目录

[1.安装 2](#_Toc418167907)

[1.1导入数据 2](#_Toc418167908)

[1.2 修改配置文件 2](#_Toc418167909)

[1.3打包部署 2](#_Toc418167910)

[2.公众号接入步骤 2](#_Toc418167911)

[3.消息处理 3](#_Toc418167912)

[3.1消息处理流程 3](#_Toc418167913)

[3.2文字消息 4](#_Toc418167914)

[3.2.1文字消息保存 4](#_Toc418167915)

[3.2.2文字消息处理 4](#_Toc418167916)

[3.3图片消息 4](#_Toc418167917)

[3.4地理消息 4](#_Toc418167918)

[3.5声音消息 4](#_Toc418167919)

[3.6关注事件 4](#_Toc418167920)

[4.菜单处理 5](#_Toc418167921)

[4.1菜单保存 5](#_Toc418167922)

[4.2菜单点击 5](#_Toc418167923)

[5.升级注意事项 5](#_Toc418167924)

技术手册

# 1.安装

## 1.1导入数据

db/mysql/db-mysql.sql，默认数据库名是xd-wechat，可在db.properties中修改

## 1.2 修改配置文件

Config/db.properties修改数据库配置

Config/spring-email.xml修改邮箱账号和密码，同时在类com.fjx.common.mail.vo中修改fromUser信息，使二者保持一致。

## 1.3打包部署

部署Tomcat或Jetty，打开浏览器，访问<http://yourhost:port/ttwx>

# 2.公众号接入步骤

在可以访问上述页面之后，进入后台管理页面（此时可以先注册一个用户；或使用测试账号，用户名：fengjx；密码：1989），在“配置授权”一栏，可以完成手机微信对接工作。配置授权代码入口：com.fjx.wechat.base.admin.action.SettingController

1、将授权信息（URL、TOKEN）配置到公众平台

2、从微信客户端向公众号发送验证码（系统生成，如：52731 ）

3、完成授权

注意：

1.请不要点击“更新授权”按钮，否则URL、TOKEN、验证码将会重新生成，此时需要重复公众号接入步骤。“更新授权”对应于com.fjx.wechat.base.admin.action.SettingController中@RequestMapping(value=”/save”, method=RequestMethod.POST)方法。

2.此版本代码不支持企业号的认证。即在企业号后台页面进行服务器资源配置时，无法通过微信服务器的认证。可能是因为企业号跟服务号的认证时所接受的参数有所不同，亦或是认证算法有所不同。

# 3.消息处理

com.fjx.wechat.base.admin.action.WechatApiAction是微信请求的入口，所有微信发起的请求会被这个类拦截。这个类提供了两个方法，分别对应Get和Post请求。Get请求用于微信认证，当跟微信服务器对接时，微信服务器会发起一个Get请求到公司服务器，以验证消息是来自于微信服务器。Post请求则是对各种消息的处理。消息类型在包com.fjx.wechat.mysdk.beans下，包括请求的消息类型、返回的消息类型等。消息处理过程在包com.fjx.wechat.mysdk.process.in.executor下。

## 3.1消息处理流程

发起微信请求

WechatInterceptor（解析消息，将XML对象转换成Map对象，放入WechatContext上下文中，使用ThreadLocal保存）

WechatApiAction（消息请求入口）

InServiceEngineImpl（消息请求处理中心）

Dispatcher（消息转发，根据消息类型，构建相应的消息处理器）

InWechatEventClickMsgExecutor（菜单点击事件）

InWechatImageMsgExecutor（图片消息事件）

InWechatLocationMsgExecutor（地理位置事件）

……（其他事件）

## 3.2文字消息

### 3.2.1文字消息保存

“自动回复”一栏功能：当用户向服务器发起微信请求时，此时消息的类型为Text，服务器会根据这一关键字返回预定义的消息。

1.文字消息保存入口：com.fjx.wechat.base.admin.action.RespMsgActionController下的save方法。

2.若关键字所对应的回复内容为文字类型，则先保存到素材表（wechat\_material），再保存到动作表（wechat\_resp\_msg\_action）。若为图文信息，则直接保存到动作表，因为图文素材的保存在“素材管理”一栏来保证，所以在“素材管理”一栏只显示图文素材，即type=news

