

开发支持 QQ 聊天软件的 Wechaty Puppet Provider 模块

学生github账号: anaivebird

导师: 李佳芮、李卓桓

QQ puppet仓库地址:

<https://github.com/wechaty/wechaty-puppet-oicq>

项目背景

- Wechaty 社区目前已经支持微信、Whatsapp、企业微信、飞书等常见流行即时通讯工具，并且能够通过多语言 SDK（比如 Python Wechaty）进行调用。
- QQ 是国内和微信并列的两大聊天软件。
- 我们在本次 Summer 2021 的项目中，Wechaty 希望可以实现对 QQ Chatbot 的支持。
- 通过 Wechaty Puppet 的接口，可以将 QQ 进行 RPA 封装
- 目标是使 wechaty-puppet-qq 供 Wechaty 开发者方便接入 QQ 平台，使其成为 Wechaty 可以使用的社区生态模块。

方案描述

- 通过对Wechaty Puppet Provider和wechaty-puppet-mock项目等进行了解和学习，并对已有项目进行梳理
- oicq项目是使用node.js语言实现的一个qq机器人接口库，为我们本次项目实施带来了极大便利
- 实现通过 Wechaty 加载 wechaty-puppet-qq模块，并包装[oicq](#)项目，实现文本消息的收发功能
- 提供一个 examples/ding-dong-bot.ts ，完成“接收到文字消息ding时，自动回复消息dong”等功能。

已完成内容（7-8月）

- 通过与导师沟通，并观看Wechaty Puppet Provider 的 workshop 视频，熟悉Wechaty Puppet Provider的业务流程
- 完成了之前掌握不够熟练的nodejs、npm、typescript等内容进行快速的了解和学习，对时间安排做好具体的规划
- 完成了puppet代码的prototyping，参考wechaty-puppet-5g-qq 模块对接口的封装格式，并通过oicq项目实现的qq协议，实现文本消息的收发功能，并且用实际的qq账号扫码登录实践成功。
- 完成了wechaty.js.org网站博客仓库的pull request，了解了git和github的基本操作

已完成内容（9月）

- 将wechaty-puppet-oicq仓库，移到wechaty账号下
- 将包上传了npm，并与wechaty-getting-started项目连接
- 实现了ESM模式（感谢[李卓桓](#)老师的帮助）
- 配置成功通过npm自动上传新版本包（感谢[李卓桓](#)老师的帮助）
- 实现了npm link方便调试未发布包
- 修复了与wechaty-getting-started项目的兼容性问题
（ <https://github.com/wechaty/wechaty-puppet-oicq/issues/11> ）
- 实现了联系人信息的读取和解析（ contactPayload 和 contactRawPayloadParser ）
- 去除了oicq发出的无关消息

启动配置（puppet版本）

- 将oicq仓库克隆到本地
 - （ <https://github.com/wechaty/wechaty-puppet-oicq> ）
- 执行npm install安装依赖
- 在环境变量中设置QQ号
 - export WECHATY_PUPPET_OICQ_QQ=需要登录的QQ号
- 输入npm run start运行bot
- 扫描二维码登录

启动配置 (wechaty-getting-started版本)

- 将wechaty-getting-started克隆到本地
 - <https://github.com/wechaty/wechaty-getting-started>
- 执行npm install安装依赖
- 设置所需puppet类型
 - `export WECHATY_PUPPET=wechaty-puppet-oicq`
- 在环境变量中设置QQ号
 - `export WECHATY_PUPPET_OICQ_QQ=需要登录的QQ号`
- 打开调试信息
 - `export WECHATY_LOG=verbose`
- 输入npm start运行bot
- 扫描二维码登录

