SlackBot プログラムの仕様書 (修正版)

2018/5/17

高家 雄太郎

1 はじめに

本資料は，資料＜ No.351-04 ＞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 3 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 3 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．

2 対象とする利用者

本プログラムでは以下のアカウントを所有する利用者を対象としている．

(1) Slack アカウント
(2) Yahoo! JAPAN[1] アカウント
Yahoo! JAPAN アカウントは本プログラムで使用する Client ID の取得に必要である．

3 機能

本プログラムは，“@AmeBot” という文字列から始まる発言をリクエストメッセージとして受け取り，それに対して SlackBot が発言する．SlackBot とは，チャットツールである Slack[2] に発言する，また Slack 上のユーザの発言を契機に何らかの処理を行うプログラムである．発言とは Slack で投稿することを指す．また “@AmeBot” 以降の内容によって SlackBot の発言内容が決定される. 本プログラムでは，図 1 のような処理の流れになる．図 1 はユーザの発言から SlackBot が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) ユーザは発言する．
(2) Slack はユーザの発言を Outgoing WebHooks を利用してサーバに POST する．
Outgoing WebHooks とは，特定の発言がされると指定した 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．
(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．

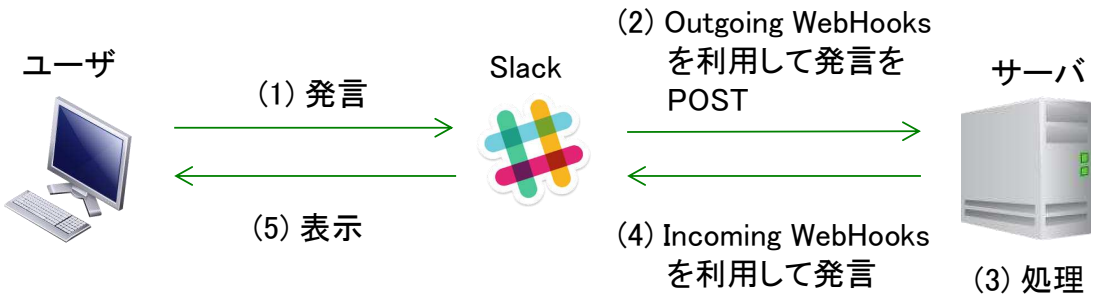

図 1　SlackBot プログラムの処理の流れ

Incoming WebHooks とは，外部サービスから Slack に発言する機能である．

(5) Slack に発言が表示される．

以下に本プログラムがもつ 2 つの機能について述べる．

(機能 1)　指定した文字列を発言する機能

リクエストメッセージに “「(任意の文字列)」と言って” という文字列を含む場合，SlackBot は “(任意の文字列)” と発言する．また，リクエストメッセージに “「(任意の文字列)」と言って” という文字列が複数存在する場合は，SlackBot は発言内容が最長となる文字列を発言する．たとえば，“@AmeBot 「おはよう」と言って，「こんにちわ」と言って” というリクエストメッセージを受け取った場合，“おはよう」と言って，「こんにちわ” と発言する．

(機能 2)　直近 60 分後の降水強度予測を発言する機能

リクエストメッセージに “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対応する雨の強さを発言する．表 1 に降水強度予測に対応する雨の強さを示す．たとえば，“@AmeBot 雨の状況” という発言に対する SlackBot の発言は，図 2 のようになる．なお，降水強度とは瞬間的な雨の強さを 1 時間あたりに換算した雨量のことであり，単位は `mm/h`(ミリメートル毎時) である．また，リクエストメッセージが “@AmeBot (住所またはランドマーク名) の雨の状況” という形式の場合，(住所またはランドマーク名) を観測地点とする．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．降水強度予測は Yahoo! デベロッパーネットワーク 気象情報 API から，また観測地点検索は Geocoding API[3] から情報を取得する．API を用いた SlackBot の処理の流れは図 3 のようになる．図 3 はユーザの発言から SlackBot が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)　ユーザは発言する．
(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．

表 1　降水強度とそれに対する雨の強さ

降水強度 (mm/h)	雨の強さ
0	降ってない
0 以上 10 未満	雨
10 以上 20 未満	やや強い雨
20 以上 30 未満	強い雨
30 以上 50 未満	激しい雨
50 以上 80 未満	非常に激しい雨
80 以上	猛烈な雨

