2019年度B4新人研修課題 SlackBot仕様書 (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 プログラムの仕様についてまとめたものである．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．

2 対象とする利用者

本プログラムでは，以下の 2 つのアカウントを所有するユーザを対象とする．

(1) Slack アカウント
(2) GitHub アカウント

3 機能

本プログラムの動作の流れを図 1 に示す．また，図 1 について，処理の流れを以下で説明する．なお，以降，ユーザおよび本プログラムの投稿メッセージは “” で囲まれた部分で示す．

(1) ユーザが，“@Hirobot” から始まるメッセージを投稿する．
(2) 本プログラムが，投稿されたメッセージを Slack の機能である Outgoing Webhook [2] を用いてリクエストメッセージとして受け取る．Outgoing Webhook とは，特定の文字列が投稿された場合に，指定した URL にデータを POST する機能である．
(3) 本プログラムが，受け取った内容を処理したのち，処理したメッセージを Slack の機能である Incoming Webhook [3] を用いて Slack 上の指定したチャンネルに投稿する．Incoming Webhook とは，指定した URL に POST された文字列を Slack へ送信する機能である．
(4) 本プログラムによるチャンネルへの投稿が，ユーザのクライアント上に反映される．

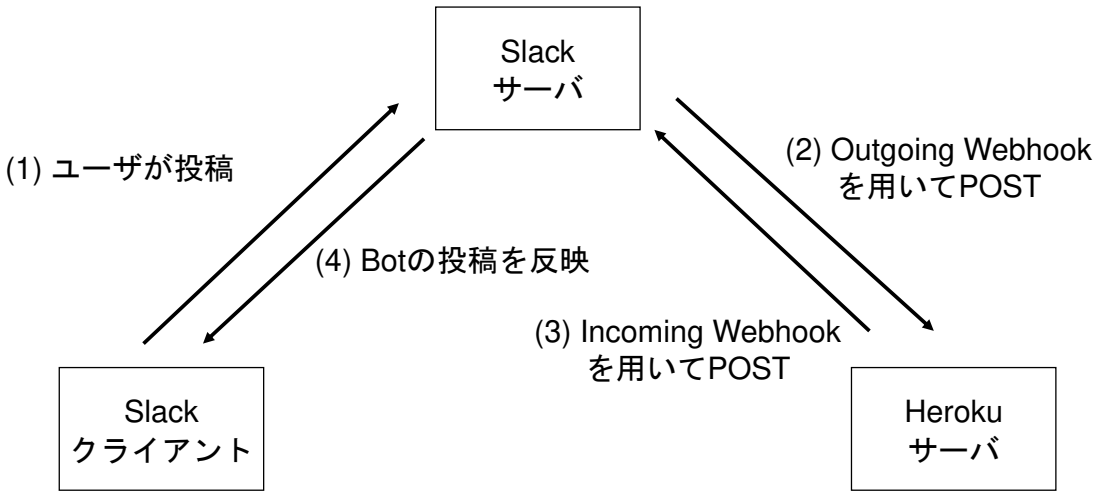

図 1　処理の流れ

また，本プログラムがもつ 3 つの機能を以下に述べる．

(機能 1)　呼びかけに応じる機能

リクエストメッセージが “@Hirobot” であった場合，“@(投稿したユーザのユーザ名) Hi!” と投稿する．

(機能 2)　指定した文字列を投稿する機能

リクエストメッセージが “@Hirobot 「(任意の文字列)」と言って” という形式であった場合，“(任意の文字列)” と投稿する．ただし，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含まれていた場合は，投稿内容が最長となる文字列を投稿する．例として，ユーザが “「あいうえお」と言って,「かきくけこ」と言って” と投稿した場合，本プログラムは “あいうえお」と言って,「かきくけこ” と投稿する．

(機能 3)　郵便番号に対し市区町村名を答える機能

リクエストメッセージが “@Hirobot (7 桁の数字)” という形式であった場合，本プログラムは (7 桁の数字) 部分を郵便番号とし，該当する県と市区町村の名前を投稿する．たとえば，ユーザが “@Hirobot 7000011” と投稿した場合，本プログラムは〒700-0011 に対応した住所である “岡山県岡山市北区学南町” と投稿する．該当する県名と市区町村名がなかった場合，“エラー！存在しない郵便番号です” と投稿する．郵便番号に対応した県名・市区町村名は，郵便番号検索 API [4] を使用して情報を取得する．

4　動作環境および動作確認環境

4.1　動作環境

本プログラムは Heroku [5] 上で動作する．Heroku とは，PaaS と呼ばれる形態のサービスである．PaaS とは，アプリケーションを実行するためのプラットフォームをインターネットを介して提供するサービスである．表 1 に，本プログラムを動作させる Heroku の環境を示す．

