SlackBot プログラムの仕様書 (修正版)

2020/6/24

三宅 貴義

1 修正箇所

(1) “ 任意の ” という言葉を “ 指定の ” に修正した.
(2) 4 章で，処理の流れを表す図を追加した.
(3) 5 章で，機能の名前を統一した.
(4) 4 章で，実際のやり取りを図に変更した.
(5) 6 章で，設定を説明する順番を修正した.
(6) 6 章で，Incoming Webhook 及び，Outgoing Webhook をアプリケーションと言及していたことを修正.
(7) “ Github ”, “ Incoming Webhooks ” や “ Place API ” などの誤記を修正した.
(8) その他の日本語の間違いを修正した.

2 はじめに

本資料は 2020 年度 B4 新人研修課題の 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，Slack 上に特定のメッセージが投稿された際に，内容に応じた返信を自動で行うプログラムである．本プログラムは以下の 3 つの機能を持つ.

(1) 文字列を返信する機能
(2) 郵便番号を住所に変換する機能
(3) コンビニを検索する機能

3 対象となる利用者

本プログラムを利用するには，以下の 3 つのアカウントを所有している必要がある.

(1) Slack アカウント
(2) Google アカウント
(3) クレジットカードアカウント

なお，クレジットカードは Google Places API を利用するために必要であり，API のリクエストが規定値を超えると請求が発生する.

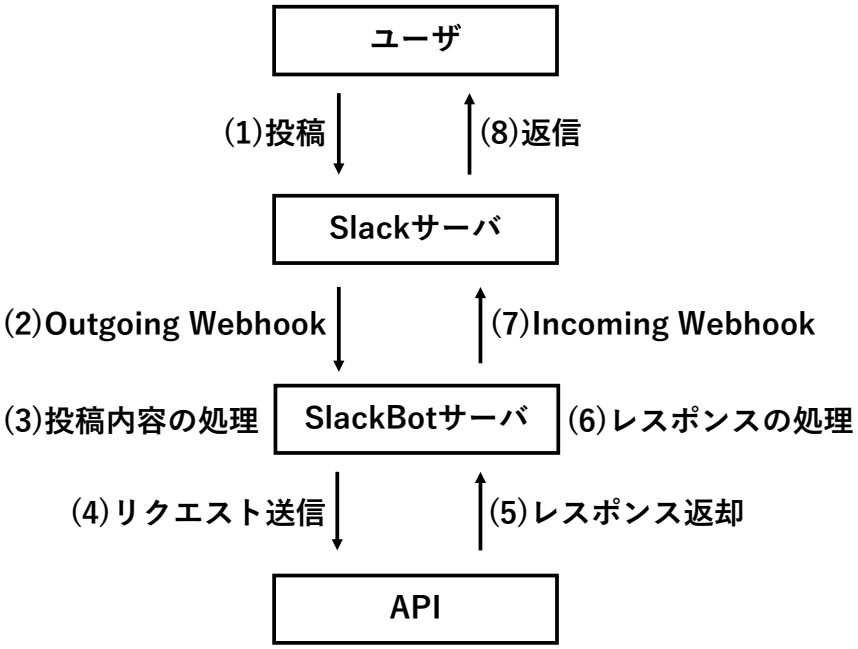

図 1　SlackBot プログラムの処理の流れ

4　SlackBot プログラムの処理の流れ

SlackBot プログラムの処理の流れを図 1 に示す.
また，図 1 の処理について以下に説明を示す.

(1) ユーザが Slack サーバにメッセージを投稿する.
(2) SlackBot サーバは，Outgoing Webhook の機能で，Slack サーバから投稿内容を受け取る.
(3) SlackBot サーバは，投稿内容を処理し，リクエストを作成する.
(4) SlackBot サーバは，API にリクエストを送信する.
(5) SlackBot サーバは，API から返却されたレスポンスを受け取る.
(6) SlackBot サーバは，レスポンスを処理し，メッセージを作成する.
(7) SlackBot サーバは，Slack サーバへ Incoming Webhook を利用してメッセージを送信する.
(8) ユーザが SlackBot サーバのメッセージを返信として受け取る.

5　機能

この章では SlackBot プログラムの機能について説明を行う.

図 2　文字列を返信する機能のやり取り

図 3　郵便番号を検索する機能のやり取り

5.1　文字列を返信する機能

この機能はユーザーが指定した文字列を返信する機能である．この機能を使用するには Slack に以下のように投稿する．
@MiyakeBot 「⟨ 文字列 ⟩」と言って
⟨ 文字列 ⟩ には返信させる文字列を入力する．
具体的なやり取りを図 2 に示す．

5.2　郵便番号を住所に変換する機能

この機能はユーザが指定した郵便番号を，住所に変換して返信する機能である．この機能を使用するには Slack に以下のように投稿する．
@MiyakeBot 「⟨ 郵便番号 ⟩」の住所を検索
⟨ 郵便番号 ⟩ には検索する郵便番号を入力する．
具体的なやり取りを図 3 に示す．

5.3　コンビニを検索する機能

この機能はユーザが指定した場所に関する情報から，その近辺のコンビニを指定した件数だけ表示する機能である．この機能を使用するには以下のように投稿する．
@MiyakeBot 「⟨ 場所 ⟩」のコンビニを ⟨ 件数 ⟩ 件検索
⟨ 場所 ⟩ には検索する場所を，⟨ 件数 ⟩ には表示する件数を数字で入力する．
具体的なやり取りを図 4 に示す．

