SlackBot プログラムの仕様書

2020/5/27

中川 雄介

1　概要

本資料は 2020 年度 B4 新人研修課題で作成した SlackBot プログラムの仕様についてまとめたものである．本プログラムで使用する Slack とは Web 上で利用できるビジネス向けのチャットツールである．SlcakBot とは，Slack 上でのユーザの特定の発言を契機に Slack に発言するプログラムである．本プログラムは，以下の 2 つの機能をもつ．

(1) 任意の文字列を発言する機能
(2) ランダムに生成された文字列を発言する機能

なお，本資料における発言とは Slack の特定のチャンネルに文字列を投稿することを指す．また，本資料においての発言内容は “ ” で囲って表す．

2　対象とする利用者

本プログラムは以下のアカウントを所有する利用者を想定している．

(1) Slack アカウント
(2) GitHub アカウント

GitHub とはソフトウェア開発のプラットフォームである．GitHub にソースコードをホスティングすることで複数人のソフトウエア開発者と協働してコードをレビューしたり、プロジェクトを管理しつつ開発を行ったりすることができる．

3　機能

本プログラムは Slack での “@nakagawabot” から始まるユーザの発言を受信し，これに対して SlackBot が発言する．SlackBot が発言する内容は “@nakagawabot” に続く文字列により決定される．以下で本プログラムがもつ 2 つの機能について述べる．

機能 1　任意の文字列を発言する機能

この機能は，ユーザの “@nakagawabot 「(任意の文字列)」と言って” という発言に対して，SlackBot が (任意の文字列) と発言する．例えば，“@nakagawabot 「こんにちは」と言って” とユーザが発言した場合，Slackbot は “こんにちは” と発言する．

機能 2　ランダムに生成された文字列を発言する機能

ユーザが “@nakagawabot 「password」” と発言した場合，SlackBot はランダムに生成された文字列を発言する．例えば，“@nakagawabot 「password」” という発言に対し，SlackBot は”knockers”のように発言する．

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した場合，SlackBot は”@(ユーザー名) Hi!”のように発言する．また，(機能 1，2) について，“@nakagawabot 「password」と言って” と発言した場合，“password” と発言する．

4 動作環境

本プログラムの動作環境を表 1 に示す．なお，本プログラムが表 1 の動作環境で動作することは確認済みである．

表 1 動作環境

項目	内容
OS	Windows 10
CPU	Intel(R) Core(TM) CPU m3-6Y30 @ 0.90GHz 1.51GHz
メモリ	4.00GB
Ruby	ruby 2.6.6p146
Ruby Gem	bundler 3.0.3
	mustermann 1.0.2
	rack 2.0.4
	rack-protection 2.0.1
	sinatra 2.0.1
	tilt 2.0.8
heroku	7.40.0 win32-x64
	node v12.16.2

5 環境構築

5.1 概要

本プログラムの動作のために必要な環境構築の項目を以下に示す．

(1) Slack の Incoming WebHook の設定
(2) Slack の Outgoing WebHook の設定
(3) API のアドレス取得

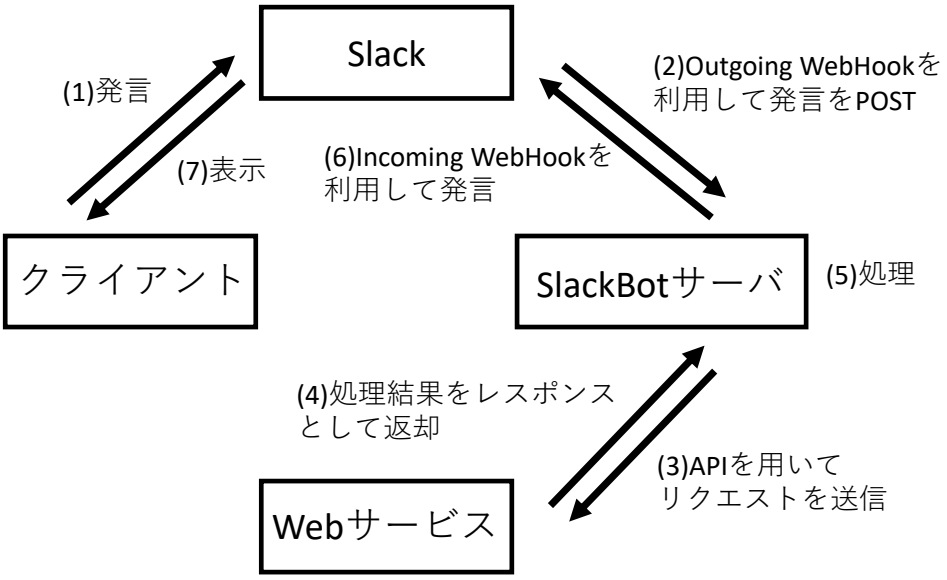

図 1　WebHook と API を利用した SlackBot の処理の流れ

(4) Heroku の設定

Incoming WebHook とは，外部サービスから Slack に発言する機能である．Outgoing WebHook 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．本プログラムでは，この 2 つの WebHook 機能を利用するため設定を行う必要がある．WebHook 機能と API を利用した SlackBot の処理の流れを以下で説明する．

