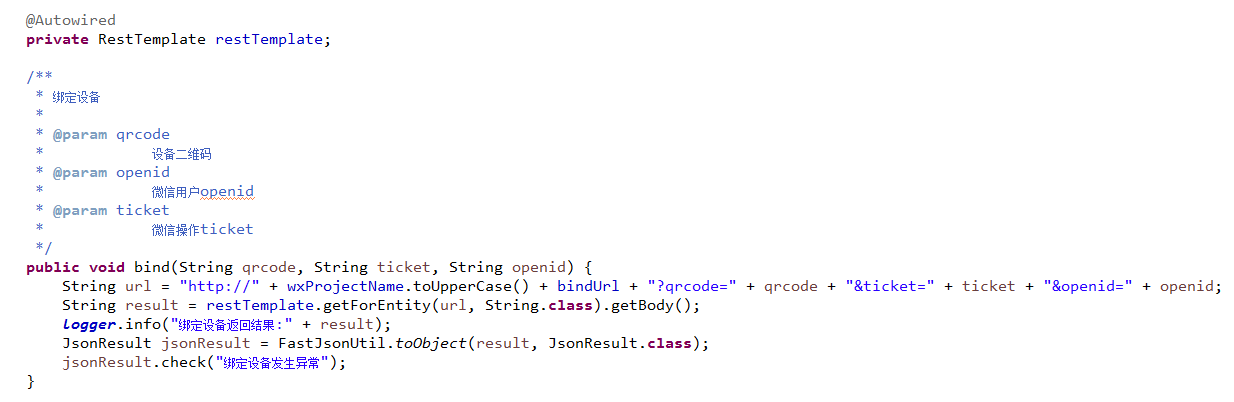


图4.38

2）、使用Post方式调用，具体写法如图4.39



图4.39

其中项目名称为配置文件application.properties中的在代码调用（如图4.40）时，必须转换为大写。



图4.40

4.20 Redis的使用

修改configs.properties中的redis配置如图4.41。

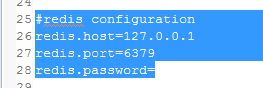


图4.41

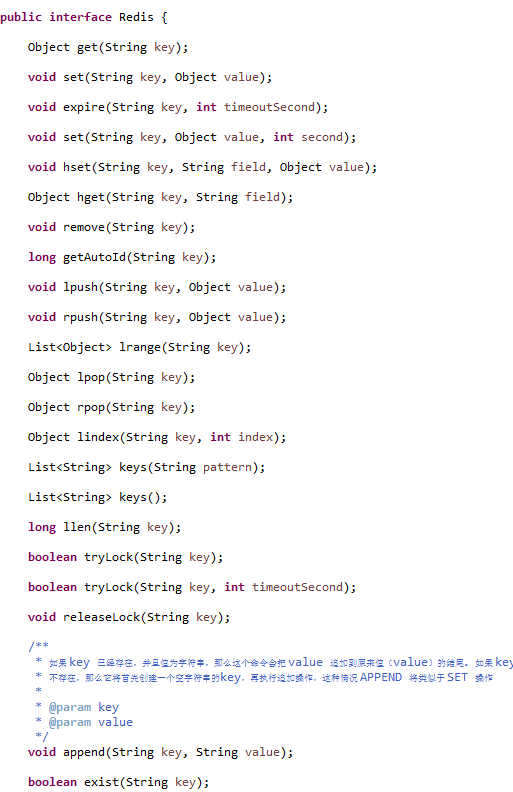


图4.42

4.21 数据库连接池Druid

系统使用的数据库连接池为druid，相关的监控配置，已经配置好了，可以直接进入druid的管理端查看数据库监控数据（关于druid的管理的使用相关信息可以网上找到）。

4.22 Json格式转换FastJson

json方式在原来的jackson的基础上，增加了fastjson实现方式直接调用FastJsonUtil类，即可使用。

4.22微信相关开发

在配置文件application.properties中配置微信相关信息。例如图4.43（1）。

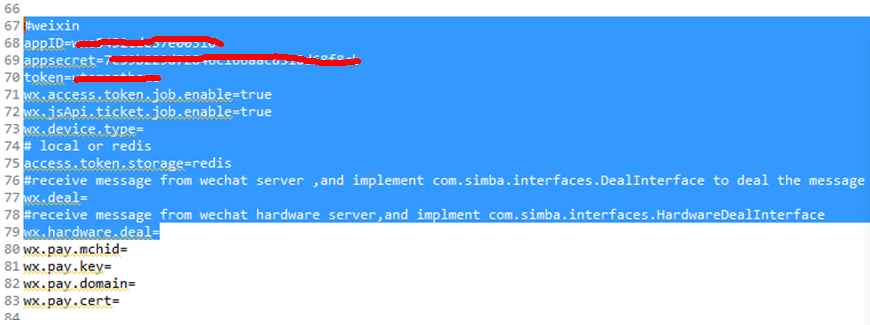


图4.43（1）

4.22.1微信公众号

1、配置服务器地址（微信公众平台）。微信公众号所有的消息和事件都会发送到这个url。例如图4.44

图4.44

实现业务逻辑处理接收的微信消息实现接口com.simba.interfaces. DealInterface。修改configs.properties文件,将实现类的完整路径配置到wx.deal如图4.45