给机器人发消息

- 给机器人发送ding的消息
- 若机器人启动成功
- 则会回复dong的消息



ding



ding

ding



ding

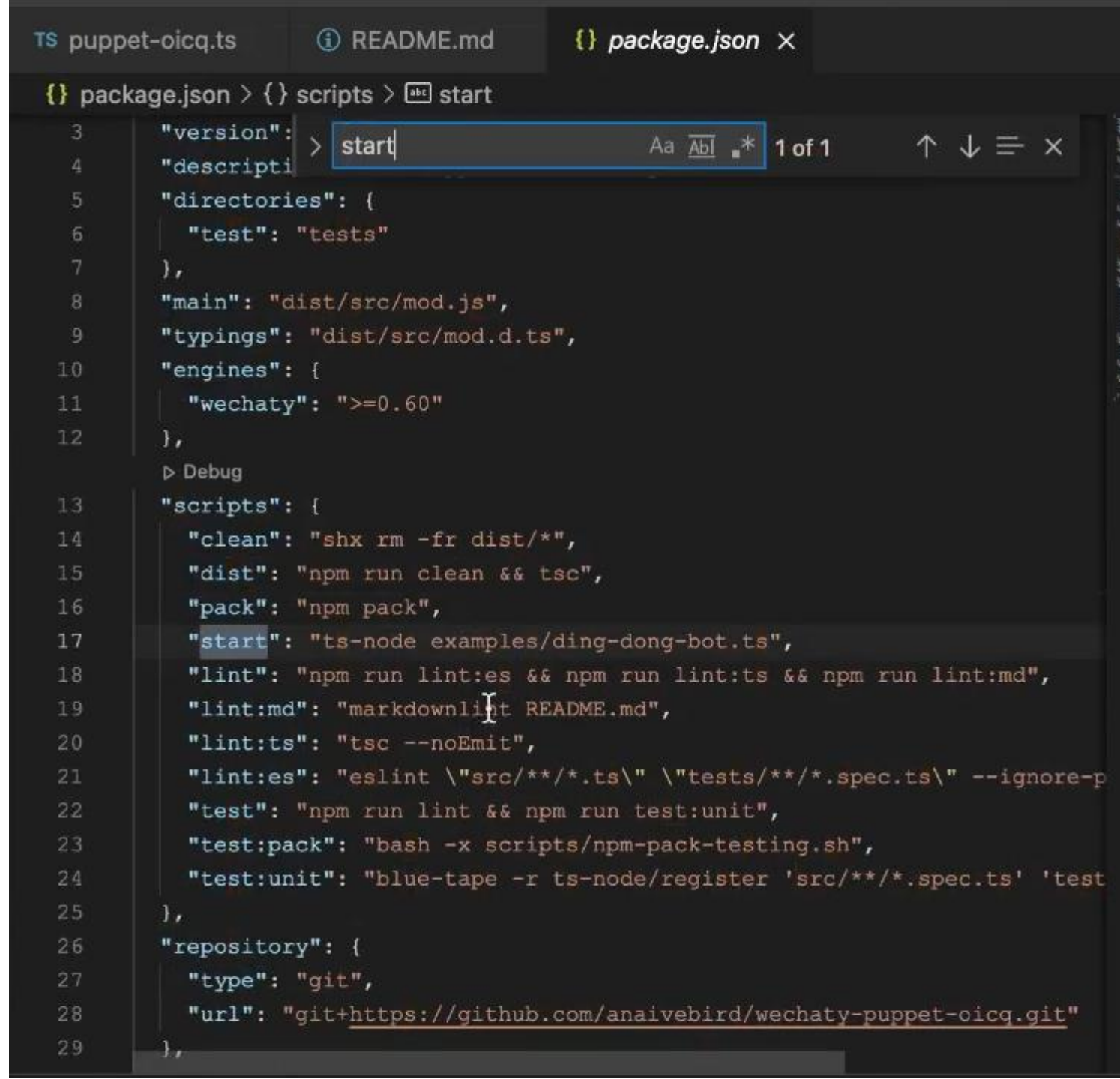


命令行消息展示

```
12:26:58 VERB StateSwitch <PuppetOICQ> on(true) <- (pending)
12:26:58 VERB PuppetOICQ login() puppet登录事件
12:26:58 VERB Puppet login(1962099319) 成功获取自身qq号
12:26:58 VERB Contact load(1962099319) init pool
12:26:58 VERB PuppetOICQ contactRawPayload(1962099319) 通过自身qq号获取自身联系人信息
12:26:58 INFO StarterBot Contact<划水狗> login 成功获取自身昵称
12:27:41 VERB Message static load(MoiqpQAACSlZx9ugYV0lvQA=)
12:27:41 VERB Message constructor(MoiqpQAACSlZx9ugYV0lvQA=) for class WechatifiedMessage
12:27:41 VERB Message ready()
12:27:41 VERB Puppet messagePayload(MoiqpQAACSlZx9ugYV0lvQA=) 通过收到的消息id获得消息信息
12:27:41 VERB Contact load(847817381) init pool 成功获得发消息的人的QQ
12:27:41 VERB PuppetOICQ contactRawPayload(847817381) 获取此人联系人信息
12:27:41 VERB Puppet selfId()
12:27:41 INFO StarterBot Message#Text[Contact<Li Dao>] ding 成功获得收到的消息内容
12:27:41 VERB Message say(dong) 成功获得此人昵称
12:27:41 VERB Puppet selfId()
12:27:41 VERB Puppet selfId()
12:27:41 VERB Message mentionList()
12:27:41 VERB Puppet selfId()
```

