



微信轻应用的先驱者，丰富的项目案例，
由浅至深，一步步带您掌握企业号开发。



实战：八个实际项目案例，从基础入门到高级应用，手把手教你成为微信大牛

简单：30天，上千行核心代码让你精通微信企业号开发

丰富：AngularJS、ECharts、Qpid、WebSocket、Servlet等10余种技术与微信的结合

创新：类似微信小程序的单页面应用开发讲解



微信企业号开发 完全自学手册

牟云飞
编著



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
BYEWEBSITE: <http://www.phei.com.cn>

微信企业号开发 完全自学手册

牟云飞
编著

電子工業出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

本书是微信公众平台企业号开发较全面、系统的一本书，以实战开发为原则，讲解微信各个模块的开发使用，以实例引导企业号的开发与运用，以 Struts、Hibernate、Servlet、HttpClient、JSP、Ajax、jQuery 等热门技术实现微信 Light App 的开发，通过 QPID、代理服务、页面有效期等方式实现数据的安全交互。除此之外，对 SPA 单页面应用框架如何在微信中运用也做了详细介绍。

本书共 11 章，涵盖的主要内容有：微信公众号概述、企业号的发展与注册、配置微信开发环境、JCE 安全策略、微信企业号开发基础知识、主动推送模式、被动回调模式、企业会话模式、JSAPI 模式、通讯录管理、语音导航、腾讯地图使用、WebSocket 微信开发、微信单页面应用、QPID、前置机数据安全访问、企业资讯、微信考勤等。

本书由简入深，实用性较强，即便没有微信开发经验的读者，也能够一步步学习微信开发，学会每个接口的调用及问题处理。有公众号开发经验的读者，则可以重点阅读 JSAPI 和数据安全章节，丰富企业号应用，解决微信 SPA 物理回退、语音导航等问题。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

微信企业号开发完全自学手册 / 牟云飞编著. —北京：电子工业出版社，2017.3
ISBN 978-7-121-30809-3

I. ①微… II. ①牟… III. ①移动终端—应用程序—程序设计—手册 IV. ①TN929.53-62

中国版本图书馆 CIP 数据核字（2017）第 011055 号

策划编辑：张月萍

责任编辑：安 娜

印 刷：北京京科印刷有限公司

装 订：三河市良远印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱

邮编：100036

开 本：787×1092 1/16

印张：29

字数：736 千字

版 次：2017 年 3 月第 1 版

印 次：2017 年 3 月第 1 次印刷

定 价：76.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

推荐序

随着移动互联网的迅猛推广，尤其是微信等移动社交平台的快速普及，企业运营协作模式也在发生深刻变化，企业信息化走向移动已经成为十分紧迫的课题。诞生于桌面 PC 时代的企业信息化目前还主要应用于桌面环境，移动化可以将信息接入从桌面向智能终端大大延伸，从而突破固有的终端种类、接入时间和地点的局限性，可以实现企业信息化真正的无缝闭环，这无疑是企业信息化发展历程中的一次质的飞跃。

企业移动信息化的实现途径多种多样，从最初的 WAP 网站方式到后来的智能 App 模式，再到轻应用模式，技术实现方式背后蕴藏着各种企业信息化要素的平衡和优化，这些要素包括用户体验、开发成本、企业信息安全、推广效率，等等。微信企业号正是可以满足这些要素的优秀解决方案，因此甫一推出就立刻受到了广泛的关注和认可。

在用户体验方面，由于企业信息化应用绝大多数涉及的只是信息的浏览和表单的处理，对用户体验的方面要求并不高，因此微信平台提供的轻应用完全可以胜任。从开发成本上考虑，由于微信企业号的开发采取的是跨平台的网页开发技术，而开发跨平台应用，相较于原生 App 开发无疑可以节省大量的开发和测试成本，对于项目来说，也就意味着可以在较短的时间内实现应用上线，从而迅速产生实际效益。

仅有项目开发的“多快好省”还不行，在数据成为企业新的重要资产的今天，互联网环境下的信息安全成为企业在部署移动化之前不得不考虑的前提。传统的移动信息安全一般要借助于移动设备管理 MDM 等系统级平台，项目投资大，对于移动设备的侵入性也非常大，对于中小型企业来说，往往难以承受。针对信息安全，微信企业号提供了相对轻量级的安全架构，将内部相对封闭的内部组织架构与个人微信号实现绑定，通过企业号后台可以对进入企业号的微信号进行认证、绑定以及后续的注销，具备基本的用户管理能力，而进一步更严格的认证措施则可以通过企业号的二次开发来实现。

最后再来看一下微信企业号的推广效率。由于微信本身已经成为覆盖绝大多数企业员工的社交平台，企业往往也已经建立了各种微信群或微信公众号，借助于这些传播渠道，微信企业号可以很轻易地获得推广，而且绑定动作相对于 App 的安装来说轻量且优雅，也不存在 App 后续的升级更新问题。

综合以上特征，个人认为微信企业号对于信息敏感性不太高的中小企业来说无疑是最适合的企业移动信息化扩展平台。通过在微信企业号上进行一定的二次开发，就可以轻易地使企业内部的信息化系统具备移动化能力。

我们海颐软件正是这样一家中等规模的软件企业，并较早成为了微信企业号的用户。本书的作者正是我们微信企业号的主要开发者。在实际开发过程中，他积累了丰富的知识和经验。相信借助于本书，您可以绕开大部分的困扰和陷阱，帮助您直达目标，迅速构建出令人满意的微信企业号应用来。

李锐

烟台海颐软件股份有限公司 副总经理

为什么要写这本书

智能手机的日渐普及不断地推动着移动互联网在各行业的应用，众多的 App 琳琅满目，App 开发也从最初的 Native App 开发，发展到 Native App、Web App 等多种开发技术。开发越来越容易，各类客户需求的分散，导致 App 越来越多，大量功能单一的应用被搁置，成为“僵尸应用”。越来越多的用户将视线聚集到微信、QQ、新浪微博等超级应用中，在超级应用倍受关注的形势下，Light App 应运而生。Light App 又被称为轻应用、微应用，是一种无须下载、即搜即用的全功能 App，既有媲美甚至超越 Native App 的用户体验，提升用户群体，又具备 Web App 快速开发节约开发成本等特性，前景更加广阔。

微信公众号是腾讯公司在微信的基础上推出的，属于 Light App 的范畴，使广大微信用户无须下载便能够借助微信直接享受个人或企业提供的各类服务。对于企业推广和发展来说，企业可以通过订阅号、服务号打造一个基于微信的服务或推广平台；而对企业内部，企业能够通过微信新推出的企业号实现对内部管理系统的集成，包括：人力资源管理系统、报销管理系统、企业论坛、新闻通知公告、即时通信系统、OA 协同办公等系统，使各类系统移动化，提高工作效率。

企业号能够高效地帮助政府、企业及组织构建自己独有的生态系统，随时随地连接员工、上下游合作伙伴及内部系统和应用，实现业务及管理的互联网移动化。2014 年 9 月企业号进行公测，2015 年笔者因工作需要开始进行微信企业号开发，当时市面上基本没有企业号开发的相关文章，笔者先后完成多个企业号项目开发，编写数个微信企业号建设方案，并在 CSDN 发布了几篇博文，收到许多读者和企业的来信。随着企业号近两年的发展，越来越多的企业想开发企业号，企业号开发人员也受到软件公司的青睐。

为了把微信公众平台开发经验以及企业号的运用更详细、系统地分享出来，笔者在源智图书李幸编辑的鼓励下编写了这本书，希望认识更多的 IT 人才。

本书内容及知识体系

第 1 篇 微信企业号概述及开发基础知识（第 1~2 章）

本篇介绍了微信公众平台企业号概述以及微信企业号开发环境的配置和开发基础知识，

主要包括微信公众号的区别、企业号的发展与注册、配置微信开发环境、JCE 安全策略的调整、微信调试工具的安装与使用、HttpClient 服务请求调用、域名发布使用以及 Properties 文件配置等。

第 2 篇 典型模块开发（第 3~7 章）

本篇介绍了微信开发各种模式下接口调用及开发实现，主要包括 AccessToken 申请、Token 缓存处理、各类消息的主动推送、素材管理、开启回调模式、消息的接收与响应、ECharts 运用、语音导航实现、WebSocket 连接实现、SPA 开发、企业 IM 与微信的对接、通讯录异步任务维护以及现场业务上报实现等。

第 3 篇 微信数据安全及数据库基础（第 8~9 章）

本篇主要介绍了微信公众号数据安全访问的方式策略，主要从软件开发角度实现数据的传输，通过识别浏览、OAuth 2.0 身份验证、页面访问有效期、QPID 以及前置机代理服务等方式实现数据的传输。

第 4 篇 项目案例实战（第 10~11 章）

本篇主要以案例的方式介绍了微信企业号应用的开发过程，从应用创建到应用开发实现，一步步带领读者学习企业号开发，学习企业资讯、微信考勤等应用的实现。

适合阅读本书的读者

- 需要全面学习微信企业号开发技术的人员；
- 微信公众号开发技术人员；
- 微信单页面应用开发人员
- Java 程序员；
- Java EE 开发工程师；
- 希望提高微信项目开发水平的人员；
- 专业培训机构的学员；
- 微信企业号应用开发项目经理；
- 需要一本微信功能查询与实现手册的人员。