图4.45

图4.46中，每个接口方法都代表了一种微信服务器发来的消息或者事件(具体含义可以查看源码注释)，这些消息或者事件都是异步处理的。先返回success给微信服务器，告诉微信服务器，我们已经接收成功，然后线程池执行业务逻辑。

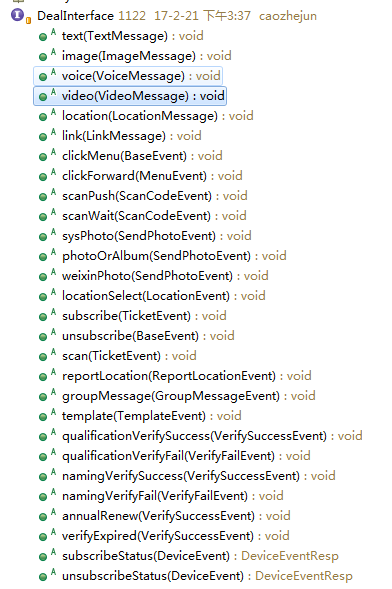


图4.46

目前框架内已经处理了所有微信公众号开发文档中提供的消息和事件，但是微信将来可能会新增消息和事件，框架同样提供了一种简单的扩展方式。如图4.47。

图4.47

其中，xml文件中的events放的是微信发过来的事件，msgs放的是消息；对于事件type代表的是微信发过来的XML，消息中的Event字段的值对于消息，type代表的是微信发过来的XML消息中的MsgType字段的值，parser是用于解析xml成对象的类，必须实现接口com.yehsmart.interfaces.ParseWxXML返回对象将成为dealMethod方法的参数值，新增一个dealMethod，就必须在com.yehsmart.interfaces.DealInterface中新增一个对应的方法。操作如图4.48

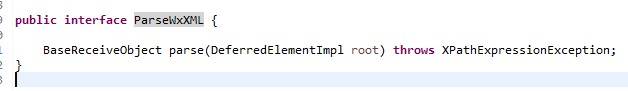






图4.48

发送请求操作微信，需要修改configs.properties文件，操作如图4.49。其中，appID appsecret token都需要在微信公众平台获取，wx.access.token.job.enable代表是否开启定时任务，每小时获取access\_token，wx.jsApi.ticket.job.enable代表是否开启定时任务，每小时获取js\_ticket ，access.token.storage代表获取的access\_token和js\_ticket存在本地内存中还是redis中，如果是单机，则两个job推荐设置为true，storage设置为local，如果为集群，则两个job可以选择某几台开启，storage必须设置为redis，如果全部job不开启，框架也做了处理，调用微信请求一样能使用，但是开启job性能更佳。

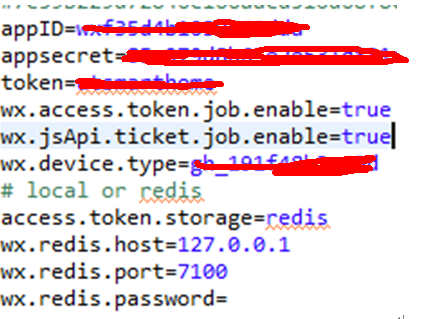


图4.49

2、微信公众号开发文档中所有的请求接口都可以在send包（如图4.50）下找到对应的实现，具体请查看源码注释。

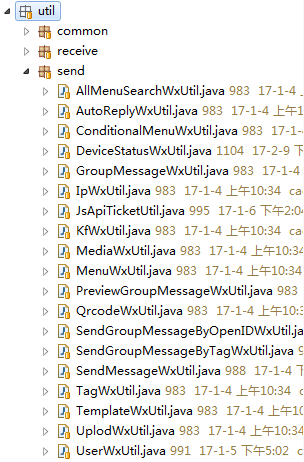


图4.50

3、H5页面使用微信JsSDK页面引入js：

<script type="text/javascript" src="服务器访问路径/js/wx/wxconfig.js"></script>如图4.51调用wx.config，其他使用详情查看微信开发文档jssdk wxconfig.js只是将签名部分搞定，其他jssdk方法调用，还得自己写。



图4.51

4.22.2微信硬件平台

1、配置服务器地址（微信公众平台—设备管理—设置）



图4.52

图4.52中的url会接收所有微信硬件平台发送过来的事件。接收事件的Controller如图4.53



图4.53

实现业务逻辑处理接收的微信硬件平台消息，实现接口com.simba. interfaces. HardwareDealInterface，修改configs.properties文件,将实现类的完整路径配置到wx.hardware.deal，如图4.54，图4.55。



图4.54

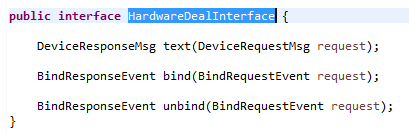


图4.55

图4.55中text:设备发送的文本消息，bind:设备与微信绑定事件，unbind：设备与微信解绑事件。

2、目前框架内已经处理了所有微信硬件平台开发文档中提供的消息和事件，但是微信，硬件平台将来可能会新增消息和事件，框架同样提供了一种简单的扩展方式，如图4.56。

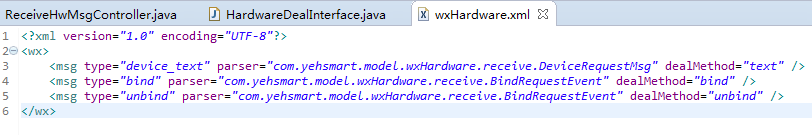


图4.56

xml文件中的msg放的是微信硬件平台发过来的消息；对于type代表的是微信硬件平台发过来的Json消息中的msg\_type字段的值，parser是用于解析json成对象的类，使用的是FastJson解析方式，parser定义一个与json对应格式的类，将类的完整路径填入，转化后的对象将成为dealMethod方法的参数值