表 1　動作環境

項目	内容
OS	Ubuntu 18.04.2 LTS
CPU	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
メモリ	512MB
Ruby	ruby 2.5.1
Ruby Gem	Bundler version 2.0.1
	json 2.1.0
	mustermann 1.0.2
	rack 2.0.4
	rack-protection 2.0.1
	sinatra 2.0.1
	tilt 2.0.8
	xml-simple 1.1.5

4.2　動作確認済み環境

表 1 に示す環境にて，動作確認済みである．

5　使用方法

5.1　環境構築

5.1.1　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) 本プログラムの Fork，clone
(2) Slack アカウントの Incoming Webhook の設定
(3) Heroku アカウントの作成と設定

(4) Slack アカウントの Outgoing Webhook の設定

5.1.2 本プログラムの Fork，clone

本プログラムをローカルリポジトリに入れるため，GitHub より `Fork` および `clone` する．手順を以下に示す．

(1) GitHub にて，`https://github.com/owata-8492/BootCamp/tree/master/2019/Slackbot` を `Fork` する．
(2) ターミナルにて，以下のとおり入力する．
```
$git clone http://github.com/owata-8492/BootCamp/2019/Slackbot.git
```

5.1.3 Slack アカウントの Incoming Webhook の設定

Slack の機能である Incoming Webhook [3] を利用するための設定を行う．設定の手順を以下に示す．

(1) `https://slack.com/intl/ja-jp` より，Slack アカウントにログインする．
(2) ページ左上のチーム名から「Slack をカスタマイズ」を選択する．
(3)「Menu」から「App 管理」を選択する．
(4) ページ左の「カスタムインテグレーション」を選択する．
(5)「Incoming Webhook」を選択し「設定を追加」をクリックする．
(6) 項目「チャンネルへの投稿」を設定する．Bot が送信するチャンネルを選択する．
(7)「Incoming Webhook インテグレーションの追加」をクリックする．
(8) 項目「Webhook URL」について，表示されている URL をコピーする．
(9) 項目「名前をカスタマイズ」を設定する．「Hirobot」と入力する．
(10)「設定を保存する」をクリックする．

5.1.4 Heroku アカウントの作成と設定

本プログラムを Heroku 上にデプロイするために，Heroku アカウントの作成と設定を行う．手順を以下に示す．

(1) `https://www.heroku.com/`より Heroku にアクセスし，「無料で新規登録」から新規のアカウントを登録する．
(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 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．

(7) 以下のコマンドを実行し，Heroku の環境変数に Webhook URL を設定する．
`$heroku config:set INCOMING_WEBHOOK_URL="<webhook_url>"`
ここで，<webhook_url>は，5.1.3 項 (8) で取得した URL である．

(8) ファイル settings.yml.sample に，取得した Incoming Webhook URL を書き込む．また，ファイル名を settings.yml に変更する．

5.1.5 Slack アカウントの Outgoing Webhook の設定

Slack が提供している Outgoing Webhook [2] の利用するための設定を行う．設定の手順を以下に示す．

(1) 5.1.3 項の (4) までを同様に行う．
(2)「Outgoing Webhook」をクリックし，「設定を追加」をクリックする．
(3)「Outgoing Webhook インテグレーションの追加」をクリックする．
(4) 項目「チャンネル」を設定する．Bot が投稿を受け取ることのできるチャンネルを選択する．
(5) 項目「引き金となる言葉」を設定する．「@Hirobot」と入力する．
(6) 項目「URL」を設定する．「https://<myapp_name>.herokuapp.com/slack」と入力する．
ここで，<myapp_name>とは，5.1.4 項 (6) にて作成したアプリケーション名である．
(7) 項目「名前をカスタマイズ」を設定する．「Hirobot」と入力する．
(8)「設定を保存する」をクリックする．

5.2 実行方法

本プログラムは，commit 作成後，Heroku 上にデプロイすることで実行することができる．デプロイするためのコマンドを以下に示す．

```
$git push heroku master
```

デプロイ後は，Slack 上において特定の投稿をすることで自動的に動作する．本プログラムがもつ機能およびその利用のための投稿内容は 3 章に記載している．

6　エラー処理と保証しない動作

6.1　エラー処理

本プログラムにおいては，特に行ったエラー処理はない．

6.2　保証しない動作

本プログラムの保証しない動作を以下に示す．

(1) Slack の Outgoing Webhook 以外から POST リクエストを受け取った場合の動作
(2) 4 章で述べた動作環境・動作確認済み環境・プラットフォーム外でのプログラム実行
(3) 使用している API およびプラットフォームの提供終了時，または仕様変更時の動作

参考文献

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

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

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

[4] zip.cgis.biz: 郵便番号検索 API, zip.cgis.biz（オンライン），入手先 ⟨`http://zip.cgis.biz/`⟩（参照 2019-04-15）.

[5] heroku: heroku, heroku (online), available from ⟨https://jp.heroku.com/⟩ (accessed 2019-04-15).