(1) クライアントが Slack に発言する．
(2) Slack が Outgoing WebHook を利用し，発言を SlackBot サーバに POST する．
(3) SlackBot サーバが 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 API に応じた処理結果をレスポンスとして SlackBot サーバに返却する．
(5) SlackBot サーバが受け取ったレスポンスを元に，Slack に発言する文字列を生成する．
(6) SlackBot サーバが Incoming WebHook を利用し，Slack に発言する．
(7) Slack が SlackBot サーバの発言を表示することで，クライアントは発言を確認できる．

5.2 手順

5.2.1 Slack の Incoming WebHook の設定

(1) 以下の URL にアクセスする．

`https://<work_space>.slack.com/apps/manage/custom-integrations`

ただし，`<work_space>`は自分のワークスペース名を示す．ワークスペースとは，他のメンバーとコミュニケーションをとり業務を行うための共有スペースである．

(2)「Incoming WebHook」をクリックする．

(3)「設定の追加」から，新たな Incoming WebHook を追加する．

(4)「チャンネルへの投稿」で送信するチャンネルを選択する，

(5)「Webhook URL」に表示されている URL を，5.2.4 項 (8) で使うためひかえておく．

(6)「設定を保存する」をクリックする．

5.2.2 Slack の Outgoing WebHook の設定

(1) 以下の URL にアクセスする．

`https://<work_space>.slack.com/apps/manage/custom-integrations`

ただし，`<work_space>`は自分のワークスペース名を示す．

(2)「Outgoing WebHook」をクリックする．

(3)「設定を追加」をクリックし，「Outgoing Webhook インテグレーションの追加」をクリックする．

(4) Outgoing WebHook に関して以下を設定する．

（A）チャンネル：発言を監視するチャンネル

（B）引き金となる言葉：WebHook が動作する契機となる文字列

（C）URL：WebHook が動作した際に POST を行う URL

(5)「設定を保存する」をクリックする．

5.2.3 ランダムなユーザー生成 API の URL の取得

Random User Generator という名前の API の URL を取得する．

(1) 以下に URL を示す．この URL に対し，リクエストを送信することでランダムなユーザー情報を受け取ることができる．

`https://randomuser.me/api/`

5.2.4 Heroku の設定

Heroku はアプリの構築、提供、監視、スケールに役立つクラウドプラットフォームである.

(1) 以下の URL より Heroku にアクセスし,「Sign up」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 with Heroku」の使用する言語として「Ruby」を選択する.

(3) ターミナルで以下のコマンドを実行し,Heroku CLI をインストールする.

```
$ curl https://cli-assets.heroku.com/install.sh | sh
```

(4) 以下のコマンドを実行し,Heroku CLI がインストールされたことを確認する.

```
$ heroku version
```

(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.

```
$ heroku create <app_name>
```

ただし,<app_name>は任意のアプリケーション名を示す.アプリケーション名には英語のアルファベットの小文字,数字,およびハイフンのみ使用できる.

(7) 以下のコマンドで生成したアプリケーションが Heroku に登録されていることを確認できる.

```
$ git remote -v
heroku https://git.heroku.com/<myapp_name>.git (fetch)
heroku https://git.heroku.com/<myapp_name>.git (push)
```

(8) 以下のコマンドを実行し,Heroku の環境変数に 5.2.1 項 (5) で取得した Webhook URL を設定する.

```
$ heroku config:set INCOMING_WEBHOOK_URL="https://*****"
```

6 使用方法

(1) 下記の Git リポジトリから本プログラムを取得する.

```
git@github.com:nakagawa1210/BootCamp.git
```

(2) 取得した本プログラムは Heroku にデプロイすることで使用できる．デプロイは以下のコマンドを実行する．

```
$ git push heroku master
```

7　エラー処理と保証しない動作

7.1　エラー処理

本プログラムでは特にエラー処理は行っていない．

7.2　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) Slack の Outgoing WebHook 以外からの POST リクエストを受け取った際の動作