Wechat

一个简单基础的微信公众号接口组件

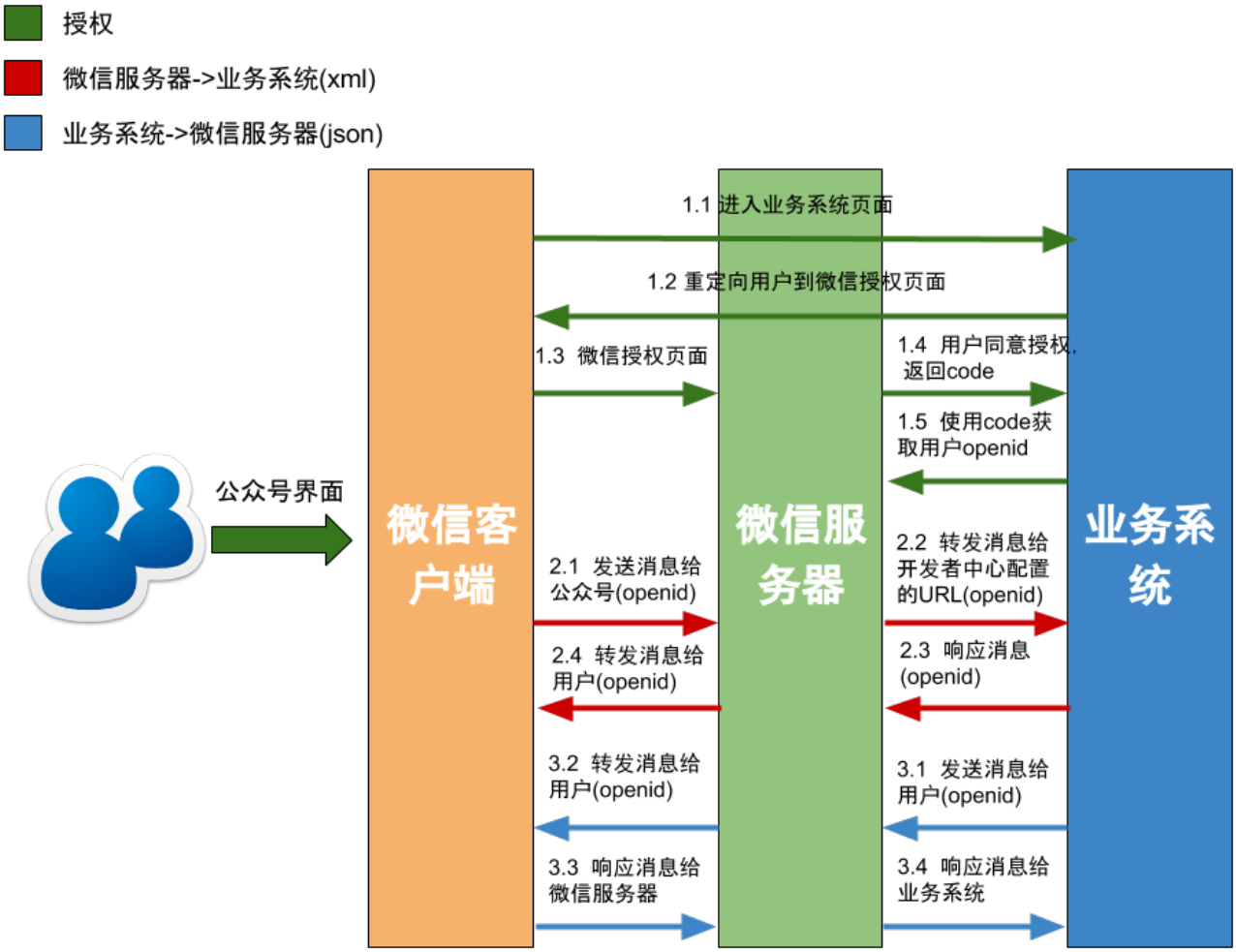
- 包引入

```Markup
<dependency>
    <groupId>me.hao0</groupId>
    <artifactId>wechat</artifactId>
    <version>1.4.0</version>
</dependency>
```