### 3.2.2文字消息处理

当微信手机客服端向公众号发起文字请求时步骤：

1.流程详见“消息处理流程”，最终进入com.fjx.wechat.mysdk.process.in.executor. InWechatTextMsgExecutor下的execute方法。

2.从动作表（wechat\_resp\_msg\_action）中查找对应的actionEntity。

3.若存在actionEntity，再从素材表中读取对应回复内容返回给用户。

4.若不存在，则进入com.fjx.wechat.extension.service.MyTextExtService，根据请求关键字，返回对应的消息。

## 3.3图片消息

暂无处理逻辑

## 3.4地理消息

后台管理页面的“LBS回复”对应的是对地理消息处理，可以在此进行设置。

## 3.5声音消息

暂无处理逻辑

## 3.6关注事件

当用户通过扫公众号的二维码时，微信服务器会将此消息转发到公司服务器上，此时就可以定制我们的欢迎词或其他对用户的操作。根据消息转发逻辑，关注事件最终进入com.fjx.wechat.mysdk.process.in.executor.InWechatEventSubscribeMsgExecutor类被处理。而设置此消息的页面在“粉丝关注回复”一栏。

# 4.菜单处理

## 4.1菜单保存

菜单增加、删除、发布等操作入口：com.fjx.wechat.base.admin.action.WechatMenuController

菜单所对应的动作，如点击事件、跳转事件，入口：com.fjx.wechat.base.admin.action.RespMsgActionController中save方法。

为了增加对于复杂菜单的处理，即点击菜单时，不仅仅显示文字或图文，而是需要增加自己的逻辑，增加了“不设置动作”一栏，若菜单以此类型来保存的话，需要实现自己的逻辑。

若菜单所对应的回复内容为文字类型，则先保存到素材表（wechat\_material），再保存到动作表（wechat\_resp\_msg\_action）。若为图文信息，则直接保存到动作表，因为图文素材的保存在“素材管理”一栏来保证，所以在“素材管理”一栏只显示图文素材，即type=news

## 4.2菜单点击

根据消息转发逻辑，菜单点击动作对应com.fjx.wechat.mysdk.process.in.executor.InWechatEventClickMsgExecutor类。处理逻辑：

1.从动作表（wechat\_resp\_msg\_action）获取相应的actionEntity

2.若actionEntity不存在，com.fjx.wechat.extension.service.MenuDispatcherService将转发菜单，进入相应的处理逻辑

3.若2）有返回数据，将数据直接返回到微信客户端

4.若2）无返回数据，在动作表（wechat\_resp\_msg\_action）中获取默认的actionEntity

5.从1）或4）中获取到actionEntity后，根据action\_type判断是从素材库还是第三方应用中读取数据，后返回数据

# 5.升级注意事项

升级时需要修改的类

1.com.fjx.wechat.mysdk.process.in.executor.InWechatEventClickMsgExecutor：增加对复杂菜单的处理逻辑。

2./WEB-INF/view/wechat/admin/menu/action\_inc.jsp：增加“不设置动作”一栏

3./static/script/admin/menu.js：在提交操作中增加“setType”，来标识此次菜单点击事件是否需要自己的处理逻辑

4.com.fjx.wechat.base.admin.service.RespMsgActionServiceImpl：修改saveMenuAction（），当菜单类型为click，且菜单设置了动作规则时，才需要保存actionEntity、materialEntity所对应的表

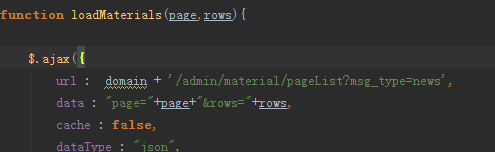
5./config/spring-mvc-interceptor.xml：微信二级页面需要跳转到其他二级页面时，如何绕开AdminInterceptor拦截器，即不会因为超时而跳到登录页面。可以增加<mvc:exclude-mapping path="…"/>或设置请求路径，只要请求路径不被<mvc:mapping path="…"/>拦截。由于<mvc:exclude-mapping path="…"/>不被spring-mvc-3.0.xsd所识别，需改为spring-mvc-3.2.xsd。