阅读本书的建议

- 没有微信开发经验的读者，建议从第 1 章顺次阅读并演练每一个实例。
- 有一定微信开发基础的读者，可以根据实际情况有选择地阅读各个模块和项目案例。
- 对于有微信公众号项目经验或者对单页面应用开发有兴趣的读者，可以重点阅读第 5 章 JS-SDK 的相关开发。
- 阅读时建议首先阅读一遍书中的模块和项目案例，然后从 Hello World 写起，“千里之行，始于足下”。大到每个案例，小到每行代码，哪怕简单的变量定义也在 SDK 中书写一遍，这样不仅能够提高代码速度、效率，而且理解起来也会更加深刻、容易。

致谢

感谢微信创始人张小龙先生及其团队创造了微信这一优秀的平台；
感谢海颐软件李锐、宋庆伟、于洋提供的微信公众账号开发机遇；
感谢徐国智、李明、马金刚在技术上的启蒙与指导；
感谢于春洋在我写书期间在生活上给予的鼓励与帮助；
感谢身边的同事以及广大 IT 网友对这本书的支持与鼓励。

目 录

第一篇 从零开始学企业号

第 1 章 微信公众平台——认识企业号	2
1.1 微信企业号简介	2
1.1.1 平台发展历程	2
1.1.2 企业号定位	3
1.1.3 与订阅号、服务号区别	3
1.1.4 企业号应用	4
1.2 企业号注册	5
1.2.1 基本信息	5
1.2.2 邮箱激活	5
1.2.3 选择类型	6
1.2.4 信息登记	7
1.2.5 公众号信息	10
1.2.6 绑定管理员	11
1.2.7 增加管理员	11
1.2.8 认证	13
1.3 应用创建	14
1.3.1 进入应用中心	14
1.3.2 选择应用类型	15
1.3.3 填写应用信息	15
1.3.4 完成应用创建	16
第 2 章 平台开发基础入门	17
2.1 JDK 及 JCE 补丁部署	17
2.1.1 安装 JDK	17
2.1.2 环境变量	19
2.1.3 JCE 安全策略补丁	21
2.2 开发环境	22
2.2.1 安装 MyEclipse	22
2.2.2 绑定服务器	24
2.2.3 调整编译环境	26
2.2.4 微信 web 开发工具	27

2.3	HttpClient 使用技巧.....	29
2.4	URLConnection 使用技巧.....	32
2.5	Properties 配置文件使用.....	36
2.6	接口调试工具.....	37
2.7	发布外网服务.....	38
2.8	公众平台消息模式.....	39
2.9	微信企业号入门 Hello World.....	40

第二篇 微信企业号开发核心技术

第 3 章	主动调用模式	46
3.1	主动调用模式介绍	46
3.2	申请 AccessToken	47
3.3	AccessToken 的缓存处理.....	50
3.4	主动调用频率限制	53
3.5	信息推送	53
3.5.1	接口说明.....	54
3.5.2	推送文本消息.....	56
3.5.3	推送图片消息.....	61
3.5.4	推送语音消息.....	62
3.5.5	推送视频消息.....	66
3.5.6	推送文件消息.....	70
3.5.7	推送新闻消息.....	73
3.5.8	推送永久图文消息.....	79
3.5.9	管理端推送消息.....	86
3.6	素材管理	87
3.6.1	接口说明.....	87
3.6.2	上传临时素材文件.....	87
3.6.3	获取临时素材文件.....	90
3.6.4	上传永久素材（非图文素材）	92
3.6.5	上传永久素材（图文素材）	93
3.6.6	获取永久素材（非图文素材）	97
3.6.7	获取永久素材（图文素材）	98
3.6.8	删除永久素材.....	99
3.6.9	修改永久图文素材.....	100
3.6.10	获取素材总数.....	101
3.6.11	获取素材列表.....	102
3.6.12	管理端素材维护	104
3.7	企业号应用管理.....	105
3.7.1	获取企业号应用.....	105
3.7.2	设置企业号应用.....	107

3.7.3	获取应用概况列表	108
3.7.4	管理端应用管理	109
3.8	主动模式自定义菜单	110
3.9	信息自动回复	111
3.10	案例：业务派单	113
第 4 章	被动回调模式	117
4.1	被动回调模式介绍	117
4.2	开启回调模式	119
4.3	加密/解密算法	123
4.4	被动模式自定义菜单	125
4.4.1	限制与说明	125
4.4.2	创建菜单	127
4.4.3	删除菜单	132
4.4.4	获取菜单列表	133
4.4.5	管理端菜单维护	134
4.5	接收消息 Dom 解析	135
4.6	消息响应 Xstream 转换	138
4.7	接收普通消息	141
4.7.1	接口说明	141
4.7.2	接收文本消息	145
4.7.3	接收图片消息	146
4.7.4	接收音频消息	147
4.7.5	接收位置消息	148
4.7.6	接收小视频消息	149
4.7.7	接收链接消息	151
4.7.8	接收视频消息	152
4.8	接收事件消息	153
4.8.1	接口说明	153
4.8.2	接收关注/取消关注事件	155
4.8.3	接收地理位置事件	157
4.8.4	接收进入应用事件	158
4.8.5	接收菜单事件	159
4.8.6	接收异步任务完成事件	166
4.9	被动响应消息	167
4.9.1	接口说明	167
4.9.2	被动响应文字消息	169
4.9.3	被动响应图片消息	171
4.9.4	被动响应音频消息	173
4.9.5	被动响应视频消息	175
4.9.6	被动响应图文消息	177
4.10	案例：企业通讯录快速搜索	180

第 5 章 JSAPI 模式	192
5.1 JSAPI 模式介绍	192
5.2 页面接口引入	193
5.2.1 配置“可信域名”	193
5.2.2 引入微信 JS 文件	194
5.2.3 权限验证	194
5.2.4 验证成功事件	199
5.2.5 验证失败事件	199
5.3 Debug 调试及基础接口说明	199
5.3.1 Debug 调试模式开启	199
5.3.2 判断当前客户端版本是否支持指定 JS 接口	200
5.3.3 接口通用函数	201
5.4 微信 JS-SDK 接口说明	201
5.5 权限接口应用	202
5.5.1 隐藏右上角菜单	202
5.5.2 GPS 定位获取位置信息	204
5.5.3 图片处理接口	205
5.5.4 语音及智能接口	206
5.6 ECharts 在微信中的应用	208
5.6.1 ECharts 简介	208
5.6.2 ECharts 快速接入	208
5.6.3 ECharts 微信应用	210
5.7 微信中的地图语音导航	214
5.7.1 微信内置地图导航	214
5.7.2 腾讯地图语音导航	215
5.7.3 百度地图语音导航	217
5.8 微信 SPA 开发	219
5.8.1 基于 AngularJS 的 onsenUI	219
5.8.2 创建 AngularJS 微信服务	220
5.8.3 SPA 下 JSAPI 模式权限初始化	221
5.8.4 SPA 下获取 OAuth 2.0 成员身份信息	222
5.8.5 解决微信物理回退	223
5.9 微信 WebSocket 开发	224
5.9.1 WebSocket 客户端	224
5.9.2 WebSocket 服务端	226
5.10 微信中的支付宝	228
5.11 常见问题	229
5.12 案例：现场业务上报	232
5.12.1 场景回顾	232
5.12.2 示例代码展示	232

第 6 章	企业会话模式	240
6.1	企业会话模式介绍	240
6.2	开启企业会话	242
6.3	推送聊天信息	245
6.3.1	信息推送接口说明	245
6.3.2	聊天消息体结构说明	247
6.3.3	创建多聊会话	250
6.3.4	修改多聊会话	253
6.3.5	退出多聊会话	255
6.3.6	获取多聊会话信息	256
6.3.7	清除未读会话状态	257
6.3.8	会话消息免打扰	258
6.4	接收聊天信息	260
6.4.1	信息接收接口说明	260
6.4.2	普通消息结构体说明	262
6.4.3	事件消息结构体说明	265
6.5	案例：企业 IM 与微信的对接	267
第 7 章	通讯录管理及异步任务	275
7.1	成员验证关注	275
7.2	部门管理	276
7.2.1	新增部门	276
7.2.2	更新部门	277
7.2.3	删除部门	278
7.2.4	获取部门列表	278
7.3	成员管理	279
7.3.1	新增成员	280
7.3.2	成员扩展属性 extattr	281
7.3.3	维护成员信息	282
7.3.4	删除单个成员	283
7.3.5	批量删除成员	284
7.3.6	获取成员信息	284
7.3.7	获取部门成员	286
7.3.8	获取部门成员及详细信息	287
7.4	异步任务管理	289
7.4.1	上传 CVS 文件	290
7.4.2	全量覆盖部门	292
7.4.3	全量覆盖成员	296
7.4.4	jobid 获取异步任务结果	299
7.4.5	callback 接收异步任务通知	302
7.5	标签管理	305
7.5.1	创建标签	305