図 2　“雨の状況” という発言に対する SlackBot の発言例

(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．

本プログラムでは 2 つの Web サービスにを利用しているため，(3)∼(6) の処理を 2 度行う．

なお，上記の (機能 1),(機能 2) のどちらにも当てはまらない発言を受信した場合，“Hi! Usage:“「○○」と言って”,“雨の状況”” という文字列を発言する．また (機能 1),(機能 2) の条件がどちらも満たす場合，(機能 1) が優先される．

4　動作環境

本プログラムは Heroku[4] 上で動作させる．Heroku とは、PaaS（Platform as a Service）と呼ばれる形態のサービスで、アプリケーションを実行するためのプラットフォームである．なお，Heroku はフリープランで利用することを想定している．表 2 に，本プログラムを動作させる Heroku の環境を示す．また，bundler 以外の Gem は Gemfile と Gemfile.lock に記述されている依存関係を用いてインストールされる．Gem とは Ruby 言語用のライブラリであり，Bundler とは Gem 専用のパッケージ管理システムである．

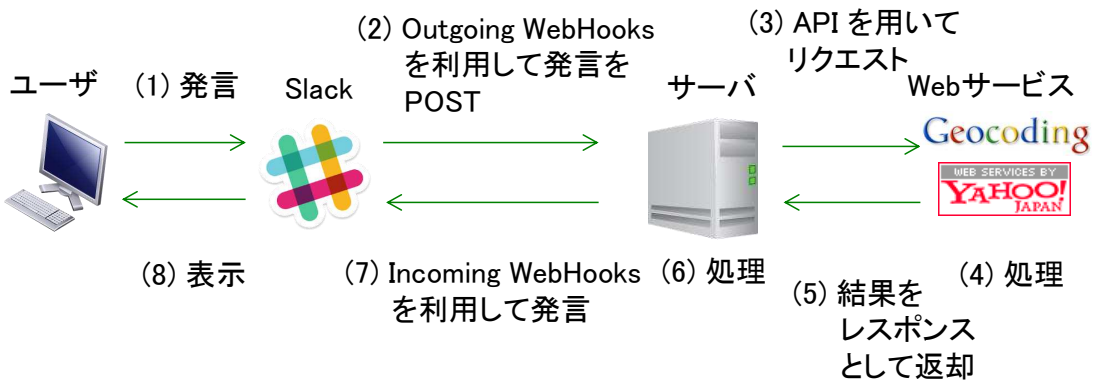

図 3　API を用いた SlackBot プログラムの処理の流れ

表 2　動作環境

項目	内容
OS	Ubuntu 16.04.4 LTS
CPU	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
メモリ	512MB
Ruby	ruby 2.5.1p57
Ruby Gem	Bundler version 1.15.2
	tilt 2.0.8
	mustermann 1.0.2
	rack 2.0.4
	rack-protection 2.0.1
	sinatra 2.0.1

5　動作確認済み環境

表 2 に示す Heroku の環境で動作確認済みである.

6　環境構築

6.1　概要

本プログラムを実行するために必要な環境構築の手順を以下に示す.

(1) Heroku アカウントの作成と設定
(2) Slack アカウントの Incoming WebHooks の設定
(3) Slack アカウントの Outgoing WebHooks の設定

(4) Yahoo! JAPAN デベロッパーネットワーク [5] Web API の Client ID の取得

次節で具体的な手順について述べる.

6.2 具体的な手順

6.2.1 Heroku アカウントの作成と設定

Heroku のアカウント作成と設定を行う.

(1) 以下の URL より Heroku にアクセスし,「無料で新規登録」から新しいアカウントを登録する.
`https://www.heroku.com/`
(2) 登録したアカウントでログインし,「Getting Started on Heroku」の使用する言語として Ruby を選択する.
(3)「I'm ready to start」をクリックし「Download the Heroku CLI for...」から OS を選択する.
(4)「Download the Heroku CLI for...」をクリックして表示されるコマンドを用いて CLI(Command Line Interface) をダウンロードする.
(5) 以下のコマンドを実行し，Heroku にログインする.

```
$ heroku login
```

(6) 作成したアプリケーションのディレクトリに移動して以下のコマンドを実行し，Heroku 上にアプリケーションを生成する.

```
$ cd ~/<myapp>
$ heroku create <myapp_name>
```

ここで`<myapp>`は作成したアプリケーションのディレクトリであり，`<myapp_name>`は任意のアプリケーション名である.