，新增一个dealMethod，就必须在com.yehsmart.interfaces. HardwareDealInterface中新增一个对应的方法，在方法的实现中处理业务逻辑，如图4.57。



图4.57

4.23WebSocket相关开发

4.23.1WebSocket使用Demo

图4.58是一个使用WebSocket的简单实例。

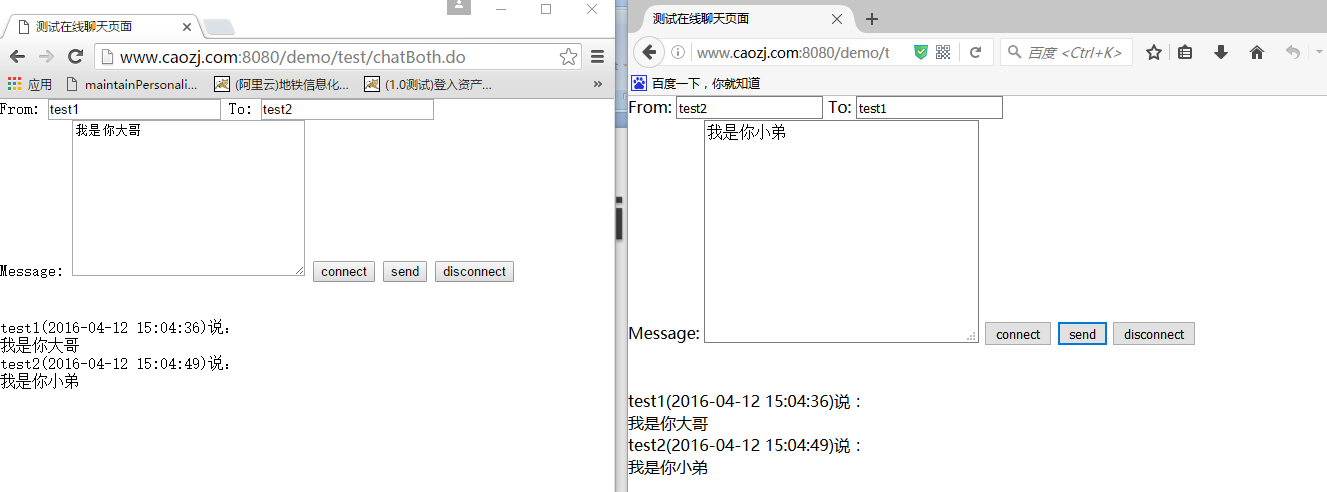


图4.58

4.23.2WebSocket的应用场景

框架中的WebSocket实现用户之间的在线聊天，实现服务端到浏览器的信息推送，代替原来的旧方案（使用浏览器定时ajax刷新）。支持IE10+，firefox，chrome等浏览器