7.5.2	新增标签成员	307
7.5.3	删除标签成员	310
7.5.4	获取标签成员	313
7.5.5	删除标签	313
7.6	案例：企业通讯录异步维护	314
第 8 章	数据安全访问策略	321
8.1	OAuth 2.0 身份验证	321
8.1.1	获取 code	322
8.1.2	根据 code 获得成员信息	323
8.2	浏览器类型安全访问	325
8.3	全局验证码变量	326
8.4	页面有效期访问	327
8.4.1	JS 定时任务校验	328
8.4.2	事件校验	329
8.5	QPID 消息队列	330
8.5.1	QPID 消息 Hello World	330
8.5.2	QPID 发送 MAP 消息	333
8.5.3	8080 端口问题	336
8.6	代理服务器	337
8.7	企业号服务 IP 白名单	339
8.8	案例：通过 DMZ 服务器获取内网图片	341
第 9 章	数据库及服务器	348
9.1	常用 SQL 语句	348
9.1.1	查询语句	348
9.1.2	新增语句	350
9.1.3	更新语句	350
9.1.4	删除语句	351
9.2	HQL 语句基础语法	351
9.3	HQL 方言处理	354
9.4	Tomcat 服务器	355
9.4.1	在 SDK 中部署	355
9.4.2	8080 端口号冲突	356
9.4.3	内存调整	358
9.4.4	清理数据缓存	358
9.5	JBoss 服务器	359
9.5.1	JBoss 在 SDK 中安装	359
9.5.2	修改 8080 端口	360
9.5.3	JBoss 内存调整	361
9.5.4	发布缓存处理	363
9.6	WebLogic 服务器	363

9.6.1 域的创建.....	363
9.6.2 WebLogic 在 SDK 中安装.....	367
9.6.3 7001 端口号调整.....	368
9.6.4 服务器缓存清理.....	368

第三篇 综合案例

第 10 章 基础应用——企业资讯.....	370
10.1 创建应用.....	371
10.2 获取开发者信息.....	371
10.3 开发实现.....	372
10.3.1 创建数据库 Table.....	372
10.3.2 生成 PO/VO 实体类.....	374
10.3.3 创建工具类 WxUtil.....	379
10.3.4 创建 Web 服务.....	382
10.3.5 Service 处理 Web 请求.....	384
10.4 开启企业资讯应用回调.....	390
10.5 创建最新资讯菜单.....	391
10.6 本章小结.....	391
第 11 章 更进一步：微信考勤.....	392
11.1 场景回顾.....	393
11.2 腾讯地图引入.....	393
11.2.1 腾讯地图 Key 申请.....	394
11.2.2 腾讯地图 Demo.....	395
11.2.3 腾讯地图坐标转换.....	397
11.3 开发实现.....	397
11.3.1 创建微信工具类.....	398
11.3.2 编写回调服务.....	406
11.3.3 考勤信息实体类.....	408
11.3.4 创建业务层服务类.....	409
11.3.5 服务跳转类.....	415
11.3.6 JSP 考勤打卡 Map 页.....	421
11.3.7 考勤查询 JSP 页.....	426
11.3.8 其他考勤页.....	433
11.4 开启微信考勤回调模式.....	435
11.5 绑定可信域名.....	436
11.6 微信考勤应用菜单.....	437
11.7 本章小结.....	437
附录 A 微信表情转换表.....	438
附录 B 返回码说明表.....	441

第 6 章

企业会话模式

企业会话模式，是与企业内部 IM 软件相结合的软件，实现企业内部 IM 软件与微信进行信息传递的方式。通过本章的学习将使读者掌握如何发送单聊、多聊信息，如何接收、解析微信消息，了解如何实现企业 IM 与微信的交流沟通。

本章主要涉及的知识点有：

- 什么是企业会话模式：学习企业会话模式基础知识。
- 会话聊天：学习单聊、多聊的消息格式，学会如何发送单聊、多聊信息。
- 回调消息：学会如何接收、解析微信会话聊天消息。
- 消息免打扰：了解消息免打扰接口的格式及使用。
- 业务与接口的实际应用：通过本章最后的示例，学习企业 IM 与微信接口的开发，熟练掌握微信企业会话聊天。

6.1 企业会话模式介绍

企业会话模式（以下简称“企业会话”）开发相当于简易版的企业号开发，它具有单独的 AccessToken、Secret，以及单独的回调链接，同样需要对主动推送消息的 AccessToken 进行缓存处理，也同样需要对接收到的消息进行解密、解析、响应，实现消息的双向互动。以企业 IM 接收微信消息流程为例，流程图如图 6.1 所示。企业会话就像企业号的“小号”，加密、缓存规则都是一致的，所以本章不会详细介绍缓存、加密等部分，大家可以通过第 3 章、第 4 章学习。

企业会话的优势在于，可以减少企业 IM 的 App 开发成本，通过企业会话服务，不仅企业号成员可基于微企通讯录直接发起微信对话，而且读者还可以通过接口实现与企业 IM 系统的对接，实现双向同步会话，如图 6.2 所示。

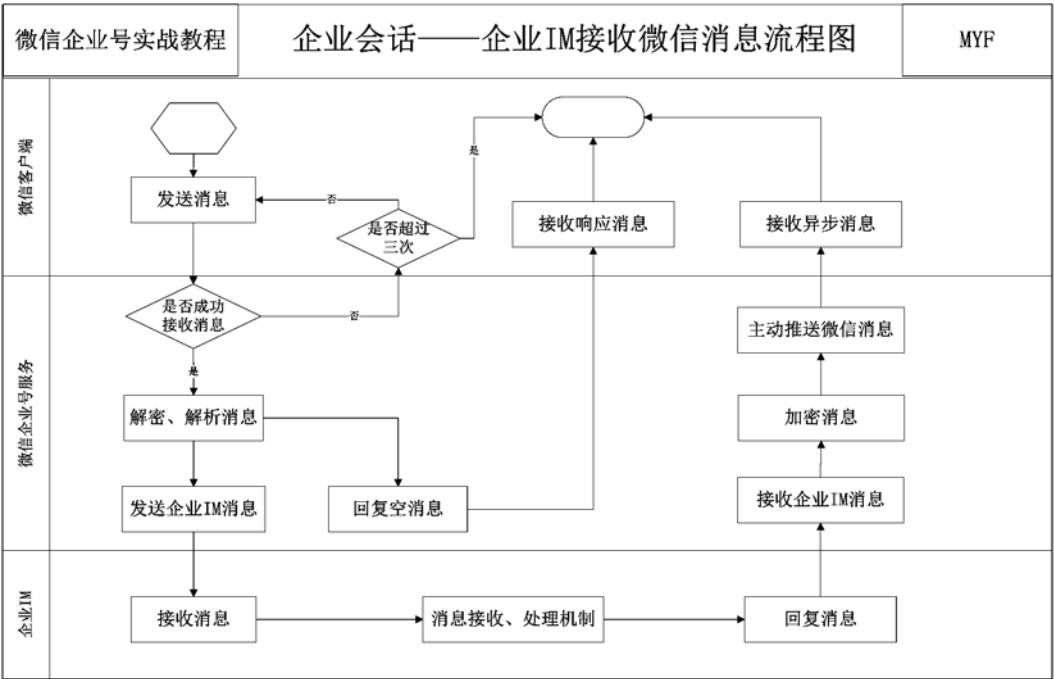


图 6.1 企业 IM 接收微信消息流程图



图 6.2 查找“企业会话”

对于企业号的配置，也需要注意以下几点：

- 企业 IM 中的组织架构需要与企业号中的一致，可以通过异步任务进行定期同步（通讯录异步任务将在第 7 章介绍）。
- 以超级管理员身份开通“企业会话”服务，并开启回调接口（企业会话的开启在 6.2 节中介绍），获得 CorpID 和 Secret。获得的管理组 Secret 为新的接口权限票据，仅支持企业会话内消息互动以及素材管理，不支持应用中的菜单管理权限等。
- 注意配置拥有会话发起人。
- 与被动回调模式一样，企业会话的回调链接需要完成消息响应。对于推送失败的消息，企业将重新推送，推送最大误差为一天。

6.2 开启企业会话

随着企业号的不断更新维护，管理界面也发生了变化，可以根据功能菜单指示进行开启，开始说明如下。

01 登录企业号管理平台，点击“服务中心”，点击“企业会话”，如图 6.3 所示。



图 6.3 查找“企业会话”

02 进入“企业会话启用”页面，单击“开始使用”，选择“使用范围”之后，单击“更多设置”，如图 6.4 所示。



图 6.4 企业会话设置页面

03 单击发起会话权限中的“添加”，设置允许发起会话的人员名单，可以根据部门、标签、成员进行选择，如果是所有人，则单击部门的根节点（即全部部门），如图 6.5 所示。