- 依赖包，注意引入项目时是否需要**exclude**:

```Markup
<dependency>
    <groupId>com.github.kevinsawicki</groupId>
    <artifactId>http-request</artifactId>
    <version>6.0</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.4.2</version>
</dependency>
<!-- 1.3.0之后引入 -->
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>18.0</version>
</dependency>
```

- 业务系统与微信公众号交互图阐述:



- API基本用法:

```java
Wechat wechat =
    WechatBuilder.newBuilder("appId", "appSecret")
                 .conf1()  // 其他可选配置
                 ...
                 .build();
// 同步调用
wechat.base().accessToken();

// 异步调用
wechat.base(). accessToken(Callback<T>);
```

- Wechat中包含以下组件:
  - 基础: `base()`
  - 用户: `user()`
  - 菜单: `menu()`
  - 多客服: `cs()`
  - 消息: `msg()`

- 二维码: `qr()`
- 素材: `material()`
- JS调用相关: `js()`
- 数据统计(TODO): `data()`

- API介绍:
  - **`base()`**:

```java
/**
 * 构建授权跳转URL
 * @param redirectUrl 授权后的跳转URL(我方服务器URL)
 * @param quiet 是否静默: true: 仅获取openId, false: 获取openId和个人信息(需用户手
 * @return 微信授权跳转URL
 * @throws UnsupportedEncodingException
 */
String authUrl(String redirectUrl, Boolean quiet)

/**
 * 获取accessToken(失效为2小时, 应该尽量临时保存一个地方, 每隔一段时间来获取)
 * @return accessToken, 或抛WechatException
 */
String accessToken()

/**
 * 获取用户openId
 * @param code 用户授权的code
 * @return 用户的openId, 或抛WechatException
 */
String openId(String code)

/**
 * 获取微信服务器IP列表
 * @param accessToken accessToken
 * @return 微信服务器IP列表, 或抛WechatException
 */
List<String> ip(String accessToken);
```

  - **`user()`**:

```java
/**
 * 创建用户分组
 * @param accessToken accessToken
 * @param name 名称
 * @return 分组ID, 或抛WechatException
 */
```

```
Boolean createGroup(String accessToken, String name)

/**
 * 获取所有分组列表
 * @param accessToken accessToken
 * @return 分组列表，或抛WechatException
 */
List<Group> getGroup(String accessToken)

/**
 * 删除分组
 * @param accessToken accessToken
 * @param id 分组ID
 * @return 删除成功返回true，或抛WechatException
 */
Boolean deleteGroup(String accessToken, Integer id)

/**
 * 更新分组名称
 * @param accessToken accessToken
 * @param id 分组ID
 * @param newName 分组新名称
 * @return 更新成功返回true，或抛WechatException
 */
Boolean updateGroup(String accessToken, Integer id, String newName)

/**
 * 获取用户所在组
 * @param accessToken accessToken
 * @param openId 用户openId
 * @return 组ID，或抛WechatException
 */
Integer getUserGroup(String accessToken, String openId)

 /**
 * 移动用户所在组
 * @param accessToken accessToken
 * @param openId 用户openId
 * @param groupId 组ID
 * @return 移动成功返回true，或抛WechatException
 */
Boolean mvUserGroup(String accessToken, String openId, Integer groupId)

/**
 * 拉取用户信息(若用户未关注，且未授权，将拉取不了信息)
 * @param accessToken accessToken
 * @param openId 用户openId
 * @return 用户信息，或抛WechatException
```

```
 */
User getUser(String accessToken, String openId)

/**
 * 备注用户
 * @param accessToken accessToken
 * @param openId 用户openId
 * @param remark 备注
 * @return 备注成功返回true，或抛WechatException
 */
Boolean remarkUser(String accessToken, String openId, String remark)
```