4.23.3WebSocket的使用说明

编写自己的Handler类，例如图4.59，发送消息时的业务逻辑写在handleMessage方法中，例如图4.60。

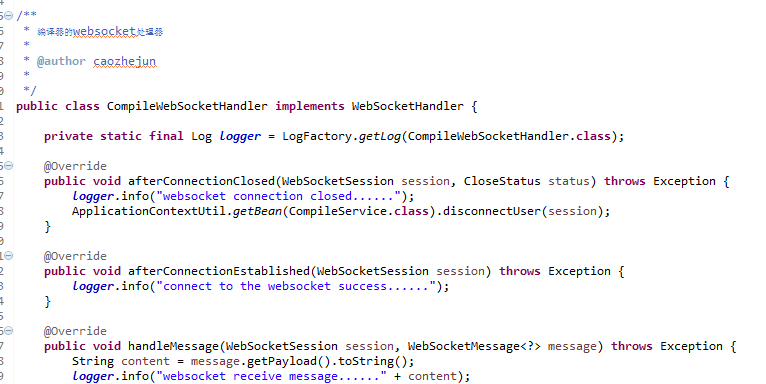


图4.59



图4.60

如果需要对发送的消息进行过滤处理，如脏话、影响社会和谐的，可以加到拦截器中，如图4.61。



图4.61

直接在configs.properties中配置websocket.config，例如图4.62，就可以实现websocket服务端的功能。



图4.62

实现WebSocketClientInterface接口,实例化WebSocketClient，例如图4.64，就可以发送websocket请求。

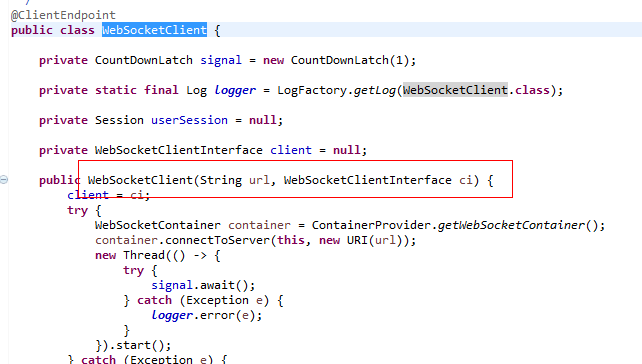


图4.64

4.24集成Ueditor在线编辑器

Demo展示如图4.65。具体使用参照Ueditor的官网说明。

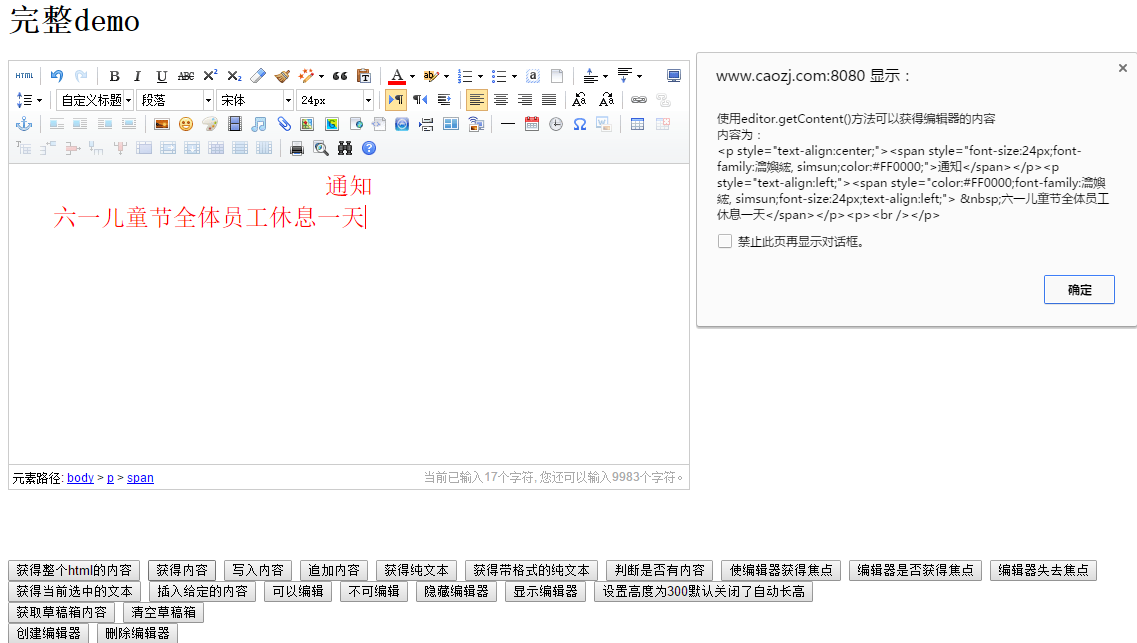


图4.65

4.25 快递查询

使用快递100接口实现，使用时需要在快递100官网申请，将分配的快递接口id填入配置文件configs.properties文件（如图4.66）中。

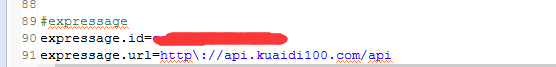


图4.66

调用工具类ExpressageUtil，传递参数就可以查询出快递信息，例如图4.67。

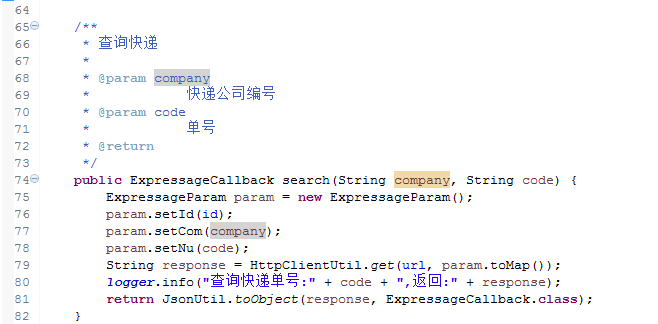


图4.67

4.26 FTP工具类

FtpServerUtil使用apache的ftp server实现，user.properties的配置方式（如图4.68）可以查询apache ftp server。

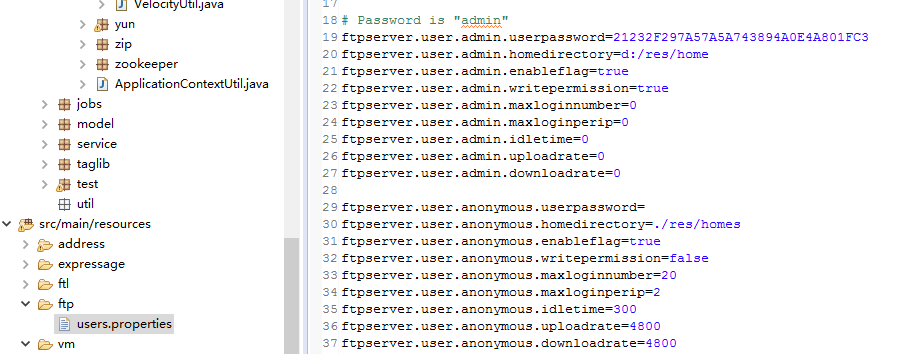


图4.68

4.27 Groovy

Groovy是一种动态脚本语言，底层使用Java实现，所以95%的java代码可以直接在Groovy中运行，GroovyUtil就是一个在Java中调用Groovy的工具类

Groovy可以在代码中直接调用Java。

Groovy可以满足系统中部分业务代码需要在服务运行期间新增修改，而不用重启服务的要求。可以直接在系统后台执行某些代码，完成运行时的操作，某些代码在开发阶段无法确定，需要在运行时才能确定。Groovy代码可以保存在数据库中，也可以保存在配置文件中。使用情况如图4.69、图4.70和图4.71。