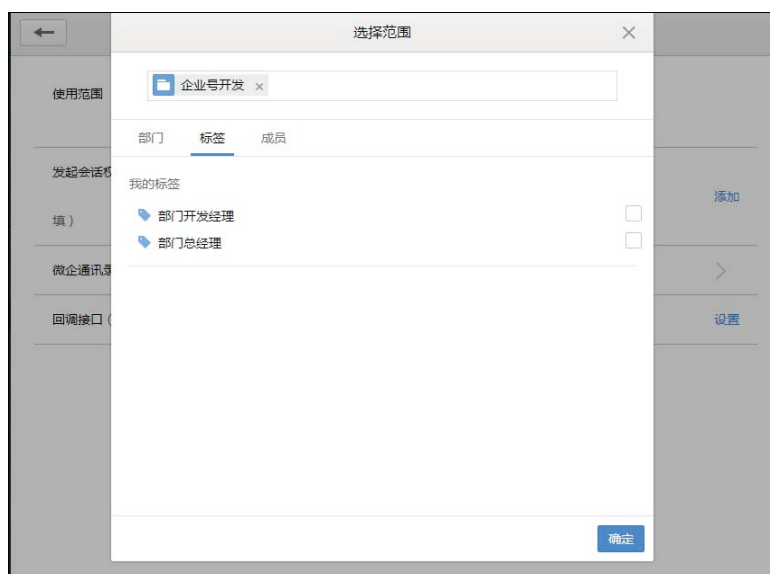


图 6.5 设置会话发起范围

04 设置微企通讯录中成员发起权限，如图 6.6 所示。



图 6.6 微企通讯录发起权限



备注：微企通讯录为企业号创建时默认创建的应用，读者可以关闭默认应用，创建自己的通讯录应用。

05 设置回调链接，设置回调链接之后，企业就能够接收微信会话消息，实现微信会话的同步，如图 6.7 所示。

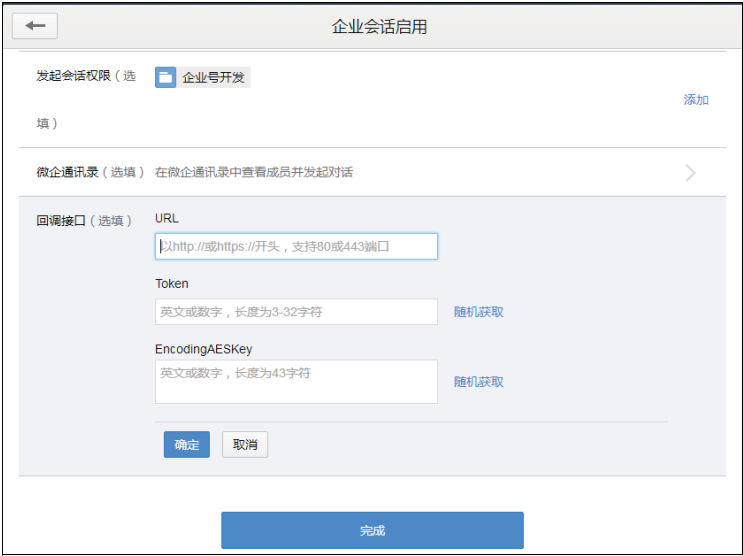


图 6.7 设置企业会话回调链接



备注：企业会话回调开启模式与“第 4 章 被动回调模式”开启方式一致，均需要 GET 验证。通过企业会话回调链接接收微信消息，通过会话 API 接口发送消息，则可以实现微信与企业 IM 的对接。

06 完成链接配置，如图 6.8 所示。

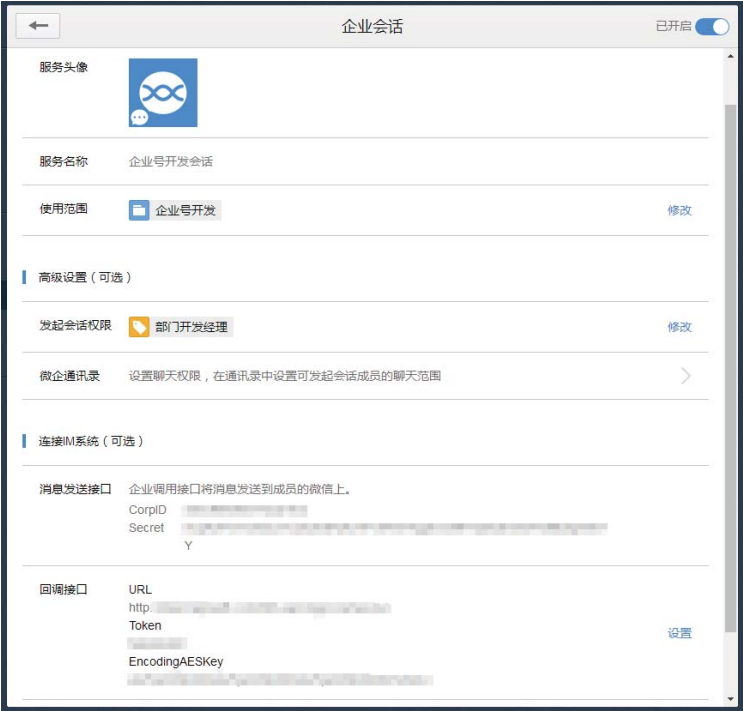


图 6.8 企业会话详细页面

6.3 推送聊天信息

聊天消息的推送主要分为单聊信息和讨论组聊天信息（以下称为多聊）两种，如图 6.9 所示。推送单聊信息时直接向企业号推送发送人和接收人即可，而多聊则需要首先创建讨论组，然后才能够发送信息。对于消息的发送者，不论在单聊还是多聊情况下，都不会收到消息提醒。



图 6.9 企业会话微信客户端界面

注意：在通讯录中但未关注企业号的人员，也能够加入讨论组，也能够对其发送信息，这并不是异常，而是正常需求，方便与企业 IM 更好地对接。

6.3.1 信息推送接口说明

消息发送支持文本、图片、文件、声音以及链接消息，不论用户是否在线都将推送消息。如果接收人不存在，则消息会发送失败，详细说明如下。

(1) Https 请求链接如下：

```
https://qyapi.weixin.qq.com/cgi-bin/chat/send?access_token=ACCESS_TOKEN
```

备注：单聊、多聊消息推送接口相同，都为该接口。

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息类型。

支持文本、图片、语音、文件、链接消息。

(4) 权限设置。

发送者需要具有会话发起权限；接收人需要全部存在。

注意：企业 IM 端发送的消息，在同步到发送者的微信上时，不会有提醒，除发送者之外的其他接收人员，在微信客户端将收到提醒。


(5) 示例代码：

```
/**
 * 企业 IM 端发送的消息，在同步到发送者的微信上时，不会有提醒
 * 除发送者之外的其他接收人员，将收到提醒
 * @param tagId 标签 id
 * @return
 */
public JSONObject send_M_Message(String jsonMessage){
    boolean flag=true;
    //获取微信号
    String token=getTalkTokenFromWx();
    try {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/
            cgi-bin/chat/send?access_token="+token);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity(jsonMessage,
            ContentType.create("text/plain", "UTF-8"));
        httpPost.setEntity(myEntity);
        ResponseHandler<JSONObject> responseHandler = new
ResponseHandler<JSONObject>() {
            public JSONObject handleResponse( final HttpResponse response)
throws ClientProtocolException, IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if (null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);

                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected status:
" + status);
                }
            }
        };
        //返回的 JSON 对象
        JSONObject responseBody = httpClient.execute(httpPost,
responseHandler);
        System.out.println(responseBody);
        return responseBody;
    }catch (Exception e) {
```



```
        e.printStackTrace();
        return null;
    }
}
```

 **备注：**getTalkTokenFromWx 方法为企业会话下的 AccessToken 缓存处理方法，处理方式与主动调用模式下的 AccessToken 一致（3.3 AccessToken 的缓存处理），读者可参照 3.3 节新增两个变量，处理企业会话下的 AccessToken 即可。

6.3.2 聊天消息体结构说明

在开发过程中，需要注意消息体的参数值，除了消息类型不一致导致的参数不同之外，还需要注意单聊、多聊不同场景下的参数值问题。例如，单聊消息 type 值为“single”，多聊消息 type 值为“group”；单聊中 ID 值为接收人企业号中的 ID（即 userid），而多聊中的 ID 为讨论组 ID（讨论组的维护将在 6.3.3 节介绍）。

（1）Text（文字消息）。

文字聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "single",
    "id": "lisi"
  },
  "sender": "zhangsan",
  "msgtype": "text",
  "text":
  {
    "content": "Holiday Request For Pony(http://xxxxxx)"
  }
}
```

单聊、多聊 text 消息体结构详细说明如表 6.1 所示。

表 6.1 单聊、多聊text消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：text
content	是	消息内容

（2）image（图片消息）。

图片聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "group",
    "id": "235364212115767297"
  },
  "sender": "zhangsan",
  "msgtype": "image",
  "image":
  {
    "media_id": "MEDIA_ID"
  }
}
```