- `menu()`：

```Java
/**
 * 查询菜单
 * @param accessToken accessToken
 * @return 菜单列表
 */
List<Menu> get(String accessToken)

/**
 * 创建菜单
 * @param accessToken 访问token
 * @param jsonMenu 菜单json
 * @return 创建成功返回true，或抛WechatException
 */
Boolean create(String accessToken, String jsonMenu)

/**
 * 删除菜单
 * @param accessToken accessToken
 * @return 删除成功返回true，或抛WechatException
 */
Boolean delete(String accessToken)
```

- `cs()`：

```Java
/**
 * 添加客服账户
 * @param accessToken accessToken
 * @param account 登录帐号(包含域名)
 * @param nickName 昵称
 * @param password 明文密码
 * @return 添加成功返回true，反之false
 */
Boolean createAccount(String accessToken, String account, String nickName, S
```

```
/**
 * 更新客服账户
 * @param accessToken accessToken
 * @param account 登录帐号(包含域名)
 * @param nickName 昵称
 * @param password 明文密码
 * @return 添加成功返回true，或抛WechatException
 */
Boolean updateAccount(String accessToken, String account, String nickName, S

/**
 * 删除客服账户
 * @param accessToken accessToken
 * @param kfAccount 客服登录帐号(包含域名)
 * @return 添加成功返回true，或抛WechatException
 */
Boolean deleteAccount(String accessToken, String kfAccount)

/**
 * 构建转发客服的XML消息(指定一个在线的客服，若该客服不在线，消息将不再转发给其他在线客服)
 * @param openId 用户openId
 * @param kfAccount 客服帐号(包含域名)
 * @return 转发客服的XML消息
 */
String forward(String openId, String kfAccount)

/**
 * 查询客服聊天记录
 * @param accessToken accessToken
 * @param pageNo 页码
 * @param pageSize 分页大小
 * @param startTime 起始时间
 * @param endTime 结束时间
 * @return 客服聊天记录，或抛WechatException
 */
List<MsgRecord> getMsgRecords(String accessToken, Integer pageNo, Integer pa

/**
 * 创建会话(该客服必需在线)
 * @param openId 用户openId
 * @param kfAccount 客服帐号(包含域名)
 * @param text 附加文本
 * @return 创建成功返回true，或抛WechatException
 */
Boolean createSession(String accessToken, String openId, String kfAccount, S

/**
```

```
 * 关闭会话
 * @param openId 用户openId
 * @param kfAccount 客服帐号(包含域名)
 * @param text 附加文本
 * @return 关闭成功返回true，或抛WechatException
 */
Boolean closeSession(String accessToken, String openId, String kfAccount, St

/**
 * 获取用户的会话状态
 * @param accessToken accessToken
 * @param openId 用户openId
 * @return 客户的会话状态，或抛WechatException
 */
UserSession getUserSession(String accessToken, String openId)

/**
 * 获取客服的会话列表
 * @param accessToken accessToken
 * @param kfAccount 客服帐号(包含域名)
 * @return 客服的会话列表，或抛WechatException
 */
List<CsSession> getCsSessions(String accessToken, String kfAccount)

/**
 * 获取未接入的会话列表
 * @param accessToken accessToken
 * @return 未接入的会话列表，或抛WechatException
 */
List<WaitingSession> getWaitingSessions(String accessToken)
```