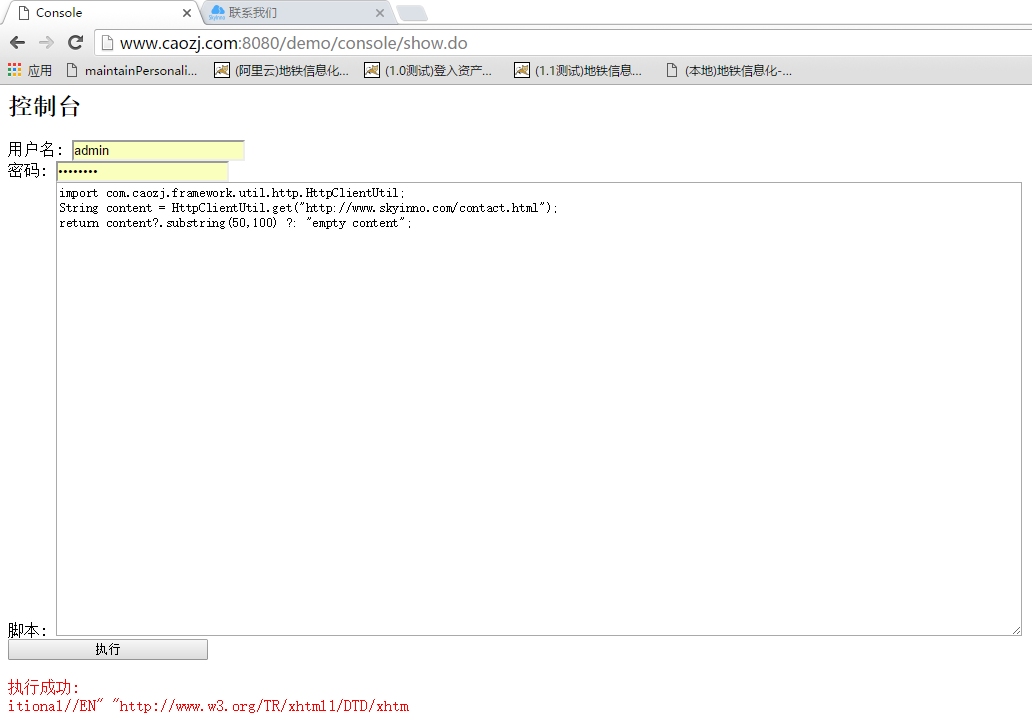


图4.69

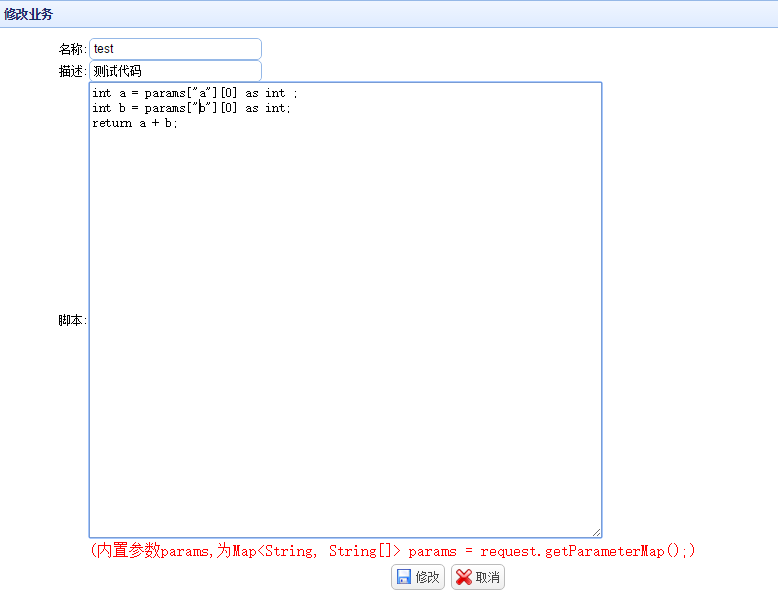


图4.70

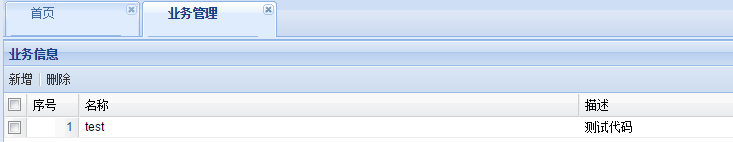


图4.71

将Groovy脚本保存在数据库中，提供接口让外部调用，返回执行结果，

可以实现代码的新增修改，不需要重启服务器。

4.28 Spring Cache

框架已经集成了Spring Cache，实现了默认的Redis缓存。直接在方法上面使用Spring Cache的注解即可，例如图4.72。

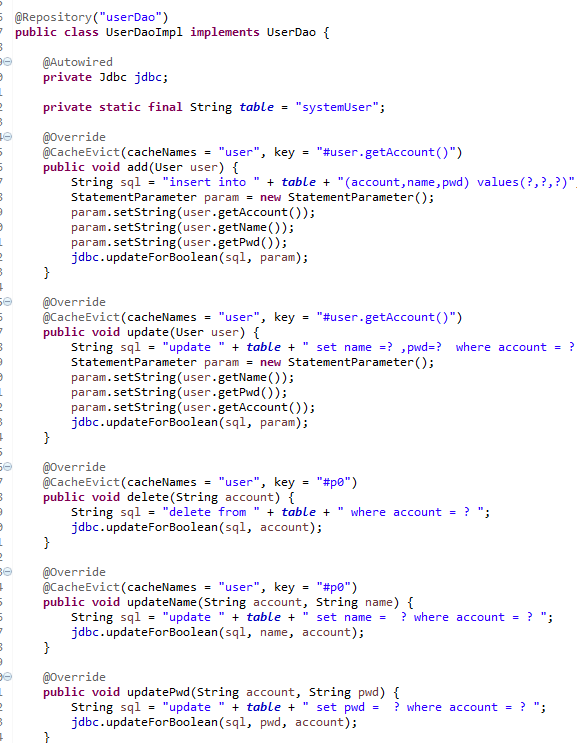


图4.72

4.29 地址联动

框架将地址联动的功能集成进去。如图4.73。

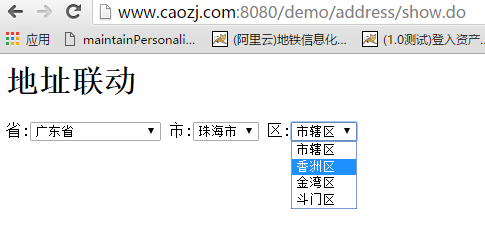


图4.73

4.30 截图

框架将截图功能集成进来。

视频截图：可以使用VideoUtil对视频进行截图。

图片截图：可以使用thumbnailator对图片进行缩略图、截取、旋转等操作，具体使用接口可网上查询。

4.31 Thirft

Thirft主要用于跨语言通信，支持多种开发语言，由fackbook开发，后转交给Apache。Thirft服务端和客户端都已经封装好，具体使用可以查看代码注释。客户端实现了3种类型：同步阻塞、同步非阻塞、异步非阻塞，服务端只实现了非阻塞方式。

4.32 集群执行功能

集群执行功能利用Redis的发布订阅功能完成，所以需要配置configs.properties文件，配置方式如图4.76。

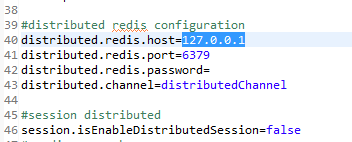


图4.76

实现接口ClusterExecute，定义集群传递的参数对象，必须实现序列化接口distributedUtil.executeInCluster执行这个方法之后，所有的服务器都会调用ClusterExecute实现类的execute方法，参数为上图4.77的clustData。



