SlackBot プログラム作成の報告書 (修正版)

2018/5/17

高家 雄太郎

1 概要

本資料は，資料＜ No.351-03 ＞の修正版である．修正箇所は以下のとおりである．

(1) 本資料の第 2 章において，“発言” の説明に “発言” という単語を用いるのは適切でないため，“発言” という単語を使用しない説明に修正した．
(2) 本資料の第 2 章において，Outgoing WebHooks と Incoming WebHooks を踏まえた SlackBot の処理を分かりやすくするために，本プログラムにおける SlackBot の処理の流れの図と説明を追加した．
(3) ソースコードを添付する際，ソースコードの一部であるメソッドのみの添付であり，情報が不足していた．このため，付録 A と付録 B を追加し，第 3 章にこの説明を追加した．

2 課題内容

Ruby を用いて SlackBot プログラムを作成する．SlackBot プログラムとはチャットツールである Slack[1] に発言する機能と，Slack 上のユーザの発言を契機に何らかの処理を行う機能をもつプログラムである．発言とは Slack で投稿することを指す．
本プログラムは，図 1 のような処理の流れである．図 1 はユーザの発言から SlackBot が処理の結果を発言するまでの処理の流れを表している．処理の流れを以下に述べる．

(1) ユーザが Slack で発言する．
(2) Slack はユーザの発言を Outgoing WebHooks を利用してサーバに POST する．
(3) サーバは，受け取った発言内容に応じた処理を行う．
(4) サーバは Incoming WebHooks を利用して発言する．
(5) Slack に発言が表示される．

Outgoing WebHooks とは，特定の発言を受信した際，指定した URL に発言内容やユーザネームを含むデータを POST する機能である．Incoming WebHooks とは，外部サービスから Slack に発言する機能である．
与えられた課題は以下の 2 つである．

(課題 1) 任意の文字列を発言するプログラムの作成
受信した発言の中に “「(任意の文字列)」と言って” という文字列を含む場合は，“(任意の文字列)” と発言するプログラムを作成する．

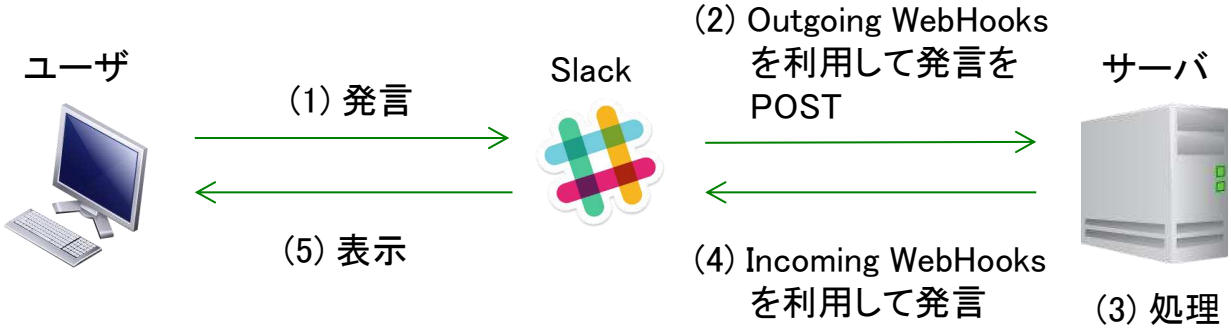

図 1　SlackBot の処理の流れ

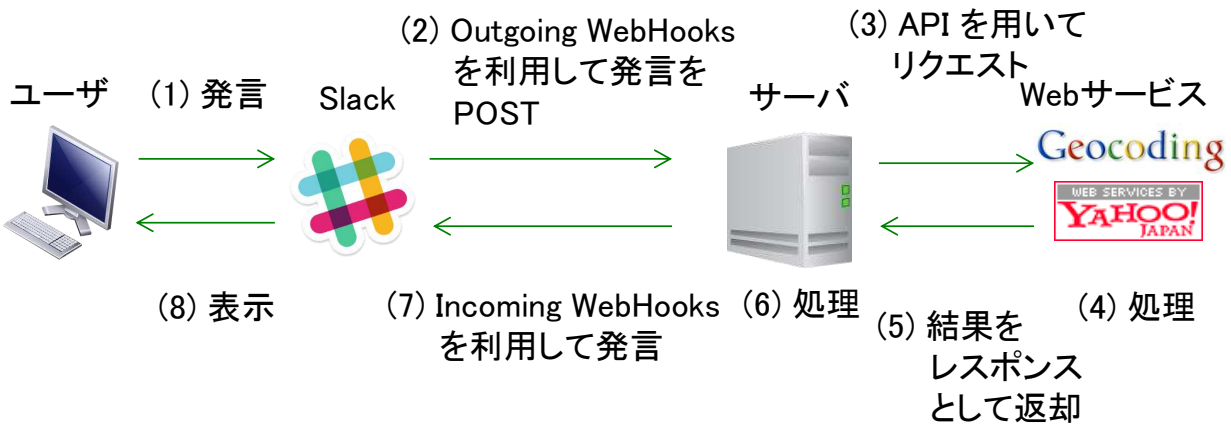

図 2　API を用いた SlackBot の処理の流れ

(課題 2)　SlackBot プログラムへの機能追加

Slack 以外の Web サービスの API を利用した機能を追加する．API とはプログラムからソフトウェアを操作するためのインタフェースである．サーバは受けとったユーザの発言の処理に API を利用する．本プログラムにおいて API を利用する場合の処理の流れは図 2 のようになる．図 2 はユーザの発言から SlackBot が処理の結果を発言するまでの処理の流れを表している．処理の流れを以下に述べる．

