sdk安装

外发版本通过下面方式安装

```
pip install qq-bot
```

更新包的话需要添加 `--upgrade` 注：需要python3.7+

sdk使用

需要使用的地方import SDK

```
import qqbot
```

外发版本 更新频率 weeks

示例API：获取机器人数据

```python
class UserAPI(APIBase):
    """用户相关接口"""

    def me(self) -> User:
        """
        :return:使用当前用户信息填充的 User 对象
        """
        url = get_url(APIConstant.userMeURI, self.is_sandbox)
        response = self.http.get(url)
        return json.loads(response.content, object_hook=User)
```

Step 1: 构造Token对象

token = Token(${appid},${token})

Step 2: 构造一个GuildAPI

api = qqbot. UserAPI(token, IS_SANDBOX)

Step 3: 调用api方法

robot = api.me()

Step 4: 使用返回的数据对象

print(robot.username)

```python
class UserAPITestCase(unittest.TestCase):
    api = qqbot.UserAPI(token, IS_SANDBOX)

    def test_me(self):
        user = self.api.me()
        self.assertEqual(ROBOT_NAME, user.username)
```

5 Python-SDK 异步使用

SDK异步模块主要通过**websocket** 实现与机器人后台的通信连接，支持**超时重连、并发及客户端水平扩展**能力

使用步骤

Step 1: 构造Token对象

token = Token(${appid},${token})

Step 2: 构造事件处理对象

① **通过qqbot.HandlerType定义监听类型**

② 设置回调处理方法

注：每种事件会有出现不同的数据对象回调，具体可参考右图

```
def _message_handler(event, message: qqbot.Message):
        # 回调处理


qqbot_handler = qqbot.Handler(
    qqbot.HandlerType.AT_MESSAGE_EVENT_HANDLER, _message_handler
)
```

Step 3: 开始监听事件（可支持多个事件）

qqbot.listen_events(t_token, False, qqbot_handler)

注：当前支持事件及回调数据对象为：

```
class HandlerType(Enum):
    PLAIN_EVENT_HANDLER = 0 #透传事件
    GUILD_EVENT_HANDLER = 1 #频道事件
    GUILD_MEMBER_EVENT_HANDLER = 2 #频道成员事件
    CHANNEL_EVENT_HANDLER = 3 #子频道事件
    MESSAGE_EVENT_HANDLER = 4 #消息事件
    AT_MESSAGE_EVENT_HANDLER = 5 #At消息事件
    # DIRECT_MESSAGE_EVENT_HANDLER = 6 #私信消息事件
    # AUDIO_EVENT_HANDLER = 7 #音频事件
```

事件回调函数的参数 1 为事件名称，参数 2 返回具体的数据对象。

```
#透传事件（无具体的数据对象，根据后台返回Json对象）
def _plain_handler(event, data):
#频道事件
def _guild_handler(event, guild:Guild):
#频道成员事件
def _guild_member_handler(event, guild_member: GuildMember):
#子频道事件
def _channel_handler(event, channel: Channel):
#消息事件 #At消息事件
def _message_handler(event, message: Message):
```

日志打印

基于自带的 logging 模块封装的日志模块，提供了日志写入以及美化了打印格式，并支持通过设置 `QQBOT_LOG_LEVEL` 环境变量来调整日志打印级别（默认打印级别为 `INFO`）。