- `msg()`：

```
Java
/**
 * 被动回复微信服务器文本消息
 * @return XML文本消息
 */
String respText(String openId, String content)

 /**
 * 被动回复微信服务器图片消息
 * @param openId 用户openId
 * @param mediaId 通过素材管理接口上传多媒体文件，得到的id
 * @return XML图片消息
 */
String respImage(String openId, String mediaId)
```

```
/**
 * 被动回复微信服务器语音消息
 * @param openId 用户openId
 * @param mediaId 通过素材管理接口上传多媒体文件，得到的id
 * @return XML语音消息
 */
String respVoice(String openId, String mediaId)

/**
 * 被动回复微信服务器视频消息
 * @param openId 用户openId
 * @param mediaId 通过素材管理接口上传多媒体文件，得到的id
 * @param title 标题
 * @param desc 描述
 * @return XML视频消息
 */
String respVideo(String openId, String mediaId, String title, String desc)

/**
 * 被动回复微信服务器音乐消息
 * @param openId 用户openId
 * @param mediaId 通过素材管理接口上传多媒体文件，得到的id
 * @param title 标题
 * @param desc 描述
 * @param url 音乐链接
 * @param hqUrl 高质量音乐链接，WIFI环境优先使用该链接播放音乐
 * @return XML音乐消息
 */
String respMusic(String openId, String mediaId, String title, String desc, S

/**
 * 被动回复微信服务器图文消息
 * @param openId 用户openId
 * @param articles 图片消息对象列表，长度 <= 10
 * @return XML图文消息
 */
String respNews(String openId, List<Article> articles)

/**
 * 接收微信服务器发来的XML消息，将解析xml为RecvMessage的子类，
 * 应用可根据具体解析出的RecvMessage是何种消息(msg instanceof ...)，做对应的业务处理
 * @param xml xml字符串
 * @return RecvMessage消息类，或抛WechatException
 */
RecvMessage receive(String xml)

/**
 * 向用户发送模版消息
```

```
 * @param accessToken accessToken
 * @param openId 用户openId
 * @param templateId 模版ID
 * @param link 点击链接
 * @param fields 字段列表
 * @return 消息ID，或抛WechatException
 */
Integer sendTemplate(String accessToken, String openId, String templateId, S

 /**
 * 群发消息：
 *  1. 分组群发：【订阅号与服务号认证后均可用】
 *  2. 按OpenId列表发：订阅号不可用，服务号认证后可用
 * @see me.hao0.wechat.model.message.send.SendMessageScope
 * @param accessToken accessToken
 * @param msg 消息
 * @return 消息ID，或抛WechatException
 */
Long send(String accessToken, SendMessage msg)

/**
 * 发送预览消息
 * @param accessToken accessToken
 * @param msg 预览消息
 * @return 发送成功返回true，或抛WechatException
 */
Boolean previewSend(String accessToken, SendPreviewMessage msg)

/**
 * 删除群发消息：订阅号与服务号认证后均可用：
     1、只有已经发送成功的消息才能删除
     2、删除消息是将消息的图文详情页失效，已经收到的用户，还是能在其本地看到消息卡片。
     3、删除群发消息只能删除图文消息和视频消息，其他类型的消息一经发送，无法删除。
     4、如果多次群发发送的是一个图文消息，那么删除其中一次群发，就会删除掉这个图文消息也，
 * @param accessToken acessToken
 * @param id 群发消息ID
 * @return 删除成功，或抛WechatException
 */
Boolean deleteSend(String accessToken, Long id)

/**
 * 检查群发消息状态：订阅号与服务号认证后均可用
 * @param accessToken acessToken
 * @param id 群发消息ID
 * @return 群发消息状态，或抛WechatException
 */
String getSend(String accessToken, Long id)
```

- `qr()`：

```java
/**
 * 获取临时二维码
 * @param accessToken accessToken
 * @param sceneId 业务场景ID，32位非0整型
 * @param expire 该二维码有效时间，以秒为单位。 最大不超过604800（即7天）
 * @return 临时二维码链接，或抛WechatException
 */
String getTempQrcode(String accessToken, String sceneId, Integer expire)

/**
 * 获取永久二维码
 * @param accessToken accessToken
 * @param sceneId 业务场景ID，最大值为100000（目前参数只支持1--100000）
 * @return 永久二维码链接，或抛WechatException
 */
String getPermQrcode(String accessToken, String sceneId)

/**
 * 将二维码长链接转换为端链接，生成二维码将大大提升扫码速度和成功率
 * @param accessToken accessToken
 * @param longUrl 长链接
 * @return 短链接，或抛WechatException
 */
String shortUrl(String accessToken, String longUrl)
```