6.2.2 Slack アカウントの Incoming WebHooks の設定

Slack が提供している Incoming WebHooks の設定を行う.

(1) 以下の URL にアクセスし，本プログラム使用者の Slack アカウントにログインする.
`https://slack.com/intl/ja-jp`
(2) ページ左上のチーム名をクリックし,「Slack をカスタマイズ」をクリックする.
(3) ページ左上の Menu をクリックし「App 管理」をクリックする.
(4) ページ左の「カスタムインテグレーション」をクリックする.
(5)「着信 Web フック」をクリックし「設定の追加」をクリックする.
(6) 送信するチャンネルを選択し,「着信 Web フックインテグレーションの追加」をクリックする.

表 3　OutgoingWebhooks の設定

項目	内容
チャンネル	発言を監視するチャンネル
引き金となる言葉	@AmeBot”
URL	`https://<myapp_name>.herokuapp.com/slack`
	`<myapp_name>`は 6.2.1 項 (6) にて指定するアプリケーション名

(7) WebHook URL を取得する．
(8)「設定を保存する」をクリックする．

6.2.3　Slack アカウントの Outgoing WebHooks の設定

Slack が提供している Outgoing WebHooks の設定を行う．

(1) 6.2.2 項の (4) まで同様の手順を踏む．
(2)「発信 Web フック」をクリックし，「設定を追加」をクリックする．
(3)「発信 Web フックインテグレーションの追加」をクリックする．
(4) 表 3 に示す設定を行い，「設定を保存する」をクリックする．

6.2.4　Yahoo! JAPAN デベロッパーネットワーク Web API の Client ID の取得

Yahoo! JAPAN デベロッパーネットワークの提供している，Web API の Client ID を取得する．また，Client ID の取得には Yahoo! JAPAN アカウントが必要である．

(1) 以下の URL から Yahoo! JAPAN デベロッパーネットワークにアクセスする．
`https://developer.yahoo.co.jp/`
(2) ページ上部の「機能」をクリックし，「アプリケーションの開発」をクリックする．
(3) 本プログラムの使用者の Yahoo! JAPAN アカウントにログインする．
(4)「新しいアプリケーションの開発」をクリックする．
(5) 表 4 に示す，アプリケーション情報を入力し，「ガイドラインに同意する」にチェックを入れ，「確認」をクリックする．
(6) Client ID を取得する．なお，Client ID を API キーとして用いる．

7　使用方法

本プログラムを実行するためのコマンドを以下に示す．本プログラムは Heroku にデプロイすることにより実行することができる．

表 4　Yahoo! JAPAN デベロッパーのアプリケーション情報

項目	内容
アプリケーションの種類	サーバーサイド
連絡用メールアドレス	本プログラムの利用者の連絡用メールアドレス
アプリケーション名	任意のアプリケーション名
URL	`https://takaieslackbot.herokuapp.com/`

```
$git push heroku master
```

8　エラー処理

本プログラムで行ったエラー処理を以下に示す.

(1) (機能 2) において，指定された住所またはランドマーク名が日本国内でない場合，“Japan Only” というメッセージを発言する.
(2) (機能 2) において，指定された住所またはランドマーク名がみつからない場合，“not found” というメッセージを発言する.

9　保証しない動作

本プログラムが保証しない動作を以下に示す.

(1) (機能 2) において，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
(2) Slack の Outgoing Webhooks 以外からの POST リクエストを受け取ったときの動作
(3) 表 2 に示す動作環境以外でプログラムを実行
(4) 使用する API の提供の終了，または，仕様変更の際の動作

参考文献

[1] ヤフー株式会社：Yahoo!JAPAN, ヤフー株式会社 (online), available from ⟨https://www.yahoo.co.jp/⟩ (accessed 2018-05-17).
[2] Slack: Slack: Where work happens, Slack (online), available from ⟨https://slack.com⟩ (accessed 2018-05-17).
[3] Geocoding: Geocoding API - 住所から緯度経度を検索，Geocoding（オンライン），入手先 ⟨`https://dashboard.heroku.com`⟩ （参照 2018-05-17）.
[4] Heroku: Heroku, Heroku (online), available from ⟨https://dashboard.heroku.com⟩ (accessed

2018-05-17).

[5] ヤフー株式会社：Yahoo!デベロッパーネットワーク，ヤフー株式会社（オンライン），入手先 ⟨https://developer.yahoo.co.jp/⟩（参照 2018-05-17）.