2019年度B4新人研修課題 SlackBot報告書 (佐藤)

2019/4/15

佐藤 宏樹

1 概要

本資料は，2019 年度 B4 新人研修課題にて作成した SlackBot プログラムの作成報告書である．SlackBot とは，チャットツールである Slack [1] において，特定の作業を自動で行うプログラムのことである．本資料では，課題内容，理解できなかった部分，作成できなかった機能，および自主的に作成した機能について述べる．

2 課題内容

Ruby を用いて SlackBot プログラムを作成する．課題として，以下の 2 つの機能を実装する．

(1) 任意の文字列を発言するプログラムの作成
SlackBot プログラムの作成には，Ruby を用いる．使用する Ruby のバージョンは 2.5.1 である．また，以下の 2 つの機能をもつ SlackBot クラスを用いる．
（A）Slack の Incoming Webhook [2] を利用し，発言する機能
（B）Slack の Outgoing Webhook [3] によって発言を取得した場合，反応する機能
上記 2 つの機能を持つ SlackBot クラスを継承したクラスを新たに作成し，以下の機能を実装する．
（A）受信した発言の中に “「○○○」と言って” という文字列があった場合は，“○○○” と発言する．

(2) SlackBot プログラムへの機能追加
SlackBot プログラムへ機能を追加する．Slack 以外の Web サービスの API や Webhook を利用した機能を追加する．

3 理解できなかった部分

本課題において理解できなかった部分は，SlackBot クラスにおける `post_message` メソッドの挙動である．プログラムを呼ばれた際に返答する SlackBot クラスのメソッド `naive_respond` は，HTTP レスポンスを返り値としているが，`post_message` メソッドにおいては `http.post` メソッドによって POST した返り値が `res` に代入され，この `res` が `post_message` メソッドの返り値となっている．プログラム自体は正しく動作するのだが，これでは 2 回 POST されてしまうのではないかと疑問に思った．6 章に，付録として該当するメソッド部分を載せる．

4　作成できなかった機能

課題内において作成できなかった機能はない．本プログラムに対して追加できると考えたが作成できなかった機能を以下に示す．

(1) 設定した Outgoing Webhook 以外からの POST を拒否する機能
現状の実装では，誰でも Slack になりすまして POST をおこない，Bot を動作させることが可能となってしまう．この問題について，自身が設定した Outgoing Webhook のみで動作するようにする必要がある．

(2) プログラム上の同じ機能を一度に複数使用する機能
本プログラムでは，1 つの投稿メッセージに対して，プログラム上の機能が 1 度しか起動しないようになっている (ただし，投稿メッセージがどの機能の起動条件も満たさない場合は投稿に反応しない)．投稿メッセージの解析方法や，各機能の起動条件を改良することで，1 つのメッセージで同じ機能を反復して使えるのではないかと考えられる．

5　自主的に作成した機能

課題外において自主的に作成した機能はないが，機能追加の課題において作成した機能を以下に示す．

(1) 入力された郵便番号に対応する県名と市区町村名を答える機能
取得した投稿が “@Hirobot (7 桁の数字)” という形式であった場合，本プログラムは入力された数字を郵便番号とみなし，対応する県名と市町村名を投稿する．詳細は別資料「2019 年度 B4 新人研修課題 SlackBot 仕様書 (佐藤)」に記載する．

6　付録

6.1　post_message メソッド

```
def post_message(string, options = {})
    payload = options.merge({text: string})
    uri = URI.parse(@incoming_webhook)
    res = nil
    json = payload.to_json
    request = "payload=" + json

    Net::HTTP.start(uri.host, uri.port, use_ssl: true) do |http|
      http.verify_mode = OpenSSL::SSL::VERIFY_NONE
```

```
    res = http.post(uri.request_uri, request)
  end

  return res
end
```

6.2 naive_respond メソッド

```
def naive_respond(params, options = {})
  return nil if params[:user_name] == "slackbot" || params[:user_id] == "USLACKBOT"
  user_name = params[:user_name] ? "@#{params[:user_name]}" : ""
  return {text: "<#{user_name}> Hi!"}.merge(options).to_json
end
```

参考文献

[1] Slack: Where work happens, Slack (online), available from ⟨https://slack.com/⟩ (accessed 2019-04-15).

[2] Slack: Incoming Webhooks, Slack (online), available from ⟨https://api.slack.com/incoming-webhooks⟩ (accessed 2019-04-15).

[3] Slack: Outgoing Webhooks, Slack (online), available from ⟨https://api.slack.com/custom-integrations/outgoing-webhooks⟩ (accessed 2019-04-15).