

システムプログラミング特論 課題 4

1. はじめに

当レポートではXXXX年XQシステムプログラミング特論の課題4成果物として作成したものである。課題の要件、プログラムの該当、ソースコードなど課題として要求された内容について説明する。

2. プログラムの概要

2.1. プログラムの動作

プログラムを実行するとAIITのブログサイト「Info Press」から記事のタイトルとURLを取得しSlackに設置した専用のチャンネルに投稿する。

処理に若干時間がかかるため、開始時にはプログラムの実行画面で「Slackの指定したチャンネルにAIITブログInfoPressの直近の記事を送信します」と、終了時には「Slackへの送信が完了しました。チャンネルを確認してください」とメッセージを表示する。

2.2. 開発環境/実行環境など

- 開発言語 : Python3
- ライブラリ : python3-pip、python3-dev、pip-feedparser
- ファイル名 : kadai4_mushup.py

2.3. 使用したAPI

- Slack用API : 着信Webフック
 - https://aiitalphateam.slack.com/apps/A0F7XDUAZ--web-?next_id=0
- AIIT infopress RSS フィード
 - ページURL : <http://aiit-isa.hatenablog.com/>
 - フィードURL : <http://aiit-isa.hatenablog.com/feed>

3. ソースコード

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import feedparser
import requests
import json

d = feedparser.parse("http://aiit-isa.hatenablog.com/feed")

#print("feed:", d.channel.title)
#print("description:", d.channel.description)
print("Slackの指定したチャンネルにAIITブログInfoPressの直近の記事を送信します")

for e in d.entries:
    # print("{}: {}".format(e.title, e.link))
    post = "{}: {}".format(e.title, e.link)
```

```
SLACK_POST_URL = "XXXXXXXX" (ここには Webhook URL が入る)
```

```
post_json = {  
    "text": post  
}  
requests.post(SLACK_POST_URL, data = json.dumps(post_json))
```

```
print("Slack への送信が完了しました。チャンネルを確認してください")
```

4. 実行結果

実行結果の出力画面は、Linux サーバーのターミナル画面と Slack のチャンネル画面の 2 種類がある。

Linux サーバのターミナル画面(Google Cloud Platform で実行)



```
Cloud Shell  
ait-alpha-team x +  
#01 セミナー「教育現場で進むICP活用」(2017/7/12): http://aiit-isa.hatenablog.com/entry/2017/06/12/014041  
#02 セミナー「IoTと画像処理で未来のフツウをつくる」(2017/7/21): http://aiit-isa.hatenablog.com/entry/2017/06/11/145316  
dashii150@ubuntu-dev-002 kadai_004]$ vim kadai4_mushup.py  
dashii150@ubuntu-dev-002 kadai_004]$ git add .  
dashii150@ubuntu-dev-002 kadai_004]$ git commit -m "とりあえず動いた"  
[master 2f52687] とりあえず動いた  
1 file changed, 2 deletions(-)  
dashii150@ubuntu-dev-002 kadai_004]$ git push  
Counting objects: 4, done.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (4/4), 365 bytes | 0 bytes/s, done.  
Total 4 (delta 2), reused 0 (delta 0)  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
github.com:al852rw/aiit_system_programing.git  
2f52687..2f52687 master -> master  
dashii150@ubuntu-dev-002 kadai_004]$ git pull  
Counting objects: 4, done.  
Compressing objects: 100% (1/1), done.  
Total 4 (delta 2), reused 4 (delta 2), pack-reused 0  
Writing objects: 100% (4/4), done.  
github.com:al852rw/aiit_system_programing  
2f52687..f44fb86 master -> origin/master  
Updating 2f52687..f44fb86  
Re-forward  
git 004/kadai4_mushup.py | 9 ++++++  
1 file changed, 6 insertions(+), 3 deletions(-)  
dashii150@ubuntu-dev-002 kadai_004]$ python3 kadai4_mushup.py  
実行結果の出力画面は、Linux サーバーのターミナル画面と Slack のチャンネル画面の 2 種類がある。
```

処理の開始時と終了時に所定のメッセージが表示されている。

Slack のチャンネル



infoPress の記事のタイトルと URL が投稿されている。

5. 工夫したところ

当初はターミナル画面上にも投稿内容を表示していたが、記事の数が多く画面が埋まってしまい処理の内容がよくわからなかった。

そのため、ターミナル画面へは非表示として代わりに開始/終了のメッセージを表示するようにした。

記事の取得においては、RSS フィードを抽出するために専用の Web サービスを用いて URL を特定した。

使用したサービス

- ・ サービス名 : BeRSS
- ・ ページ URL : <https://berss.com/feed/Find.aspx>

手順の面で工夫した点は下記の通りである。

当初は Window の環境で WSL の Ubuntu を使用してプログラムを組んでいたが、この環境では pip のインストールができないことがわかったため、動作確認には Google Cloud Platform 上に構築した Ubuntu を使用した。

データの共有には Git-Hub を使用し、WSL でプログラムを作成→Git-Hub にプッシュ→Google Cloud Platform 上で pull の順で操作することで実現した。これにより使用している PC のスペックにかかわらずクラウド上で安定した動作確認をすることができた。