(1)　ユーザが Slack で発言する．
(2)　Slack はユーザの発言を Outgoing WebHooks を利用してサーバに POST する．
(3)　サーバは API を用いて Web サービスにリクエストを送信する．
(4)　Web サービスがリクエストに応じた処理を行う．
(5)　Web サービスは結果をレスポンスとして返却する．
(6)　サーバは API のレスポンスを用いて，ユーザの発言内容に応じた処理を行う．
(7)　サーバは Incoming WebHooks を利用して発言する．
(8)　Slack に発言が表示される．

本課題における Ruby のバージョンは，2.5.1 である．

3　理解できなかった部分

理解できなかった部分を以下に示す.

(1) POST の方法の違い

本課題では，サンプルプログラムが与えられる．与えられたサンプルプログラムの一部のソースコードのを付録 A，付録 B に示す．`SlackBot.rb` は以下の 2 つの機能をもつ SlackBot クラスを定義するファイルである.

（A）Incoming WebHooks を利用し発言する機能（14-27 行目）

（B）Outgoing WebHooks によって発言を取得した場合に反応する機能（29-34 行目）

`MySlackBot.rb` は SlackBot クラスを継承した MySlackBot クラスの定義と，ある URL に対応するリクエストがどの HTTP メソッドかに応じて処理内容を記述するファイルである．このサンプルプログラムで使用する以下の 2 つの方法での POST の違いが理解できなかった.

（A）Sinatra の戻り値を用いる方法

本プログラムは Web アプリケーションフレームワークである Sinatra を用いており，ルーティングブロックの戻り値は HTTP レスポンスボディとして返される．ルーティングブロックとは HTTP レスポンスに対する処理の定義が行われている部分を指す．サンプルプログラムでは，`MySlackBot.rb` の 14-16 行目，18-21 行目がルーティングブロックである.

（B）http.post メソッドを用いる方法

`SlackBot.rb` の 14-24 行目の `post_message` メソッドは http.post メソッドを使用してリクエストを返すメソッドである.

4　課題 2 で追加した機能

自主的に作成した機能を以下に示す.

(1) 直近 60 分後の降水強度予測を発言する機能

取得した発言に “雨の状況” という文字列を含む場合，受信時刻から 10 分刻みに直近 60 分後までの降水強度予測とそれに対応する雨の強さを発言する．また，観測地点を指定できる．観測地点の指定がない場合は岡山大学を観測地点とする．詳しくは SlackBot プログラムの仕様書に記載する．仕様書は< No.351-04 >として提出している.

5　作成できなかった機能

作成できなかった機能を以下に示す.

(1) 設定した Outgoing WebHooks 以外からの POST を拒否する機能

(2) ランドマークの住所を表示する機能

本プログラムでは，第 4 章 (1) の機能において，観測地点を住所またはランドマーク名によって指定できる．しかし，指定したランドマークと同名のランドマークが複数存在する場合の動作を保証していない．そこで，ランドマークの住所を表示することで，出力結果が意図した観測地点における結果であるか否かを確認できる．

付録 A　SlackBot.rb

```
require 'json'
require 'uri'
require 'yaml'
require 'net/https'

class SlackBot
  def initialize(settings_file_path = "settings.yml")
    config = YAML.load_file(settings_file_path)
              if File.exist?(settings_file_path)
    # This code assumes to set incoming webhook url
       as evironment variable in Heroku
    # SlackBot uses settings.yml as config when it serves on local
    @incoming_webhook = ENV['INCOMING_WEBHOOK_URL']
                         || config["incoming_webhook_url"]
  end

  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

  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
end
```

付録 B　MySlackBot.rb

```
$LOAD_PATH.unshift(File.dirname(__FILE__))

require 'sinatra'
require 'SlackBot'
```

```
 5
 6 class MySlackBot < SlackBot
 7   # cool code goes here
 8 end
 9
10 slackbot = MySlackBot.new
11
12 set :environment, :production
13
14 get '/' do
15   "SlackBot Server"
16 end
17
18 post '/slack' do
19   content_type :json
20   slackbot.naive_respond(params, username: "Bot")
21 end
```

参考文献

[1] Slack: 人々の働き方を変える、次世代のビジネスコラボレーションツール — Slack, Slack（オンライン），入手先 ⟨https://slack.com/⟩（参照 2018-05-17）.