图4.77

4.33 Spring常用注解

在本框架中常用到的注解有：@Controller，@RequestMapping，@Resource，@Autowired，@Component，@Repository，@Service，@Transcational。

@Controller：控制器Controller 负责处理由DispatcherServlet 分发的请求，它把用户请求的数据经过业务处理层处理之后封装成一个Model ，然后再把该Model 返回给对应的View 进行展示，标注方式如图4.78.

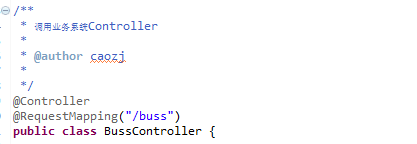


图4.78

@RequestMapping：一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路。标注方式如图4.79

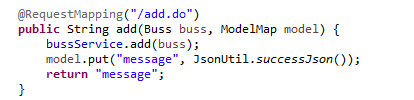


图4.79

@Resource：按照ByName自动注入。

@Autowired：按照类型（byType）装配依赖对象，默认情况下它要求依赖对象必须存在，如果允许null值，可以设置它的required属性为false。

@Component：通用的注解，一个普通的Spring Bean。

@Repository：用于持久层Dao上的注解，也是Spring Bean。

@Service：用于持久层Service上的注解，也是Spring Bean。

@Transcational：一般用于Service层，代表使用数据库事务管理，可用于类和方法上。

4.34分布式session

分布式Session主要是利用第三方组件来保存公共的Session中的数据。提供两种方式1.Redis；2.Zookeeper。推荐使用Redis，使用方式如图4.80