使用方法

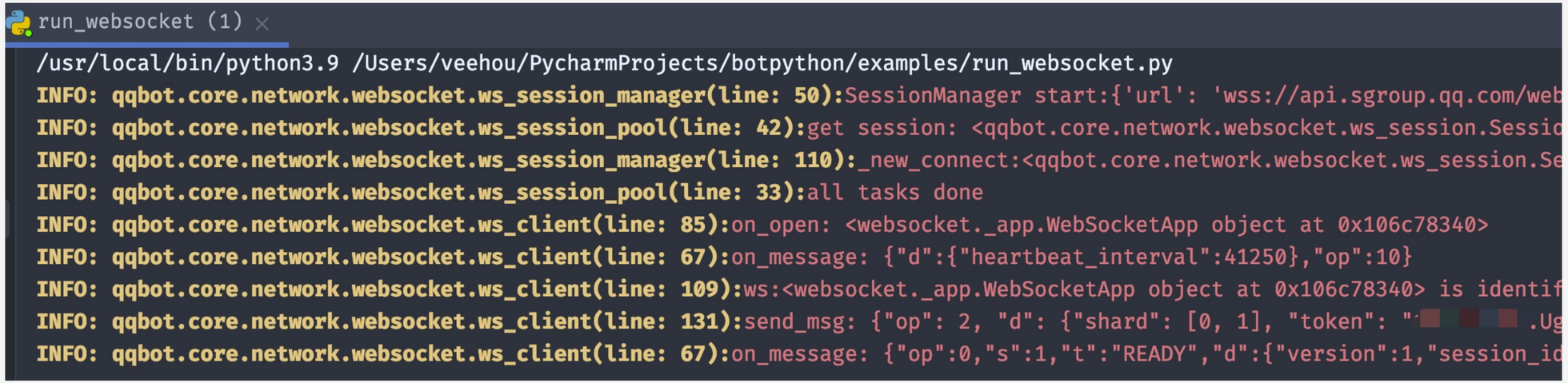
引用模块，并获取 `logger` 实例：

```
from core.util import logging

logger = logging.getLogger(__name__)
```

然后就可以愉快地使用 logger 进行打印。例如：

```
logger.info("hello world!")
```



7 Python-SDK 完整例子-平台注册、配置

获取appid、token

配置语料

7 Python-SDK 完整例子-代码编写

比如下面这个例子：需要监听机器人被@后消息并进行相应的回复。

- 先初始化需要用的 `token` 对象
- 通过 `qqbot.listen_events` 注册需要监听的事件
- 通过 `qqbot.HandlerType` 定义需要监听的事件（部分事件可能需要权限申请）

```
t_token = qqbot.Token(test_config["token"]["appid"], test_config["token"]["token"])
# 注册事件类型和回调，可以注册多个
qqbot_handler = qqbot.Handler(qqbot.HandlerType.AT_MESSAGE_EVENT_HANDLER, _message_handler)
qqbot.listen_events(t_token, False, qqbot_handler)
```

- 最后定义注册事件回调执行函数,如 `_message_handler` 。

```
def _message_handler(event, message: Message):
    msg_api = qqbot.MessageAPI(t_token, False)
    # 打印返回信息
    qqbot.logger.info("event %s" % event + ",receive message %s" % message.content)
    # 构造消息发送请求数据对象
    send = qqbot.MessageSendRequest("<@%s>谢谢你，加油" % message.author.id, message.id)
    # 通过api发送回复消息
    msg_api.post_message(message.channel_id, send)
```

7 Python-SDK 完整例子-测试及上架

Github：

sdk开发

环境配置

```
pip3 install -r requirements.txt   # 安装依赖的pip包

pre-commit install                 # 安装格式化代码的钩子

python3 setup.py sdist bdist_wheel # 打包SDK
```

单元测试

代码库提供API接口测试和websocket的使用Demo，位于`tests`目录中, 如果需要自己运行，可以在根目录添加.test.yaml文件后添加自己的测试参数启动测试

```yaml
# test yaml 用于设置test相关的参数，开源版本需要去掉参数
token:
  appid: "xxx"
  token: "xxxxx"
test_params:
  guild_id: "xx"
  guild_owner_id: "xx"
  guild_owner_name: "xx"
  guild_test_member_id: "xx"
  guild_test_role_id: "xx"
  channel_id: "xx"
  channel_name: "xx"
  robot_name: "xxx"
  is_sandbox: False
```

SDK当前处于基础建设阶段，欢迎大家踊跃提issue，同时别忘记star哦！