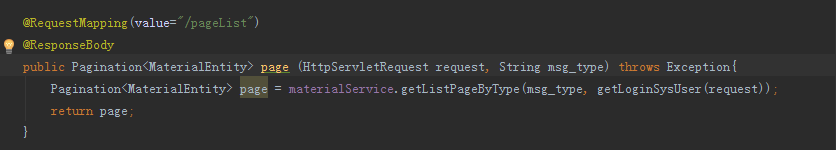
6./config/admin-menu.json：主页所显示的导航栏，修改导航栏时，只需要增加或删除相应的JSON数据即可。

7.素材管理（与分页显示相关问题）：

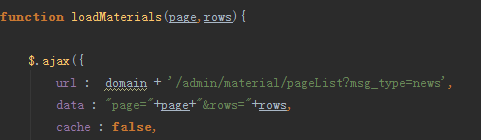
素材分页显示的逻辑是没有问题的，问题在material.js中loadMaterials方法向URL提交请求时，未被识别为是分页请求，所以导致无法分页显示。如何被识别为是分页请求？在web.xml配置文件里设置了拦截分页请求的规则：page\_url。需要修改相应的地方有：

Material.js:



com.fjx.wechat.base.admin.action.MaterialController：修改映射路径

当然在action\_inc.js中也用到了loadMaterials方法，因此也需要修改。



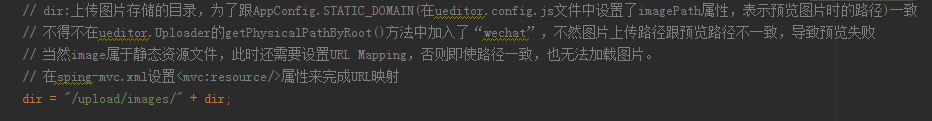
8.素材管理（图片上传和预览）

图片上传的功能使用了第三方组件ueditor，需要修改里面相应的jsp和java文件。

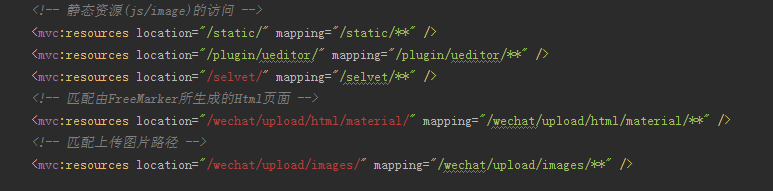
/plugin/ueditor/jsp/imageUp.jsp：这个文件负责图片的上传，原先使用了绝对路径做测试使用，现在修改为上传到服务器目录。其实只需要修改

这一语句即可。

但是若只是修改此处地方的话，会导致图片信息无法预览，为了保持图片预览路径和上传路径一致，还需要修改ueditor.Uploader的getPhysicalPathByRoot()方法中加入了“wechat”，因此实际上传的路径是/wechat/${dir}。

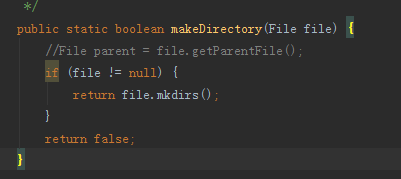


同时修改映射信息如下：

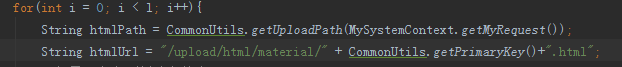


9.素材管理（由FreeMarker生成的HTML）

由FreeMarker生成的HTML页面也是保存在上传目录文件夹html下。原先使用这个功能时提示无法找到系统路径，是因为com.fjx.common.utils.FileUtil中makeDirectory方法只是创建了父目录而已，修改如下：



同时还需修改com.fjx.wechat.base.admin.service.MaterialServiceImpl



在com.fjx.common.utils.CommonUtils类中增加了获取上传根目录getUploadPath的方法。