图4.80

4.35 自定义注解

可以参照图4.81、图4.82的例子编写自己的功能注解。



图4.81



图4.82

4.36 前端使用说明

前端显示系统结构如图4.83和图4.84。

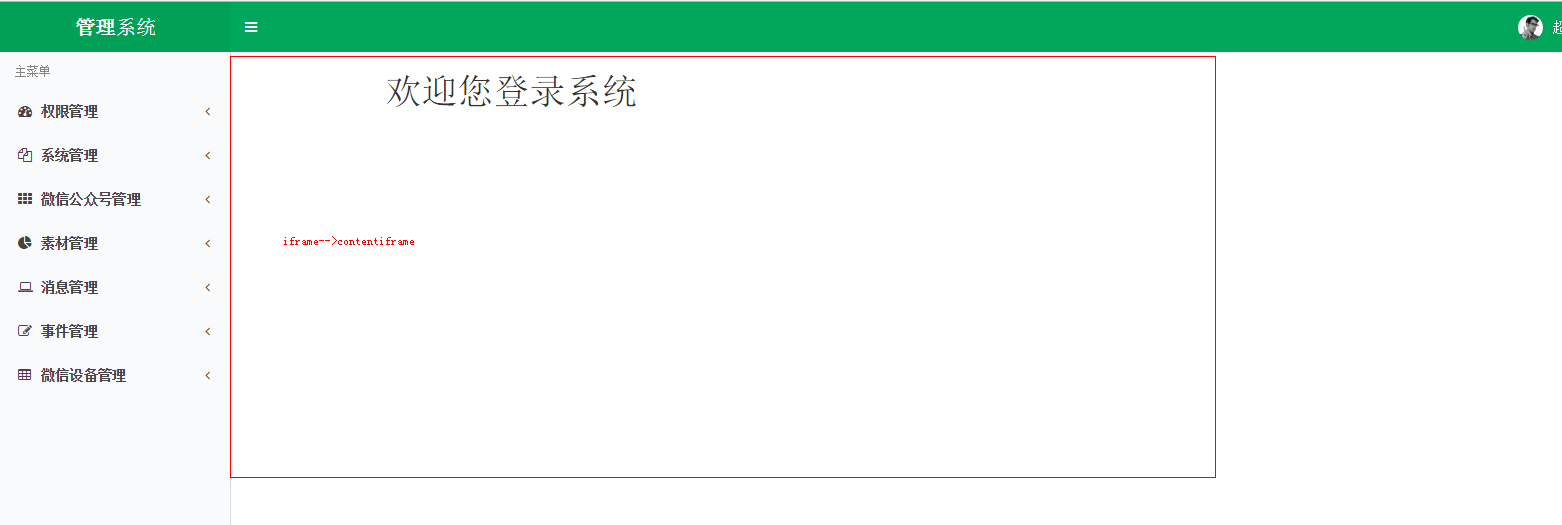


图4.83

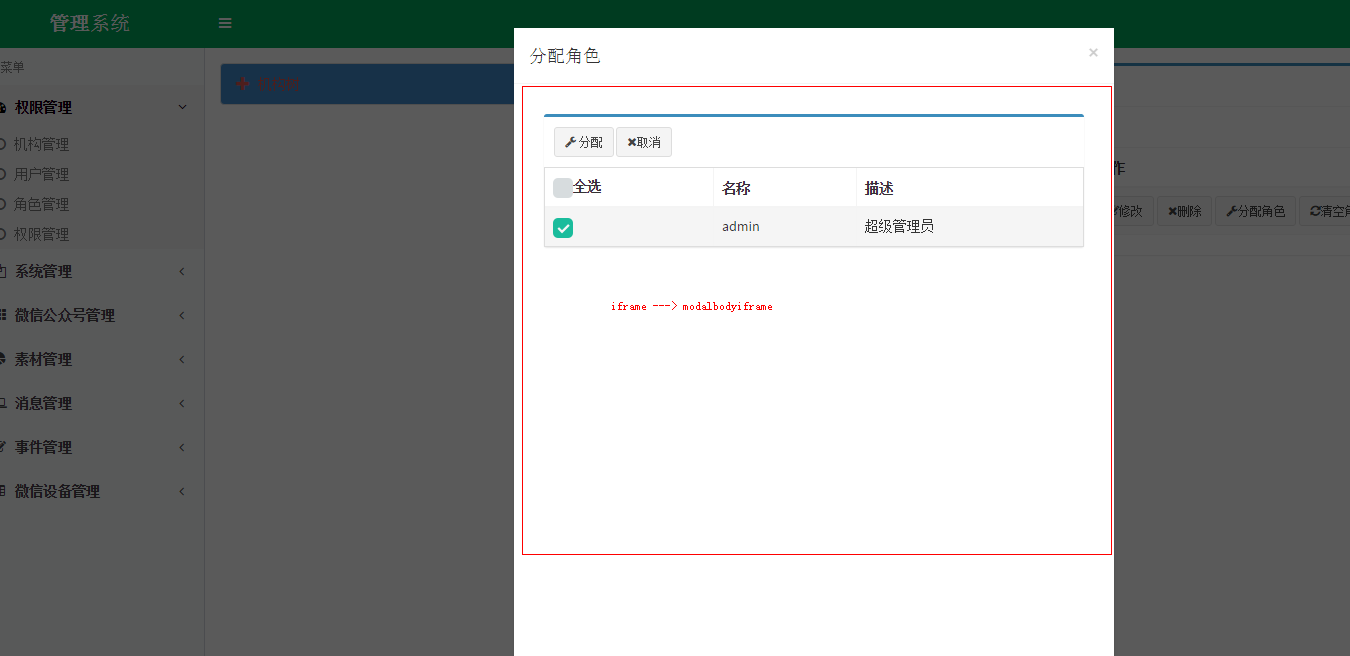


图4.84

在ifame内，可以使用parent.或者top.调用如图4.85和4.86的方法