备注：图片信息推送的是 media_id，图片信息需要优先上传至微信服务器。由于永久素材有数量限制，因此建议上传临时图片素材。

单聊、多聊 image 消息体结构详细说明如表 6.2 所示。

表 6.2 单聊、多聊image消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：image
media_id	是	图片media_id，可以调用上传素材文件接口获取

(3) file（文件消息）。

文件聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "group",
    "id": "235364212115767297"
  },
  "sender": "zhangsan",
  "msgtype": "file",
  "file":
  {
    "media_id": "MEDIA_ID"
  }
}
```

 备注：文件信息与图片消息基本相同，建议上传临时文件素材。


单聊、多聊 file 消息体结构详细说明如表 6.3 所示。

表 6.3 单聊、多聊file消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：file
media_id	是	图片media_id，可以调用上传素材文件接口获取，文件必须大于4字节

（4）voice（音频消息）。
音频聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "group",
    "id": "235364212115767297"
  },
  "sender": "zhangsan",
  "msgtype": "voice",
  "voice":
  {
    "media_id": "MEDIA_ID"
  }
}
```

 备注：音频信息与图片消息基本相同，建议上传临时文件素材。

单聊、多聊 voice 消息体结构详细说明如表 6.4 所示。


表 6.4 单聊、多聊voice消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：voice
media_id	是	图片media_id，可以调用上传素材文件接口获取，文件大小必须大于4字节

(5) link（链接消息）

链接聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "single",
    "id": "lisi"
  },
  "sender": "zhangsan",
  "msgtype": "link",
  "link":
  {
    "title": "CSDN 博客",
    "description": "link 链接消息描述",
    "url": " http://blog.csdn.net/myfmyfmyfmyf ",
    "thumb_media_id":
"177fIcVBfOLYa703hzBByU1EH3_sdp4hyyaxN4Gfdc-o66vG7k-lXgEacQqfuCcJ-VbZnP1UKJD
F8ig_8Zgh6-g"
  }
}
```

 **备注：**需要注意 title、description 文字勿超过限制数。

单聊、多聊 link 消息体结构详细说明如表 6.5 所示。

表 6.5 单聊、多聊link消息体结构说明

参数	是否必须	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：link
title	是	消息标题，不超过128字节
description	否	消息描述，不超过512字节
url	是	跳转的URL
thumb_media_id	是	图片media_id，可以调用上传素材文件接口获取

6.3.3 创建多聊会话

单人聊天只需向指定的单个人员发送信息，而多人聊天则需要先将信息人“圈”起来，将大家聚集在一起然后发送信息。本节将为读者介绍如何将信息接收人聚集起来，并建立会话讨论组。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/create?access_token=ACCESS_TOKEN


(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体。

此方式是通过接口创建讨论组，如果通过微信客户端创建讨论组，那么读者服务端将收到微信创建讨论组的回调信息，消息请求结构体如下：


```
{
  "chatid": "1",
  "name": "企业应用中心",
  "owner": "zhangsan",
  "userlist": ["zhangsan", "lisi", "wangwu"]
}
```

 **注意：**userlist 中包含 owner（会话创建人）；发起人必须具有发起会话权限；会话中人员必须存在。

(4) 请求参数说明。详细说明如表 6.6 所示。

表 6.6 创建讨论组请求参数说明

参数	是否必需	说明
chatid	是	会话ID。字符串类型，最长32个字符。只允许字符0-9及字母a-z、A-Z。如果值内容为64bit无符号整型：要求值范围在[1, 2^63)之间，[2^63, 2^64)为系统分配会话id区间
name	是	会话标题
owner	是	管理员UserID，必须是该会话UserList的成员之一
userlist	是	会话成员列表，成员用UserID来标识。会话成员必须在3人或以上，1000人以下

 **使用技巧：**对于企业 IM 中存在、但微信企业号中不存在的用户，且只有三个人的会话，可以通过增加“企业号小助手”（此人员为读者虚拟出的、具有微信号的人员）来完成企业会话的创建。

(5) 示例代码。

封装聊天会话处理方法 sendMsg2WX，通过会话创建传递处理链接以及 JSON 数据，示例代码如下：

```
/**
 * 向微信传递 JSON 信息，并返回信息
 * @param url 如：“https://qyapi.weixin.qq.com/cgi-bin/chat/update?
access_token=”
 * @param jsonStr
 * @return
```

```
    */
    public JSONObject sendMsg2WX(String url,String jsonStr){
        //获取微信号
        String token=getTalkTokenFromWx();
        try {
            CloseableHttpClient httpClient = HttpClients.createDefault();
            HttpPost httpPost= new HttpPost(url+token);
            //发送 JSON 格式的数据
            StringEntity myEntity = new StringEntity(jsonStr,ContentType.
create("text/plain", "UTF-8"));
            httpPost.setEntity(myEntity);
            ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
                public JSONObject handleResponse(
                    final HttpResponse response) throws ClientProtocolException,
IOException {
                    int status = response.getStatusLine().getStatusCode();
                    if (status >= 200 && status < 300) {
                        HttpEntity entity = response.getEntity();
                        if(null!=entity){
                            String result= EntityUtils.toString(entity);
                            //根据字符串生成 JSON 对象
                            JSONObject resultObj = JSONObject.fromObject
(result);

                            return resultObj;
                        }else{
                            return null;
                        }
                    } else {
                        throw new ClientProtocolException("Unexpected response
status: " + status);
                    }
                }
            };
            //返回的 JSON 对象
            JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
            System.out.println(responseBody);
            return responseBody;
        }catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

封装会话聊天处理方法:

```
/**
```

```

* 创建微信聊天组 JSON
* @param util 工具类
* @param wxChatId 会话 Id
* @param title 会话标题
* @param owner 会话发起人
* @param userList 会话人员列表
* @param assistIsExit 是否需要小助手
* @return 返回创建信息
*/
private JSONObject createChatWindow(WxUtil util,String wxChatId,String
title,String owner,String userList,boolean assistIsExit){
    //是否需要增加"企业号小助手"
    if(true==assistIsExit){
        userList+=",\""+ WxUtil.ASSIST_CODE+"\"";
    }
    //创建会话 JSON 数据
    String jsonStr="{\"chatid\": \"\"+wxChatId+\"\", \"name\": \"\"+title+
\", \"owner\": \"\"+owner+\"\", \"userlist\": \"\"+userList+\"\"}";
    JSONObject jsobObj= util.sendMsg2WX(WxUtil.CREATE_CHAT_URL, jsonStr);
    return jsobObj;
}

```

 **备注：**企业号小助手（WxUtil.ASSIST_CODE）既可以使用管理员的微信号代替，也可以虚拟一个成员。

（6）返回结果。

返回结果主要以 `errcode` 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```

{
    "errcode": 0,
    "errmsg": "ok"
}

```

6.3.4 修改多聊会话

讨论组可能出现人员增加、修改标题等操作，这就需要会话的修改操作，接口说明如下。

（1）Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/update?access_token=ACCESS_TOKEN

（2）数据请求方式。

POST 方式进行数据请求。

（3）消息请求结构体。

```

{
    "chatid": "296875212115767297",
    "op_user": "muyunfei",

```



```
    "name": "企业应用中心",
    "owner": "zhangsan",
    "add_user_list": ["lisi"],
    "del_user_list": ["wangwu"]
}
```

(4) 请求参数说明。详细说明如表 6.7 所示。

表 6.7 修改讨论组请求参数说明

参数	是否必需	说明
chatid	是	会话ID
op_user	是	操作人UserID
name	否	会话标题
owner	否	管理员UserID必须是该会话userlist的成员之一
add_user_list	否	会话新增成员列表，成员用UserID来标识
del_user_list	否	会话退出成员列表，成员用UserID来标识



备注：chatid 为已经创建的讨论组 ID，不能是读者新生成的 ID。

(5) 示例代码：

```
/**
 * 修改会话信息
 * @param util 工具类
 * @param addList 添加的人员名单
 * @param deleteList 删除的人员名单
 * @param name 会话标题
 * @param owner 拥有者
 * @param oper 操作者
 * @param wxChatId 讨论组 ID
 * @return
 */
private JSONObject updateChatWindow(WxUtil util, List<WxChatMessageGroup>
addList, List<WxChatMessageGroup> deleteList, String name, String owner, String
oper, String wxChatId) {
    String jsonStr="{\"chatid\": \""+wxChatId+"\", \"
        +\"op_user\": \""+owner+"\", \"
        +\"name\": \""+name+"\", \"
        +\"owner\": \""+owner+"\", \"
    String addStr="";
    //将 list 转变成字符串
    if(null!=addList&&0!=addList.size()){
        for (int i = 0; i < addList.size(); i++) {
            if(null!=addStr&&"".equals(addStr)){
                addStr+=",";
            }
        }
    }
}
```