npm run start

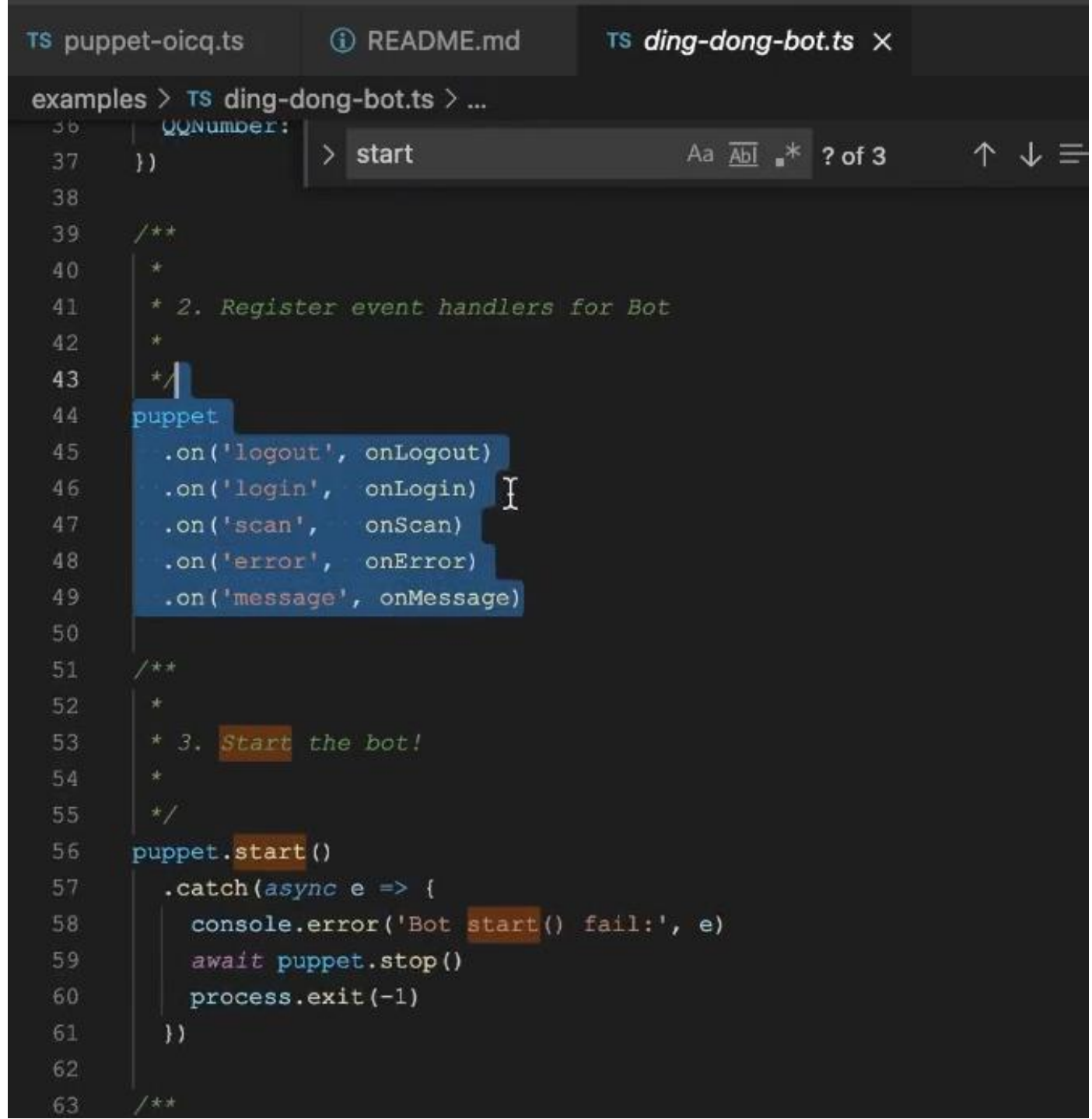
- node.js小知识
- 在package.json中
- 定义了npm run start作用
- 将会运行ding-dong-bot.ts
- 其他脚本也一样在这里定义



```
TS puppet-oicq.ts  README.md  {} package.json X
{} package.json > {} scripts > start
3  "version": "1.0.0"
4  "description": "A puppet-oicq bot"
5  "directories": {
6    "test": "tests"
7  },
8  "main": "dist/src/mod.js",
9  "typings": "dist/src/mod.d.ts",
10 "engines": {
11   "wechaty": ">=0.60"
12 },
13 "scripts": {
14   "clean": "shx rm -fr dist/*",
15   "dist": "npm run clean && tsc",
16   "pack": "npm pack",
17   "start": "ts-node examples/ding-dong-bot.ts",
18   "lint": "npm run lint:es && npm run lint:ts && npm run lint:md",
19   "lint:md": "markdownlint README.md",
20   "lint:ts": "tsc --noEmit",
21   "lint:es": "eslint \"src/**/*.ts\" \"tests/**/*.spec.ts\" --ignore-p",
22   "test": "npm run lint && npm run test:unit",
23   "test:pack": "bash -x scripts/npm-pack-testing.sh",
24   "test:unit": "blue-tape -r ts-node/register 'src/**/*.spec.ts' 'test",
25 },
26 "repository": {
27   "type": "git",
28   "url": "git+https://github.com/anaivebird/wechaty-puppet-oicq.git"
29 },
```

事件是如何绑定的

- puppet.on(事件名, 函数名)
- 用这种方法可以
- 当puppet收到事件的时候
- 对应事件处理函数被调用
- 事件处理函数绑定是
- ding-dong-bot.ts的主要任务



```
TS puppet-oicq.ts  README.md  TS ding-dong-bot.ts X
examples > TS ding-dong-bot.ts > ...
36  QQNumber:
37  })
38
39  /**
40   *
41   * 2. Register event handlers for Bot
42   *
43   */
44  puppet
45    .on('logout', onLogout)
46    .on('login', onLogin)
47    .on('scan', onScan)
48    .on('error', onError)
49    .on('message', onMessage)
50
51  /**
52   *
53   * 3. Start the bot!
54   *
55   */
56  puppet.start()
57    .catch(async e => {
58      console.error('Bot start() fail:', e)
59      await puppet.stop()
60      process.exit(-1)
61    })
62
63  /**
```

ding-dong bot代码

```
110  async function onMessage (payload: EventMessagePayload) {  
111      const msgPayload = await puppet.messagePayload(payload.messageId)  
112      console.info(JSON.stringify(msgPayload))  
113      if (msgPayload.text === 'ding') {  
114          await puppet.messageSendText(msgPayload.fromId!, 'dong')  
115      }  
116  }
```

实现ding dong逻辑

```
86  function onLogin (payload: EventLoginPayload) {  
87      console.info(`${payload.contactId} login`)  
88      puppet.messageSendText(payload.contactId, 'Wechaty login').catch(console.error)  
89  }  
90
```

登录时给自己发送Wechaty login消息

messagePayload相关代码

```
this.oicqClient.on('message', function (oicqMessage: any) {  
  puppetThis.messageStore[oicqMessage.message_id] = oicqMessage  
  puppetThis.emit('message', { messageId: oicqMessage.message_id })  
})
```

```
override async messageRawPayloadParser (rawPayload: any): Promise<MessagePayload> {  
  // OICQ qq message Payload -> Puppet message payload  
  
  const payload: MessagePayload = {  
    fromId: rawPayload.sender.user_id,  
    id: rawPayload.message_id,  
    text: rawPayload.raw_message,  
    timestamp: Date.now(),  
    toId: rawPayload.user_id,  
    type: MessageType.Text, // TODO: need to change if message type changed to image  
  }  
  return payload  
}  
  
override async messageRawPayload (oicqMessageId: string): Promise<any> {  
  return this.messageStore[oicqMessageId]  
}
```

联系人相关代码

```
this.oicqClient.on('system.online', async function () {  
  puppetThis.state.on(true)  
  
  for (const [id, friend] of this.fl.entries()) {  
    puppetThis.contactStore[id.toString()] = friend  
  }  
  await puppetThis.login(puppetThis.qq.toString())  
})
```

登录时将所有联系人存入contactStore

联系人相关代码

```
async contactRawPayload (_contactId: string): Promise<any> {  
  log.verbose('PuppetOICQ', 'contactRawPayload(%s)', _contactId)  
  return this.contactStore[_contactId]!  
}
```

根据qq号从contactStore中取出联系人信息

```
async contactRawPayloadParser (_rawPayload: any): Promise<ContactPayload> {  
  const genderStringToType: { [key: string]: ContactGender } = {  
    female: ContactGender.Female,  
    male: ContactGender.Male,  
    unknown: ContactGender.Unknown,  
  }  
  
  return {  
    avatar : 'unknown',  
    gender : genderStringToType[_rawPayload.sex]!,  
    id      : _rawPayload.user_id,  
    name    : _rawPayload.nickname,  
    phone   : ['unkown'],  
    type    : ContactType.Individual,  
  }  
}
```

将联系人格式转换成Wechaty的ContactPayload格式