図 4　コンビニを検索する機能のやり取り

表 1　動作環境

項目	内容
CPU	Intel(R) Xeon(R) CPU E5-2670 v2 @2.50GHz
メモリ	512MB
OS	Ubuntu 18.04.4 LST
Rub	ruby 2.6.6p146
Gem	activesupport 6.0.2.2
	bundler 2.0.2
	rack 2.2.2
	rack-protection 2.0.8.1
	ruby2_keyeords 0.0.2
	sinatra 2.0.8.1
	tilt 2.0.10
	mustermann 1.1.1

6　動作環境

本プログラムは Heroku で動作する．Heroku とはクラウド上でアプリケーションの実行が可能な PaaS である．Heroku の環境を表 1 に示す．

7　環境構築

本プログラムを利用するには以下の手順に従って設定を行う．

(1) 本プログラムの入手
(2) Incoming Webhook の設定
(3) Google Places API の設定
(4) Heroku の設定

(5) Outgoing Webhook の設定

それぞれの項目について，詳細を説明する．

7.1 本プログラムの入手

本プログラムは GitHub 上で公開，管理されている．本プログラムを利用するには以下のコマンドを実行し，BootCamp レポジトリをクローンする．

```
$ git clone https://github.com/miyake13000/BootCamp
```

クローンしたレポジトリ内の SlackBot ディレクトリを作業ディレクトリに変更する．

```
$ cd BootCamp/2020/SlackBot
```

7.2 Incoming Webhook の設定

Incoming Webhook とは外部サービスから Slack にメッセージを投稿する機能である．
以下に Incomig Webhook の設定の手順を示す．

(1) 自身の Slack アカウントにログインし，ワークスペースの設定から，「その他の管理項目」，「アプリを管理する」，「カスタムインテグレーション」，「Incoming Webhook」，「Slack に追加」の順にアクセスする．
(2)「チャンネルへの投稿」の項目で，投稿するチャンネルを指定する．
(3) Incoming Webhook URL を SlackBot ディレクトリ内の settings.yml ファイルに以下のように記述する．ただし，`<Incoming Webhook URL>`は自身の Incoming Webhook URL である．
`INCOMING_WEBHOOK_URL : <Incomig Webhook URL>`
(4)「設定を保存」をクリックする．

7.3 Google Places API の設定

本プログラムは Google Places API を用いているため，Google が発行する API キーがなければ利用することができない．
以下の手順で Places API の設定を行う．

(1) ブラウザで以下のサイトにアクセスし，自身の Google アカウントにログインする．
`https://cloud.google.com/maps-platform/`
(2) 右上の「開始」ボタンをクリックし，「新しいプロジェクトの作成」を選択し，「プロジェクトの選択」から「新しいプロジェクト」に進み，プロジェクト名をつけた後「作成」をクリックする．
(3)「請求先アカウントの作成」をクリックし，クレジットカード情報を入力する．
(4)「API の概要に移動」をクリックし，「Places API」を選択，「有効にする」をクリックする．
(5)「認証情報」を選択し，API キーが表示されていることを確認し，SlackBot ディレクトリ内の

setting.yml に以下のように追記する．ただし，<Your_API_key>は自身の API キーである．

```
API_KEY : <Your_API_key>
```

7.4 Heroku の設定

Heroku の設定について説明を行っていく．

(1) 以下のサイトにアクセスし，「Sign up」から新しいアカウントを作成する．
```
https://www.heroku.com
```
(2) 登録したアカウントにログインし，Ruby を選択し，「I'm ready to start」をクリックする．
(3) 自身の OS を選択し，Heroku　CLI をダウンロードする．
(4) 以下のコマンドを実行し，メールアドレスとパスワードを入力して Heroku にログインする．
```
$ heroku lonin
```
(5) 以下のコマンドを実行して，Heroku 上にアプリケーションを作成する．
```
$ heroku create
```
ここで表示された Heroku アプリケーションの URL が SlackBot サーバの URL となる．これは後の設定で用いるため，控えておく．
(6) 以下のコマンドを順次実行し，heroku 上に SlackBot プログラムをデプロイする．
```
$ git add --all
$ git commit -m "First deploy."
$ git push origin master
```

7.5 Outgoing Webhook の設定

Outgoing Webhook とは，Slack に特定の文字列を含む投稿がされると，外部サービスにメッセージを送信する機能である．

以下に Outgoing Webhook の設定の手順を示す．

(1) 自身の Slack アカウントにログインし，ワークスペースの設定から，「その他の管理項目」，「アプリを管理する」，「カスタムインテグレーション」，「Outgoing Webhook」，「Slack に追加」の順にアクセスする．
(2)「チャンネルへの投稿」の項目で，投稿するチャンネルを指定する．
(3) 引き金となる単語を設定する．
(4) URL に Heroku の設定で控えた SlackBot サーバの URL を入力する．
(5)「設定を保存」をクリックする．

8　使用方法

Slack 上で指定したチャンネルで，5 章で示した発言をすることで，自動的に SlackBot の返信が行われる．

9　エラー処理

本プログラムで実装しているエラー処理に関して以下に示す．

(1) 郵便番号を検索する機能に関して，郵便番号以外の値が入力された場合への対処．この場合，以下のようなメッセージを返す．
Invalid Zipcode

(2) コンビニを検索する機能に関して，表示できるコンビニが存在しない場合への対処．この場合，以下のようなメッセージを返す．
No hit!

(3) 無効なリクエストを送信した場合や，API との接続に失敗した場合への対処．この場合，それぞれ以下のようにメッセージを返す．
郵便番号を検索する機能：Failed to exchange zipcode to address.
コンビニを検索する機能：Failed to search convenience_store.

10　保証しない動作

本プログラムは以下に示す状況に関して動作の保証をしていない．

(1) コンビニを検索する機能で，件数に負の数や，小数が入力された場合．

11　おわりに

本資料では新人研修課題の SlackBot プログラムの仕様を説明した．