- `material()`：

```java
/**
 * 获取素材总数统计
 * @param accessToken accessToken
 * @return 素材总数统计，或抛WechatException
 */
MaterialCount count(String accessToken)

/**
 * 获取素材列表
 * @param accessToken accessToken
 * @param type 素材类型
 * @param offset 从全部素材的该偏移位置开始返回，0表示从第一个素材返回
 * @param count 返回素材的数量，取值在1到20之间
 * @param <T> Material范型
 * @return 素材分页对象，或抛WechatException
 */
<T> Page<T> gets(String accessToken, MaterialType type, Integer offset, Inte
```

```
/**
 * 删除永久素材
 * @param accessToken accessToken
 * @param mediaId 永久素材mediaId
 * @return 删除成功返回true，或抛WechatException
 */
Boolean delete(String accessToken, String mediaId)

/**
 * 上传临时素材:
     图片（image）: 1M，bmp/png/jpeg/jpg/gif
     语音（voice）: 2M，播放长度不超过60s，mp3/wma/wav/amr
     视频（video）: 10MB，支持MP4格式
     缩略图（thumb）: 64KB，bmp/png/jpeg/jpg/gif
     媒体文件在后台保存时间为3天，即3天后media_id失效。
 * @param accessToken accessToken
 * @param type 文件类型
 * @param input 输入流
 * @return TempMaterial对象，或抛WechatException
 */
TempMaterial uploadTemp(String accessToken, MaterialUploadType type, String

/**
 * 下载临时素材
 * @param accessToken accessToken
 * @param mediaId mediaId
 * @return 文件二进制数据
 */
byte[] downloadTemp(String accessToken, String mediaId)

/**
 * 添加永久图文素材(其中内容中的外部图片链接会被过滤，所以需先用uploadPermNewsImage转换
 * @param accessToken accessToken
 * @param items 图文素材列表
 * @return mediaId
 */
String uploadPermNews(String accessToken, List<NewsContentItem> items)

/**
 * 添加永久图文素材(其中内容中的外部图片链接会被过滤，所以需先用uploadPermNewsImage转换
 * @param accessToken accessToken
 * @param mediaId 图文mediaId
 * @param itemIndex 对应图文素材中的第几个图文项，从0开始
 * @param newItem 新的图文项
 * @return 更新成功返回true，反之false
 */
Boolean updatePermNews(String accessToken, String mediaId, Integer itemIndex
```

```
/**
 * 上传永久图文素材内容中引用的图片
 * @param accessToken accessToken
 * @param fileName 文件名
 * @param in 文件输入流
 * @return 微信内部图片链接
 */
String uploadPermNewsImage(String accessToken, String fileName, InputStream

/**
 * 上传永久(图片，语音，缩略图)素材
     永久素材的数量是有上限的，请谨慎新增。图文消息素材和图片素材的上限为5000，其他类型为
     图片（image）：1M，bmp/png/jpeg/jpg/gif
     语音（voice）：2M，播放长度不超过60s，mp3/wma/wav/amr
     缩略图（thumb）：64KB，bmp/png/jpeg/jpg/gif
 * @param accessToken accessToken
 * @param type 文件类型
 * @param file 文件
 * @return PermMaterial对象，或抛WechatException
 */
PermMaterial uploadPerm(String accessToken, MaterialUploadType type, File fi

/**
 * 上传永久视频素材(10M大小)
 * @param accessToken accessToken
 * @param fileName 文件名
 * @param input 输入流
 * @param title 标题
 * @param desc 描述
 * @return PermMaterial对象，或抛WechatException
 */
PermMaterial uploadPermVideo(String accessToken, String fileName, InputStrea
```

- `js()`：

```Java
/**
 * 获取临时凭证
 * @param accessToken accessToken
 * @param type 凭证类型
 *             @see me.hao0.wechat.model.js.TicketType
 * @return Ticket对象，或抛WechatException
 */
Ticket getTicket(String accessToken, TicketType type)

/**
 * 获取JSSDK调用前的配置信息
 * @param jsApiTicket jsapi凭证
 * @param nonStr 随机字符串
 * @param timestamp 时间戳(s)
 * @param url 调用JSSDK的页面URL全路径(去除#后的)
 * @return Config对象
 */
Config getConfig(String jsApiTicket, String nonStr, Long timestamp, String u
```