```

        addStr=addStr+"\""+addList.get(i).getUserCode()+"\"";
    }
    jsonStr+="\"add_user_list\":["+addStr+"],"";
}
String deleteStr="";
if(null!=deleteList&&0!=deleteList.size()){
    for (int i = 0; i < deleteList.size(); i++) {
        if(null!=deleteStr&&!"".equals(deleteStr)){
            deleteStr+=",";
        }
        deleteStr=deleteStr+"\""+deleteList.get(i).getUserCode()+"\"";
    }
    jsonStr+="\"del_user_list\":["+deleteStr+"]}";
}
//发送修改会话请求，并获得结果
JSONObject jsobObj=util.sendMsg2WX(WxUtil.UPDATE_CHAT_URL, jsonStr);
return jsobObj;
}

```



备注：sendMsg2WX 方法为封装方法在 6.3.3 节中介绍。

(6) 返回结果。

返回结果主要以 errcode 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```

{
    "errcode": 0,
    "errmsg": "ok"
}

```

6.3.5 退出多聊会话

成员可以通过企业 IM 退出讨论组，这时需要对应的微信客户端也作出相应的响应，退出多聊会话组，此操作需要通过接口实现，接口说明如下。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/quit?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体：

```

{
    "chatid": "296875212115767297",
    "op_user": "muyunfei",
}

```


(4) 请求参数说明。详细说明如表 6.8 所示。

表 6.8 退出讨论组请求参数说明

参数	是否必需	说明
chatid	是	会话ID
op_user	是	操作人UserID

(5) 示例代码:

```
//退出会话
String jsonStr3="{\"chatid\": \"wx12345678\", \"op_user\": \"wangjiankun\"}";
demo.sendMsg2WX(URL_EXIT_CHAT_GROUP, jsonStr3);
```

 **备注:** sendMsg2WX 方法为封装方法, 在 6.3.3 节中曾介绍过。URL_EXIT_CHAT_GROUP 为请求链接全局变量。

(6) 返回结果。

返回结果主要以 errcode 参数为主, 0 代表成功, 1 代表失败, 示例结果如下所示:

```
{
  "errcode": 0,
  "errmsg": "ok"
}
```

6.3.6 获取多聊会话信息

获取企业会话讨论组详细信息, 接口说明如下。

(1) Https 请求链接如下:

https://qyapi.weixin.qq.com/cgi-bin/chat/get?access_token=ACCESS_TOKEN&chatid=CHATID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 请求参数说明。详细说明如表 6.9 所示。


表 6.9 获取讨论组信息请求参数说明

参数	是否必需	说明
Access_token	是	企业会话接口票据
chatid	是	讨论组ID

(4) 示例代码:

```
//获得会话信息
String url="https://qyapi.weixin.qq.com/cgi-bin/chat/get?chatid=
wx12345678&access_token=";
String jsonStr4="";
JSONObject result = demo.sendMsg2WX(url, jsonStr4);
System.out.println(result.toString());
//获得 chatid
```

```
System.out.println(result.getJSONObject("chat_info").get("chatid"));
//获得 userlist 中第一条数据
System.out.println(result.getJSONObject("chat_info").getJSONArray("userl
ist").get(0));
```

 **备注：**sendMsg2WX 方法为封装方法，在 6.3.3 节中曾介绍过。获得结果为 JSONObject 对象（net.sf.json.JSONObject），对于 chat_info 对象，可以使用 getJSONObject("chat_info") 方法获得；对于 userlist 数组，可以通过 getJSONArray("userlist")获得。

(5) 返回结果。

返回结果主要以 errcode 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "chat_info":
  {
    "chatid": "235364212115767297",
    "name": "企业应用中心",
    "owner": "zhangsan",
    "userlist": ["zhangsan", "lisi", "wangwu"]
  }
}
```

返回结果详细说明如表 6.10 所示。

表 6.10 讨论组获取信息参数说明

参数	说明
errcode	返回码
errmsg	返回码的文本描述信息
chat_info	会话信息
chatid	会话ID
name	会话标题
owner	管理员UserID
userlist	会话成员列表，成员用UserID来标识

6.3.7 清除未读会话状态

标识会话状态为已读，消除未读会话状态，接口说明如下。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/clearnotify?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。


(3) 消息请求结构体：

```
{
  "op_user": "zhangsan",
  "chat": {
    "type": "single",
    "id": "lisi"
  }
}
```

(4) 请求参数说明。详细说明如表 6.11 所示。


表 6.11 清除未读会话状态请求参数说明

参数	是否必需	说明
op_user	是	会话所有者的UserID
chat	是	会话
type	是	会话类型: single group, 分别表示: 单聊 多聊
id	是	会话值, 为userid chatid, 分别表示: 成员ID 会话ID

 **备注:** 当 type 值为 single 时, id 值为成员 id (userid); 当 type 值为 group 时, id 值为会话 id (chatid)。

(5) 示例代码:

```
//清除未读状态
String jsonStr3="{\"op_user\": \"muyunfei\", \"
    +\"chat\":{\"type\": \"group\", \"id\": \"wx12345678\"}}";
demo.sendMsg2WX(URL_CLEAR_CHAT_GROUP, jsonStr3);
```

 **备注:** sendMsg2WX 方法为封装方法, 在 6.3.3 节中曾介绍过。URL_CLEAR_CHAT_GROUP 全局变量为清除未读信息的请求链接。

(6) 返回结果。

返回结果主要以 errcode 参数为主, 0 代表成功, 1 代表失败, 示例结果如下所示:

```
{
  "errcode": 0,
  "errmsg": "ok"
}
```

6.3.8 会话消息免打扰

该接口主要用于设置成员微信客户端是否接收消息提醒, 接口说明如下。

(1) Https 请求链接如下:

https://qyapi.weixin.qq.com/cgi-bin/chat/setmute?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体：

```
{
  "user_mute_list":
  [
    {
      "userid": "zhangsan",
      "status": 0
    },
    {
      "userid": "lisi",
      "status": 1
    }
  ]
}
```



备注：该接口可用于以下场景。

- a. 成员在企业 IM 设置微信“消息免打扰”。
- b. 当成员企业 IM 在线时，微信客户端可以设置“消息免打扰”（也可以不向微信客户端发送信息，由读者制定规则）。当成员企业 IM 离线或者退出时，可以关闭“消息免打扰”，进行微信客户端提醒。

(4) 请求参数说明。详细说明如表 6.12 所示。

表 6.12 清除未读会话状态请求参数说明

参数	是否必需	说明
user_mute_list	是	免打扰成员数组，最大支持10000个成员
userid	是	成员UserID
status	是	免打扰状态：0为关闭，1为打开，默认为0。当打开时所有消息不提醒；当关闭时，以成员对会话的设置为准，如图6.10所示



备注：消息免打扰接口，可一次性提交不超过 10000 人。

(5) 示例代码：

```
//消息免打扰
String jsonStr6="{\"user_mute_list\":["
    +\"{\"userid\": \"muyunfei\", \"status\": 1},\"
    +\"{\"userid\": \"yuchunyang\", \"status\": 0}\"
    +\"]}\";
demo.sendMsg2WX(URL_ALERT_CHAT_GROUP, jsonStr6);
```



图 6.10 成员会话设置



备注：sendMsg2WX 方法为封装方法，在 6.3.3 节中曾介绍过。返回结果中，invaliduser 参数为无效成员，其余为有效成员，正常执行消息免打扰操作。

(6) 返回结果。

返回结果主要以 errcode 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "invaliduser": ["zhangsan"]
}
```

6.4 接收聊天信息

消息有发送就有接收，6.3 节我们学习了会话消息的推送，本节将介绍微信会话聊天信息的接收。

6.4.1 信息接收接口说明

接收微信聊天信息，与被动回调模式基本一致，当成员在微信端发消息或更改会话成员时，企业号会推送相应的消息、事件，获得消息之后需要经过 URL 验证、消息解密、消息响应，不同之处在于，回复消息时为明文消息并不是加密消息，而且接收到的数据消息包含多条消息或事件。对于信息的接收需要注意以下几点：

- 当连接失败、请求超时等回调失败时，最大重试间隔为 20 分钟，最大重试时长为 1 天。

- 回复响应消息时，需要明文回复 PackageId 节点值。
- 对于每条消息的排重，可以通过 Item 节点中的 MsgId 或者 FromUserName+ CreateTime 的方式排重。
- 回调数据包中，每条消息为数据包中的一个 Item 节点，数据包包含多条普通消息或事件消息（即多个 Item 节点）。
- 聊天消息的接收与被动回调模式 URL 验证、解密方式一致，读者可以参照第4章学习，接收到的会话信息如下：

```
<xml>
  <Encrypt><![CDATA[ENCRYPT_DATA]]></Encrypt>
  <ToUserName>CORPID</ToUserName>
  <AgentType>chat</AgentType>
</xml>
```

