SlackBot プログラムの仕様書

2020/6/24

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) 指定の文字列を発言する機能
(2) ランダムな単語を発言する機能

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　対象とする開発者

本プログラムは以下のアカウントを所有する開発者を想定している．

(1) Slack アカウント

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　指定の文字列を発言する機能

この機能は，ユーザの “@nakagawabot 「(指定の文字列)」と言って” という発言に対して，SlackBot が (指定の文字列) 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　パスワードの候補となる文字列を発言する機能

この機能は，ユーザの “@nakagawabot 「password」” という発言に対して，SlackBot はパスワードの候補となる文字列を発言する．例えば，“@nakagawabot 「password」” とユーザが発言した場合，SlackBot は “knockers” のように発言する．この発言内容はよく利用されているパスワード上位 1 万のリストの中からランダムに選択されている．

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”@(ユーザ名) Hi!”のように発言する．また，機能 1，機能 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

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

表 1 Heroku の環境

項目	内容
OS	Ubuntu 18.04.4 LTS
CPU	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
メモリ	512MB
Ruby	ruby 2.6.6p146
Ruby Gem	bundler 1.17.3
	mustermann 1.1.1
	rack 2.2.2
	rack-protection 2.0.8.1
	ruby2_keywords 0.0.2
	sinatra 2.0.8.1
	tilt 2.0.10

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
(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 の処理の流れを図 1 に示し，以下で説明する．

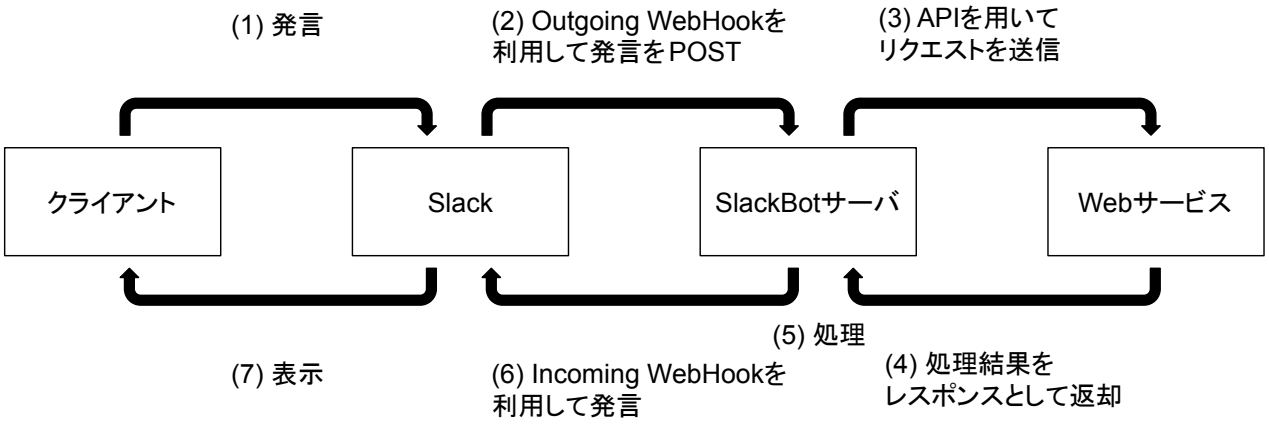

図 1　WebHook と API を利用した SlackBot の処理の流れ

(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

以下の Slack に関する設定はワークスペースが「nomlab」の場合である.

5.2.1　Slack の Incoming WebHook の設定

(1) 自分の Slack アカウントにログイン
(2) Slack 画面左上の「nomlab」->「その他管理項目」->「以下をカスタマイズ：nomlab」をクリック．左上の「Menu」から「App 管理」->「カスタムインテグレーション」をクリックする.
(3)「Incoming WebHook」をクリックする.
(4)「Slack に追加」から，新たな Incoming WebHook を追加する.

(5)「チャンネルへの投稿」で送信するチャンネルを選択する，
(6)「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．
(7)「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) 自分の Slack アカウントにログイン
(2) Slack 画面左上の「nomlab」->「その他管理項目」->「以下をカスタマイズ：nomlab」をクリック．左上の「Menu」から「App 管理」->「カスタムインテグレーション」をクリックする．
(3)「Outgoing WebHook」をクリックする．
(4)「Slack に追加」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．
(5) 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
(6)「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情報を生成する API の動作の確認

Random User Generator という API がある．この API を利用してランダムなユーザの情報を生成し，その情報を受け取る．このプログラムでは生成された情報の中の password の項目を利用している．

(1) 以下に API の 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に生成されたユーザの情報を受け取ることができる．

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

(2) 使用した API のサイトの URL を以下に示す [1]．このページを Web ブラウザで開くとその際にランダムに生成されたユーザの情報を確認することができる．

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

5.2.4 Heroku の設定

Heroku はアプリの構築，提供，監視，及び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 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 を選択し，Heroku CLI をダウンロードする．

(4) ターミナルで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) 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>
```

<app_name>は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/<app_name>.git (fetch)
heroku https://git.heroku.com/<app_name>.git (push)
```

(8) Heroku の環境変数に Webhook URL を設定する．以下のコマンドの"https://*****"を 5.2.1 項 (6) で取得した Webhook URL に置き換え実行する．

```
$ 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．このため (機能 2) で使用しているサイトが閉鎖していたり，障害でアクセスできなかったりするとプログラムが正常に動作せず，発言が返ってこない．

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

参考文献

[1] Random User Generator,　入手先<https://randomuser.me/ >(参照 2020-06-23).