- 组件扩展: 如果想自己扩展组件，可以继承 `Component`，调用 `register`：

```Java
public class MyComponent extends Component {
    // ...
}
MyComponent myComp = new MyComponent();
wechat.register(myComp);
```

- **AccessToken管理**:

> 由于微信服务器限制**AccessToken**请求次数，并且频繁请求**AccessToken**并不是一个明智之举，需要将获取的**AccessToken**保存下来，待过期时，再去请求新的**AccessToken**，所以以上API均提供了无accessToken版本，如:

```Java
List<String> ip();
List<String> ip(String accessToken);
```

- 实现 `AccessTokenLoader`：

```java
public interface AccessTokenLoader {

    /**
     * 获取accessToken
     * @return accessToken，若""或NULL会重新从微信获取accessToken，并触发refresh方法
     */
    String get();

    /**
     * 刷新accessToken，实现时需要保存一段时间，以免频繁从微信服务器获取
     * @param token 从微信获取的新AccessToken
     */
    void refresh(AccessToken token);
}
```

- 默认的AccessTokenLoader实现(**生产环境不推荐使用**):

```java
public class DefaultAccessTokenLoader implements AccessTokenLoader {

    private volatile AccessToken validToken;

    @Override
    public String get() {
        return (validToken == null
                || Strings.isNullOrEmpty(validToken.getAccessToken())
                || System.currentTimeMillis() > validToken.getExpiredAt()) ? nul
    }

    @Override
    public void refresh(AccessToken token) {
        validToken = token;
    }
}
```

- **Ticket管理**: 同AccessToken类似，需自己实现接口 `TicketLoader` :

```java
public interface TicketLoader {

    /**
     * 获取Ticket
     * @param type ticket类型
     *             @see me.hao0.wechat.model.js.TicketType
     * @return 有效的ticket，若返回""或null，则重新从微信请求Ticket，并触发refresh方法
     */
    String get(TicketType type);

    /**
     * 刷新Ticket
     * @param ticket 从微信获取的新Ticket
     */
    void refresh(Ticket ticket);
}
```

- 默认的TicketLoader实现(**生产环境不推荐使用**):

```java
public class DefaultTicketLoader implements TicketLoader {

    private final Map<TicketType, Ticket> tickets = new ConcurrentHashMap<>();

    @Override
    public String get(TicketType type) {
        Ticket t = tickets.get(type);
        return (t == null
                || Strings.isNullOrEmpty(t.getTicket())
                || System.currentTimeMillis() > t.getExpireAt()) ? null : t.getT
    }

    @Override
    public void refresh(Ticket ticket) {
        tickets.put(ticket.getType(), ticket);
    }
}
```

- 具体例子，可见测试用例。
- PDF文档下载。
- 历史版本
  - 1.0.0:
    - 基础功能实现。

- 1.1.0:
    - 实现代码简化，个别类访问权限修改;
    - 实现MATERIAL组件。
- 1.2.0:
    - 废弃 ~~`Wechat.newWechat`~~ 构建方法，替换为 `WechatBuilder` 方式。
    - `*Loader` 设置过期时刻。
    - 实现JSSDK组件。
- 1.3.0:
    - 引入guava。
    - API支持异步调用。
- 1.4.0:
    - 组件懒加载。
    - 改变组件访问方式，由变量到**方法**。

- 微信相关文档
    - 公众号接口权限说明
    - 接口频率限制说明
    - 接口返回码说明
    - 报警排查指引

- 你是好人
    - 倘若你钱多人傻花不完，小弟乐意效劳😊，掏出你的**微信神器**做回好人吧:

      
    - 倘若你还不够尽兴，继续掏出你的**支付宝神器**，疯狂扫吧: