Wechaty 插件那些事

Gcaufy / May, 30

个人介绍

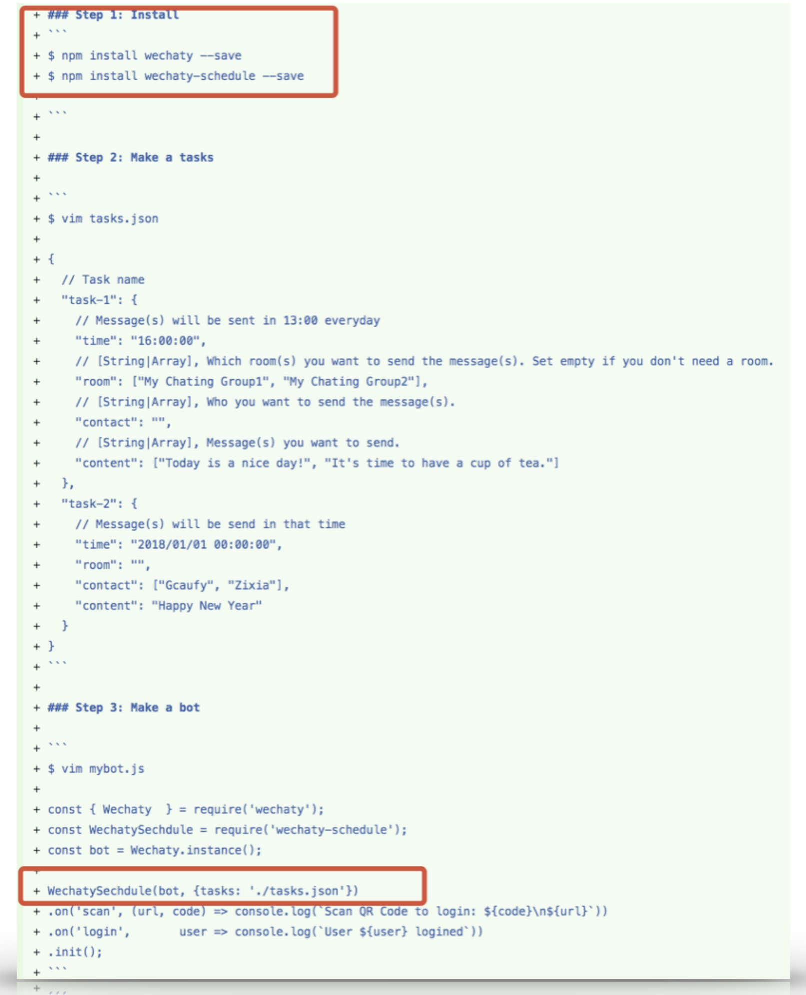
龚澄 @Gcaufy

腾讯前端工程师

WQX → Warcraft → WebForms → WebFrontend → Weapp

背景

2017 年，关键字： 分享



~~~~
+ ### Step 1: Install
+ ```
+ $ npm install wechaty --save
+ $ npm install wechaty-schedule --save
+
+ ```
+
+ ### Step 2: Make a tasks
+
+ ```
+ $ vim tasks.json
+
+ {
+   // Task name
+   "task-1": {
+     // Message(s) will be sent in 13:00 everyday
+     "time": "16:00:00",
+     // [String|Array], Which room(s) you want to send the message(s). Set empty if you don't need a room.
+     "room": ["My Chating Group1", "My Chating Group2"],
+     // [String|Array], Who you want to send the message(s).
+     "contact": "",
+     // [String|Array], Message(s) you want to send.
+     "content": ["Today is a nice day!", "It's time to have a cup of tea."]
+   },
+   "task-2": {
+     // Message(s) will be send in that time
+     "time": "2018/01/01 00:00:00",
+     "room": "",
+     "contact": ["Gcaufy", "Zixia"],
+     "content": "Happy New Year"
+   }
+ }
+ ```
+
+ ### Step 3: Make a bot
+
+ ```
+ $ vim mybot.js
+
+ const { Wechaty  } = require('wechaty');
+ const WechatySechdule = require('wechaty-schedule');
+ const bot = Wechaty.instance();
+
+ WechatySechdule(bot, {tasks: './tasks.json'})
+ .on('scan', (url, code) => console.log(`Scan QR Code to login: ${code}\n${url}`))
+ .on('login',       user => console.log(`User ${user} logined`))
+ .init();
+ ```
~~~~

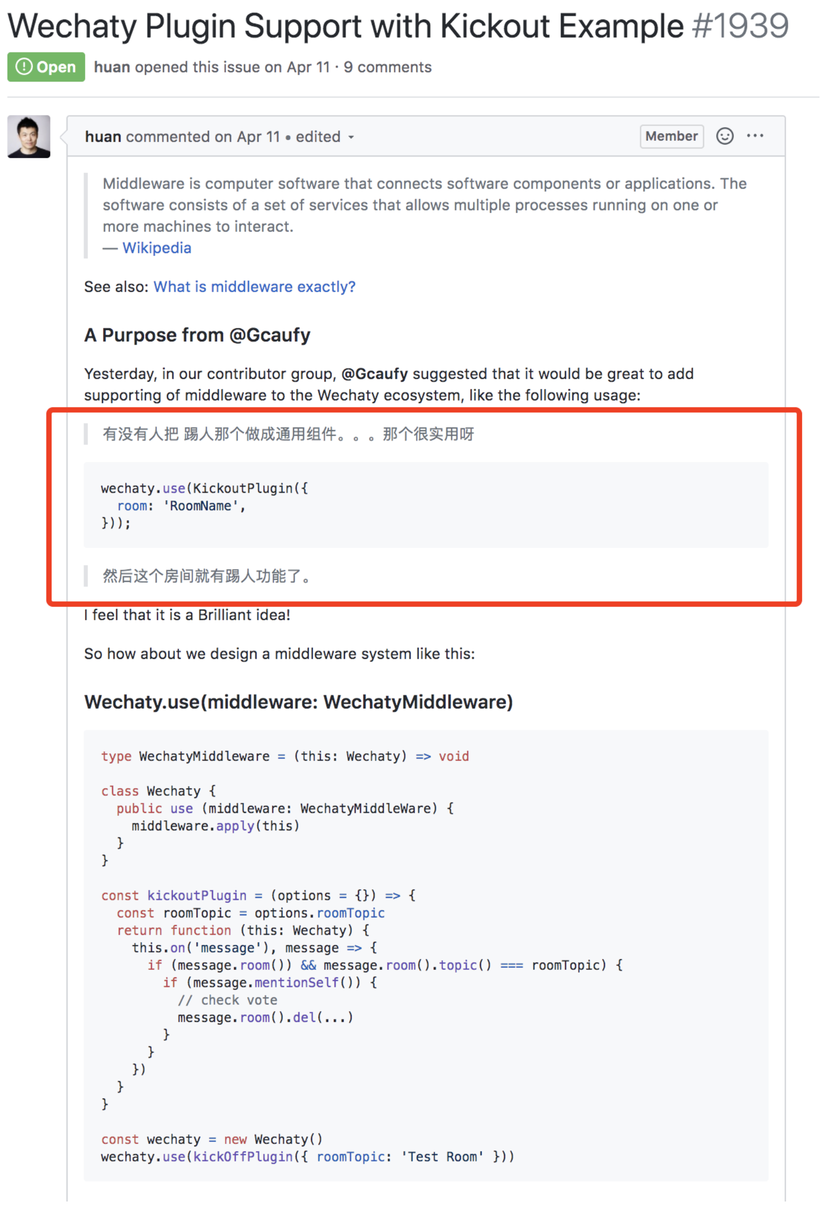
https://github.com/Gcaufy/wechaty-schedule/commit/8d95dfe877c7cb90bccfab7e0705c0016bb168ca

背景

2018 年，关键字： 北京、饭局

背景

2020年，关键字： 投票、踢人



Wechaty Plugin Support with Kickout Example #1939

Open huan opened this issue on Apr 11 · 9 comments

huan commented on Apr 11 • edited

> Middleware is computer software that connects software components or applications. The software consists of a set of services that allows multiple processes running on one or more machines to interact.
> — Wikipedia

See also: What is middleware exactly?

A Purpose from @Gcaufy

Yesterday, in our contributor group, **@Gcaufy** suggested that it would be great to add supporting of middleware to the Wechaty ecosystem, like the following usage:

> 有没有人把 踢人那个做成通用组件。。。那个很实用呀

```
wechaty.use(KickoutPlugin({
  room: 'RoomName',
}));
```

> 然后这个房间就有踢人功能了。

I feel that it is a Brilliant idea!

So how about we design a middleware system like this:

Wechaty.use(middleware: WechatyMiddleware)

```
type WechatyMiddleware = (this: Wechaty) => void

class Wechaty {
  public use (middleware: WechatyMiddleWare) {
    middleware.apply(this)
  }
}

const kickoutPlugin = (options = {}) => {
  const roomTopic = options.roomTopic
  return function (this: Wechaty) {
    this.on('message'), message => {
      if (message.room()) && message.room().topic() === roomTopic) {
        if (message.mentionSelf()) {
          // check vote
          message.room().del(...)
        }
      }
    })
  }
}

const wechaty = new Wechaty()
wechaty.use(kickOffPlugin({ roomTopic: 'Test Room' }))
```

https://github.com/wechaty/wechaty/is
1939

开发

定义名称：	Plugin or MiddleWare
定义规范:	Funtional Style or Class Style
定义规则：	Mutiple times install / Keep or Forget
定义作用域：	Use scope

https://github.com/wechaty/wechaty/pull
46

为什么需要插件

对于开发者：

- 逻辑解藕
- 代码复用

对于开源项目：

- 构建开发者生态

https://github.com/wechaty/wechaty/p
46

插件开发最佳实践

1. 单一职责原则

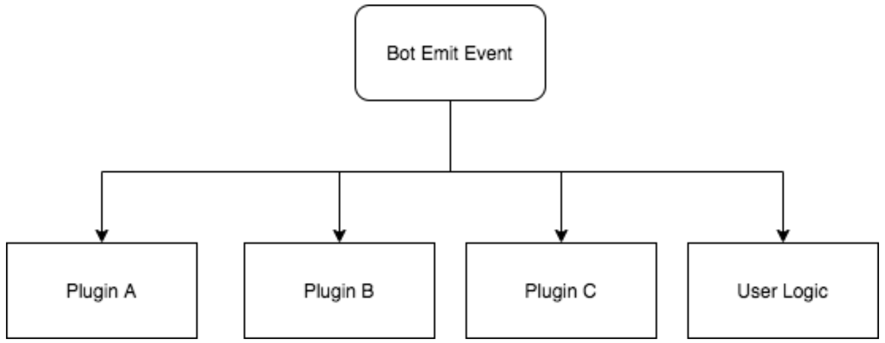
2. 组合原则

参考：

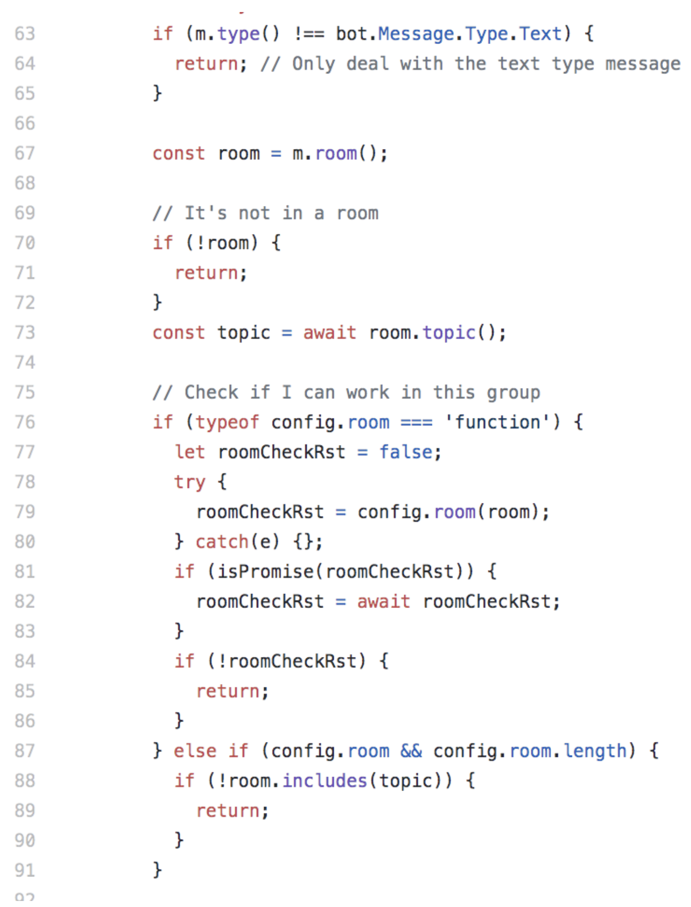
* Unix Philosophy

* KISS

插件的不足



* 平等关系　* 决策能力　* 插件间沟通能力



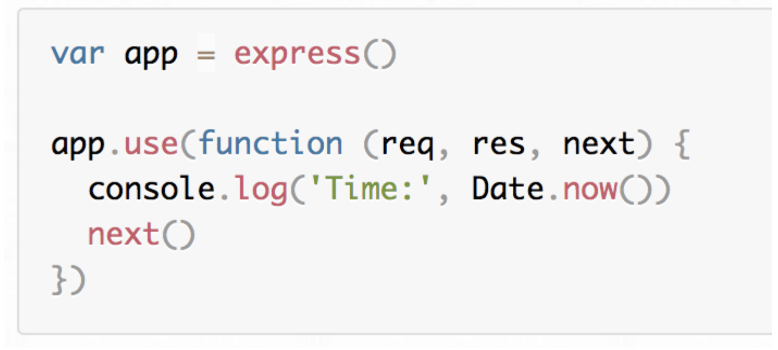
```
      if (m.type() !== bot.Message.Type.Text) {
        return; // Only deal with the text type message
      }

      const room = m.room();

      // It's not in a room
      if (!room) {
        return;
      }
      const topic = await room.topic();

      // Check if I can work in this group
      if (typeof config.room === 'function') {
        let roomCheckRst = false;
        try {
          roomCheckRst = config.room(room);
        } catch(e) {};
        if (isPromise(roomCheckRst)) {
          roomCheckRst = await roomCheckRst;
        }
        if (!roomCheckRst) {
          return;
        }
      } else if (config.room && config.room.length) {
        if (!room.includes(topic)) {
          return;
        }
      }
```

中间件



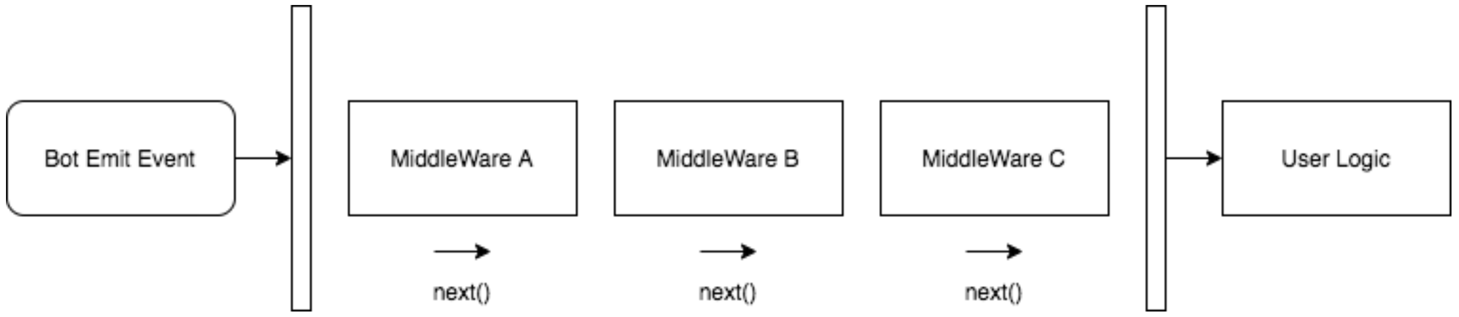
```
var app = express()

app.use(function (req, res, next) {
  console.log('Time:', Date.now())
  next()
})
```

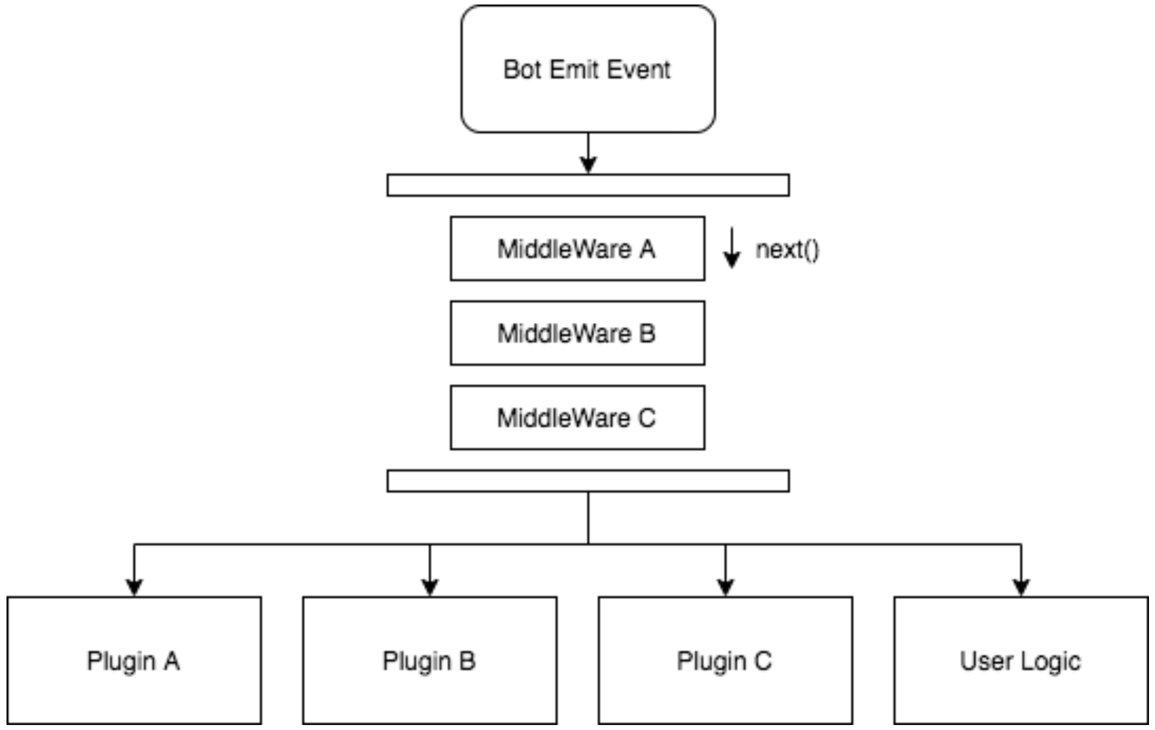
Middleware functions can perform the following tasks:

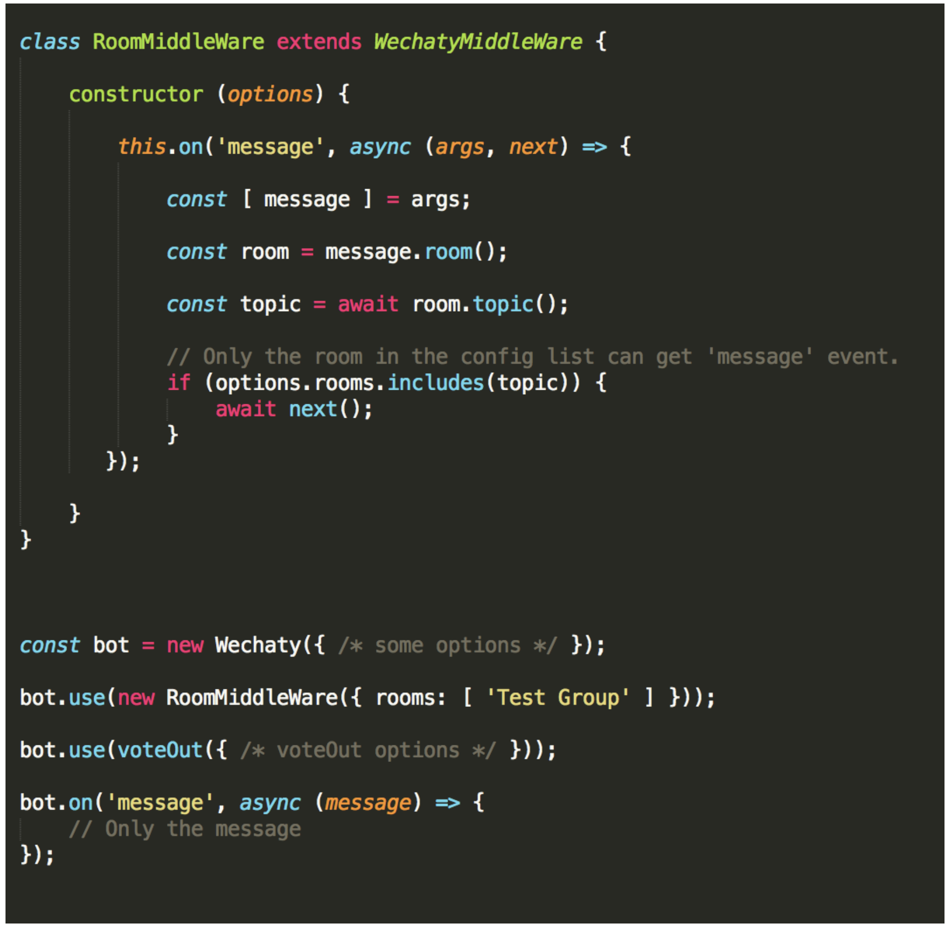
- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.
- Call the next middleware in the stack.

If the current middleware function does not end the request-response cycle, it must call `next()` to pass control to the next middleware function. Otherwise, the request will be left hanging.



结合中间件





```
class RoomMiddleWare extends WechatyMiddleWare {

    constructor (options) {

        this.on('message', async (args, next) => {

            const [ message ] = args;

            const room = message.room();

            const topic = await room.topic();

            // Only the room in the config list can get 'message' event.
            if (options.rooms.includes(topic)) {
                await next();
            }
        });

    }
}



const bot = new Wechaty({ /* some options */ });

bot.use(new RoomMiddleWare({ rooms: [ 'Test Group' ] }));

bot.use(voteOut({ /* voteOut options */ }));

bot.on('message', async (message) => {
    // Only the message
});
```

Thanks