ENCRYPT_DATA 为密文消息，需要解密，解密后的数据包如下所示：

```
<xml>
  <AgentType>chat</AgentType>
  <ToUserName>CORPID</ToUserName>
  <ItemCount>2</ItemCount>
  <PackageId>3156175696255</PackageId>
  <Item>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1348831860</CreateTime>
    <MsgType><![CDATA[event]]></MsgType>
    <Event><![CDATA[update_chat]]></Event>
    <Name><![CDATA[事件消息更新会话]]></Name>
    <Owner><![CDATA[zhangsan]]></Owner>
    <AddUserList><![CDATA[zhaoliu]]></AddUserList>
    <DelUserList><![CDATA[lisi|wangwu]]></DelUserList>
    <ChatId><![CDATA[235364212115767297]]></ChatId>
  </Item>
  <Item>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1348831860</CreateTime>
    <MsgType><![CDATA[event]]></MsgType>
    <Event><![CDATA[事件消息创建会话]]></Event>
    <ChatInfo>
      <ChatId><![CDATA[235364212115767297]]></ChatId>
      <Name><![CDATA[企业应用中心]]></Name>
      <Owner>zhangsan</Owner>
      <UserList>zhangsan|lisi|wangwu</UserList>
    </ChatInfo>
  </Item>
</xml>
```

6.4.2 普通消息结构体说明


普通消息分为六类，包括 text 消息、image 消息、voice 消息、file 消息、link 消息和 location 消息，每类消息又分为单聊消息和多聊消息。对于单聊、多聊信息，只需注意 Receiver 节点即可，单聊信息 Receiver 节点中的 Type 为 single、ID 为成员 userid，而多聊信息 Receiver 节点中的 Type 为 group、ID 为讨论组 id，消息的结构体详细说明如下。

(1) text 文字消息（结构体详细说明如表 6.13 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[text]]></MsgType>
  <Content><![CDATA[this is a test]]></Content>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.13 企业会话text回调消息参数说明

参数	说明
FromUserName	发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：text
Content	消息内容，支持表情，表情见附录A
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：成员ID 会话ID

 **备注：**普通消息可以直接使用 MsgId 进行排重。


(2) image 图片消息（结构体详细说明如表 6.14 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[image]]></MsgType>
  <PicUrl><![CDATA[this is a url]]></PicUrl>
  <MediaId><![CDATA[media_id]]></MediaId>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
```

```
</Receiver>
</Item>
```

表 6.14 企业会话image回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：image
PicUrl	图片链接
MediaId	图片media_id，可以调用获取素材文件接口拉取数据
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：成员ID 会话ID

 **备注：**通过 MediaId 获得图片、音频、文件等素材，读者可以通过 3.6 节学习，本节不再介绍。

（3）voice 音频消息（结构体详细说明如表 6.15 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[voice]]></MsgType>
  <MediaId><![CDATA[media_id]]></MediaId>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.15 企业会话voice回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：voice
MediaId	语音media_id，可以调用获取素材文件接口拉取数据
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：用户ID 会话ID

（4）file 文件消息（结构体详细说明如表 6.16 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[file]]></MsgType>
  <MediaId><![CDATA[media_id]]></MediaId>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.16 企业会话file回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：file
MediaId	语音media_id，可以调用获取素材文件接口拉取数据
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：用户ID 会话ID



备注：file 消息与 link 消息成员可以通过转发或者我的收藏发送消息。

（5）link 链接消息（结构体详细说明如表 6.17 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[link]]></MsgType>
  <Title><![CDATA[TITLE]]></Title>
  <Description><![CDATA[DESCRIPTION]]></Description>
  <Url><![CDATA[URL]]></Url>
  <PicUrl><![CDATA[PIC_URL]]></PicUrl>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.17 企业会话link回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID

续表

参数	说明
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：link
Title	标题
Description	描述
Url	链接
PicUrl	图片链接
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：用户id 会话id



备注：通过 URL 链接利用 httpclient 或者 httpurlconnection 等方式实现数据抓取。

6.4.3 事件消息结构体说明

接收聊天消息的时间与推送聊天事件消息一致，均为多聊会话事件，包括：创建会话事件、修改会话事件和退出会话事件，每条事件消息同样作为回调数据包中的一个 Item 节点，事件详细说明如下。

（1）创建会话事件（结构体详细说明如表 6.18 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[create_chat]]></Event>
  <ChatInfo>
    <ChatId><![CDATA[235364212115767297]]></ChatId>
    <Name><![CDATA[企业应用中心]]></Name>
    <Owner>zhangsan</Owner>
    <UserList>zhangsan|lisi|wangwu</UserList>
  </ChatInfo>
</Item>
```

表 6.18 创建会话事件回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，此时固定为：create_chat
ChatId	会话ID
Name	会话标题

续表

参数	说明
Owner	管理员UserID
UserList	会话成员列表，成员用UserID标识，成员间以竖线“ ”分隔



使用技巧：UserList 中成员 ID 使用“|”分隔，可以使用 `String[] userArray=userStr.split("|");` 方法进行分隔得到 `String` 数组。

(2) 修改会话事件（结构体详细说明如表 6.19 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[update_chat]]></Event>
  <Name><![CDATA[企业应用中心]]></Name>
  <Owner><![CDATA[zhangsan]]></Owner>
  <AddUserList><![CDATA[zhaoliu]]></AddUserList>
  <DelUserList><![CDATA[lisi|wangwu]]></DelUserList>
  <ChatId><![CDATA[235364212115767297]]></ChatId>
</Item>
```

表 6.19 修改会话事件回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，此时固定为：update_chat
Name	会话名称
Owner	会话所有者（管理员）
AddUserList	会话新增成员列表，成员间以竖线分隔，如：zhangsan lisi
DelUserList	会话删除成员列表，成员间以竖线分隔，如：zhangsan lisi
ChatId	会话ID

(3) 退出会话事件（结构体详细说明如表 6.20 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[quit_chat]]></Event>
  <ChatId><![CDATA[235364212115767297]]></ChatId>
</Item>
```

表 6.20 退出会话事件回调消息参数说明

参数	说明
FromUserName	信息发送人, 成员UserID
CreateTime	消息创建时间 (整型)
MsgType	消息类型, 此时固定为: event
Event	事件类型, 此时固定为: quit chat
ChatId	会话ID

6.5 案例：企业 IM 与微信的对接

通过对前几节的学习, 我们已经了解了什么是企业会话模式, 以及企业会话模式中的各个接口, 本节将介绍开发企业 IM 与微信对接的中间服务。为什么需要创建中间服务, 直接在企业 IM 中集成是否可行? 企业 IM 中可以直接集成, 但为了增加系统的独立性和容灾性, 防止中间服务的异常, 而影响整体企业 IM。其次, 企业 IM 多以内网服务部署, 而微信为外网服务, 中间服务的创建多以 DMZ 区 (危险管理区, 将在第 8 章介绍) 为主, 所以创建中间服务也是必要的。


微信与企业 IM 互动的中间服务分为两类: 一类是转发企业 IM 的消息服务类, 这里我们命名为 `sendMessageServlet.java`; 另一类则是转发微信消息的消息服务类, 这里我们命名为 `ChatMessageServlet.java`。两个服务类的详细说明如下。

(1) 转发企业 IM 消息服务类 `sendMessageServlet.java`。

企业 IM 消息服务数据传递以 JSON 格式数据流形式传递, 使用 `req.getInputStream()` 获得数据流进行传递:

```
/**
 * 企业 IM, 信息服务类
 * 接收企业 IM 信息, 并转发至微信客户端
 */
public class sendMessageServlet extends HttpServlet {
    protected void doGet (HttpServletRequest arg0, HttpServletResponse arg1)
        throws ServletException, IOException {
        this.doPost (arg0, arg1);
    }
    protected void doPost (HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        //从输入流读取内容
        BufferedReader br = new BufferedReader(new InputStreamReader
(req.getInputStream()));
        String line = null;
        StringBuilder sb = new StringBuilder();
        while ((line = br.readLine()) != null) {
            sb.append(line);
        }
    }
}
```

```
String content=sb.toString();
//正则去除\
content=content.replaceAll("\\\\", "\\\\");
//获取调用方法
String action =req.getParameter("action");
String resultMsg="";//返回结果
if ("sendMessage".equals(action)) {
    //发送单个消息
    resultMsg=sendMessage(content);
}else if ("sendGroupMessage".equals(action)) {
    //
    resultMsg=sendGroupMessage(content);
}
//返回输出结果
OutputStream out = resp.getOutputStream();
out.write(resultMsg.getBytes());
out.flush();
out.close();
//resp.getOutputStream();
}
}
```

 **备注：**sendMessage 发送单聊信息方法与 sendGroupMessage 发送多聊信息可以参照 6.3.1 节创建。对于特殊字符 “\” 的转义可以使用 replaceAll("\\\\", "\\\\")。

在 web.xml 中创建 servlet 服务连接，示例代码如下：

```
<!-- 企业 IM 消息发送接口 -->
<servlet>
    <servlet-name>webservice1</servlet-name>
    <servlet-class>
        com.webservice.sendMessageServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>webservice1</servlet-name>
    <url-pattern>/webservice.slt</url-pattern>
</servlet-mapping>
```