图4.85

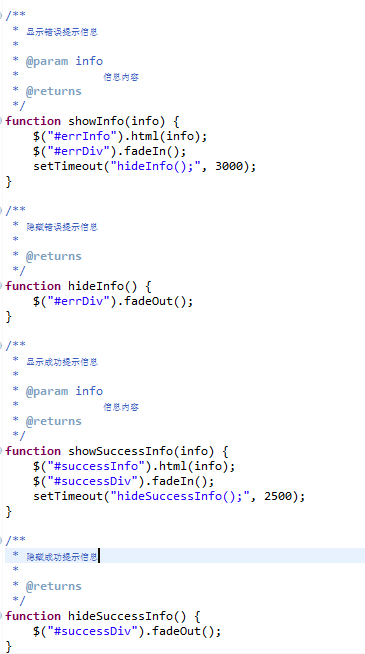


图4.86

显示的模态窗口效果、错误提示信息和成功提示信息分别如图4.87、图4.88和图4.89。

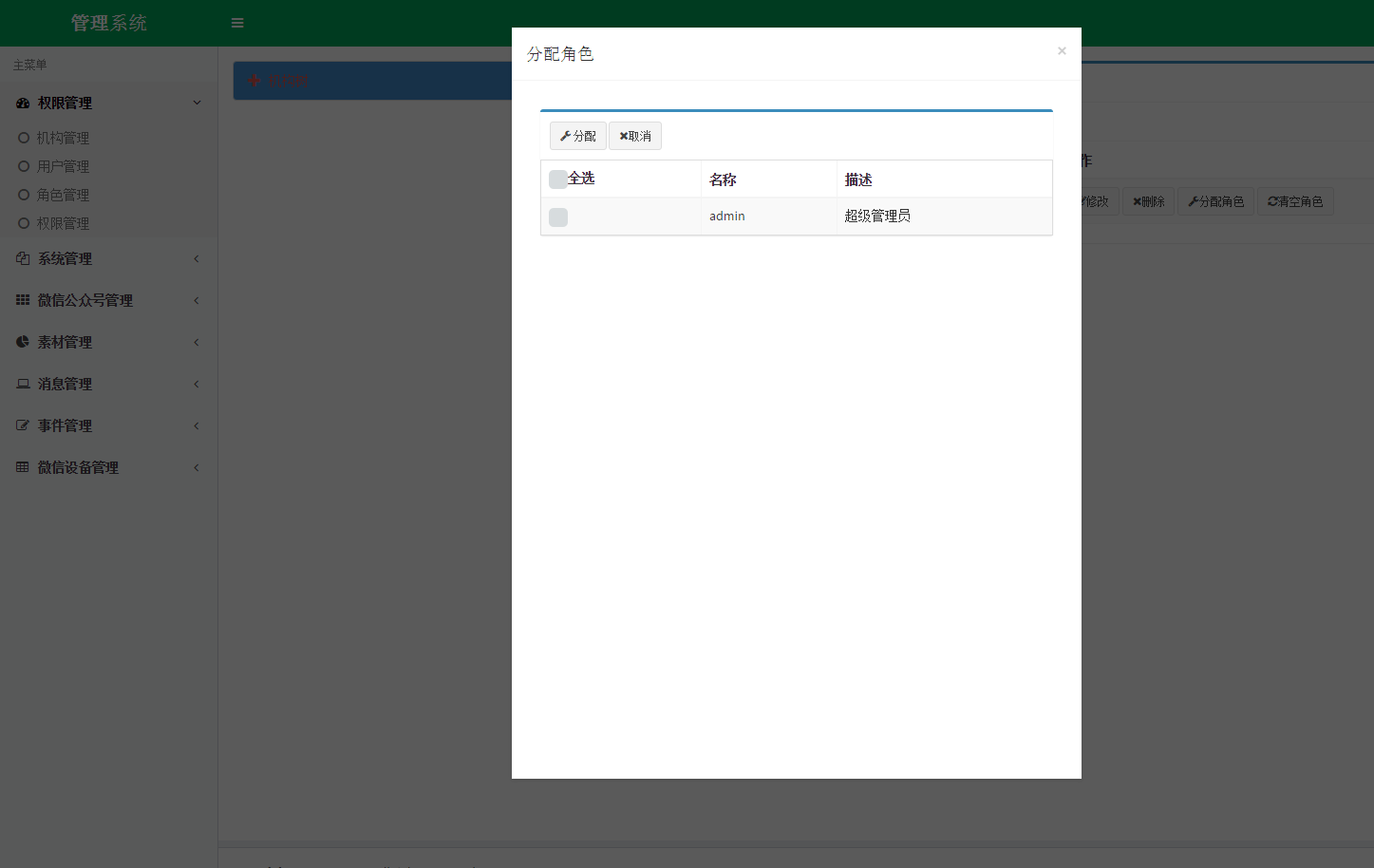


图4.87

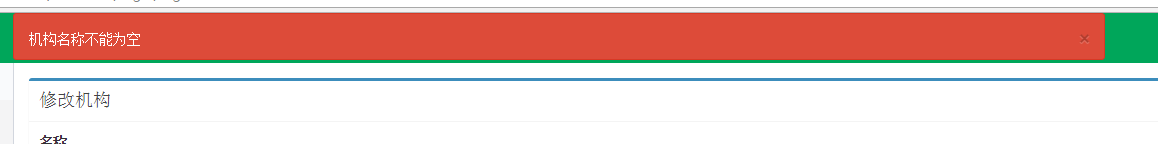


图4.88

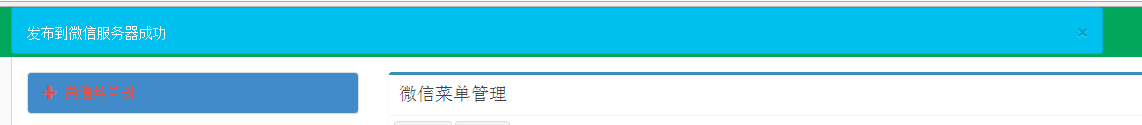


图4.89