测试连接示例代码如下：

```
public class TestMain {
    public static void main(String[] args) {
        //m+单人消息
        //String jsonMessage="{\"mchatid\":\"1\", \"receiver\":\"muyunfei\",
        \"sender\":\"xinyueqiao\", \"msgtype\":\"text\", \"content\":\"特殊符号\\\"}\"";
        //m+讨论组
```



```

String jsonMessage="{\"mchatid\": \"1\", \"
    +\"name\": \"小伙伴\", \"
    +\"owner\": \"muyunfei\", \"
    +\"userlist\": [\"xinyueqiao\", \"muyunfei\",
    \"yuyang2\", \"songyanqin\", \"kangxiyao\"], \"
    +\"sender\": \"xinyueqiao\", \"
    +\"msgtype\": \"text\", \"
    +\"content\": \"特殊符号: + - * % ( ) {} !, #, $, &,
    ', (, ), *, +, ,, -, ., /, :, ;, =, ?, @, _, ~, 0-9, a-z, A-Z, \\ \r\n 表情/:, @f/:pd
    \"}";

try {
    CloseableHttpClient httpClient = HttpClients.createDefault();
    //单聊
    //HttpPost httpPost= new HttpPost("http://服务地址/demo/
//webservice.slt?action=sendSingleMessage");
    HttpPost httpPost= new HttpPost("http://服务地址/demo/webservice.
slt?action=sendGroupMessage");
    //发送 JSON 格式的数据
    StringEntity myEntity = new StringEntity(jsonMessage, ContentType.
create("text/json", "GBK"));
    httpPost.setEntity(myEntity);
    ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
        public JSONObject handleResponse(
            final HttpResponse response) throws ClientProtocolException,
IOException {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                if (null!=entity) {
                    String result= EntityUtils.toString(entity);
                    //根据字符串生成 JSON 对象
                    JSONObject resultObj = JSONObject.fromObject
(result);

                    return resultObj;
                } else {
                    return null;
                }
            } else {
                throw new ClientProtocolException("Unexpected: " + status);
            }
        }
    };
    JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
    System.out.println(responseBody);
} catch (Exception e) {

```

```
        e.printStackTrace();
    }
}
}
```

(2) 转发微信消息服务类 ChatMessageServlet.java。

doGet 方法用于配置回调链接时的链接有效性验证，doPost 方法用于接收真正的数据包。数据包经过解密之后将获得多条 Item 节点数据，对 Item 解析处理即可，ChatMessageServlet 示例代码如下：

```
/**
 * 企业会话，信息回调服务类
 * 接收微信客户端消息，并转发至企业 IM
 */
public class ChatMessageServlet extends HttpServlet {
    private static final long serialVersionUID = 4440739483644821986L;
    /**
     * 请求校验（确认请求来自微信服务器）
     */
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //微信加密签名
        String signature = request.getParameter("msg_signature");
        System.out.println("signature:"+signature);
        //时间戳
        String timestamp = request.getParameter("timestamp");
        System.out.println("timestamp:"+timestamp);
        //随机数
        String nonce = request.getParameter("nonce");
        System.out.println("nonce:"+nonce);
        //随机字符串
        String echostr = request.getParameter("echostr");
        System.out.println("echostr:"+echostr);
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.RESP_MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);

            String sEchoStr; //需要返回的明文
            sEchoStr = wxcpt.VerifyURL(signature, timestamp, nonce, echostr);
            System.out.println("verifyurl echostr: " + sEchoStr);
            //验证 URL 成功，将 sEchoStr 返回
            PrintWriter out = response.getWriter();
            out.write(sEchoStr);
            out.flush();
        }
    }
}
```

```

        out.close();
    } catch (Exception e) {
        //验证 URL 失败, 错误原因请查看异常
        e.printStackTrace();
    }
}

/**
 * 处理微信服务器发来的消息
 */
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    //读取消息, 执行消息处理
    //微信加密签名
    String sReqMsgSig = request.getParameter("msg_signature");
    //时间戳
    String sReqTimeStamp = request.getParameter("timestamp");
    //随机数
    String sReqNonce = request.getParameter("nonce");
    String sToken = WxUtil.RESP_MESSAGE_TOKEN;
    String sCorpID = WxUtil.RESP_MESSAGE_CORPID;
    String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
    try {
        //POST 请求的密文数据
        //sReqData = HttpUtils.PostData();
        ServletInputStream in = request.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader
(in));

        String sReqData="";
        String itemStr="";//作为输出字符串的临时串, 用于判断是否读取完毕
        while(null!=(itemStr=reader.readLine())){
            sReqData+=itemStr;
        }
        //对消息进行处理获得明文
        WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey, sCorpID);
        String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce, sReqData);
        //输出解密后的文件
        System.out.println("after decrypt msg: " + sMsg);
        //TODO: 解析出明文 XML 标签的内容进行处理
        //For example:
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        StringReader sr = new StringReader(sMsg);
        InputSource is = new InputSource(sr);
        Document document = db.parse(is);
        Element root = document.getDocumentElement();
    }
}

```

```
//PackageId,用于回复消息
NodeList nodelist_msgType = root.getElementsByTagName("PackageId");
String contentPackageId = nodelist_msgType.item(0).getTextContent();
System.out.println("-----contentPackageId:"+contentPackageId);
//信息节点个数
NodeList itemCount_note=root.getElementsByTagName("ItemCount");
long itemCount=Long.valueOf(itemCount_note.item(0).getTextContent()+"");
//Item 列表, 遍历信息节点
NodeList itemNodeRootList = root.getElementsByTagName("Item");
for (int i = 0; i < itemCount; i++) {
    //查看数据库 FromUserName+CreateTime 或 MsgID, 进行排重
    //重复消息, 直接结束本次循环, 进行下次循环 (continue)
    //数据库保存 FromUserName+CreateTime 或 MsgID
    NodeList itemNodeRoot = itemNodeRootList.item(i).getChildNodes();
    String msgType="";//消息类型
    for (int j = 0; j < itemNodeRoot.getLength(); j++) {
        if("MsgType".equals(itemNodeRoot.item(j).getNodeName()+"")){
            msgType=itemNodeRoot.item(j).getTextContent();
            break;
        }
    }
    if("event".equals(msgType)){
        //如果是事件
        executeEventMsg(itemNodeRoot);
    }else if("text".equals(msgType)){
        //如果是文本消息
    }else if("image".equals(msgType)){
        //如果是图片消息
    }else if("voice".equals(msgType)){
        //如果是声音消息
    }
}
//输出, 明文回复 PackageId
PrintWriter out = response.getWriter();
out.write(contentPackageId);
out.flush();
out.close();
} catch (Exception e) {
    e.printStackTrace();
}
}

/**
 * 处理事件消息
 * @param itemNodeRoot
 */
```

```

public static void executeEventMsg( NodeList itemNodeRoot){
    HashMap<String, String> map = new HashMap<String, String>();
    for (int j = 0; j < itemNodeRoot.getLength(); j++) {
        //放入 map 中
        map.put(itemNodeRoot.item(j).getNodeName(), itemNodeRoot.item(j).
getTextContent()+"");
    }
    //获得事件类型
    String eventType=map.get("Event");
    if(eventType.equals("create_chat")){
        //创建会话事件
        System.out.println("这是创建会话事件");
        String chatId="";//讨论组 ID
        String name="";//讨论组标题
        String owner="";//管理员 UserID
        String[] userArray=null;
        String userStr = "";
        //获得创建信息
        NodeList chatInfo;
        for (int j = 0; j < itemNodeRoot.getLength(); j++) {
            //放入 map 中
            if("ChatInfo".equals(itemNodeRoot.item(j).getNodeName())){
                chatInfo = itemNodeRoot.item(j).getChildNodes();
                //获得讨论组信息
                for (int i = 0; i < chatInfo.getLength(); i++) {
                    if("ChatId".equals(chatInfo.item(i).getNodeName()+
"")){
                        chatId=itemNodeRoot.item(i).getTextContent();
                    }else if("Name".equals(chatInfo.item(i).getNodeName()+
+"")){
                        name=itemNodeRoot.item(i).getTextContent();
                    }else if("Owner".equals(chatInfo.item(i).
getNodeName()+"")){
                        owner=itemNodeRoot.item(i).getTextContent();
                    }else if("UserList".equals(chatInfo.item(i).
getNodeName()+"")){
                        //会话成员列表, 成员用 UserID 标识, 成员间以竖线 "|" 分隔
                        userStr=itemNodeRoot.item(i).getTextContent();
                        if(null!=userStr&&"!".equals(userStr)){
                            userArray=userStr.split("|");
                        }
                    }
                }
                break;
            }
        }
    }
}

```

```
        //信息处理，根据不同的企业 IM 接口进行处理
        System.out.println("chatId:"+chatId+" name:"+name+" owner:"+
owner+" userStr:"+userStr);
    }else if(eventType.equals("update_chat")){
        //修改会话人员
        System.out.println("这是修改会话人员事件");
    }else if(eventType.equals("quit_chat")){
        //退出会话
        System.out.println("这是退出会话事件");
    }else if(eventType.equals("subscribe")){
        //订阅事件，开启企业会话
        System.out.println("这是订阅事件");
    }else if(eventType.equals("unsubscribe")){
        //取消订阅事件，关闭企业会话
        System.out.println("这是取消订阅事件");
    }
}
}
```



备注:当接收到的微信客户端消息过多或者因其他原因无法及时回应时,需要回复空消息,进行异步处理。