README

仕様について

今回は課題書に書かれた機能をつけたAPIサーバーをscalaを用いて作成した。今回初めてscalaを触りスケーラブルな言語という意味を少し理解できたと思う。
主な機能はTwitterのようなマイクロブログアプリケーションを作成するというものである。保存する先はMySQLを用いる。そのためDBとの非同期通信が必要になる。また、パスを応じてのルーティングやその処理も行う必要がある。

できた部分

今回は時間内に指定した通りの仕様にすることはできなかった。そのためチェックコードでの評価が悪かった。しかし、必要な機能は全て設定することはできた。 それについて説明を行いたい。
まず用いたDBのデータを表示する。最初の図が何も操作していない状態。もう一つの図が全操作を行った後の状態を指す。

```
mysql> SELECT * FROM POST;
+--------------------------------------+--------------------------------------+-------+----------------+---------------+---------------------+
| ID                                   | USER_ID                              | TEXT  | PARENT_POST_ID | COMMENT_COUNT | POSTED_AT           |
+--------------------------------------+--------------------------------------+-------+----------------+---------------+---------------------+
| c7ca642d-ca86-441e-8128-67a327bc6c0f | 22222222-2222-2222-2222-222222222222 | hi    |                |             1 | 2019-06-18 23:48:46 |
| abfd8036-8bb8-4962-8bc0-b7939d92a762 | 11111111-1111-1111-1111-111111111111 | hello |                |             1 | 2019-06-18 23:48:46 |
| e4da8c6f-7c41-4c98-a52b-036ac86f35fb | 33333333-3333-3333-3333-333333333333 | luck  |                |             1 | 2019-06-18 23:48:46 |
+--------------------------------------+--------------------------------------+-------+----------------+---------------+---------------------+
3 rows in set (0.01 sec)
```

```
mysql> SELECT * FROM POST;
+--------------------------------------+--------------------------------------+-----------+--------------------------------------+---------------+---------------------+
| ID                                   | USER_ID                              | TEXT      | PARENT_POST_ID                       | COMMENT_COUNT | POSTED_AT           |
+--------------------------------------+--------------------------------------+-----------+--------------------------------------+---------------+---------------------+
| c7ca642d-ca86-441e-8128-67a327bc6c0f | 22222222-2222-2222-2222-222222222222 | hi        |                                      |             1 | 2019-06-18 23:48:46 |
| abfd8036-8bb8-4962-8bc0-b7939d92a762 | 11111111-1111-1111-1111-111111111111 | hello     |                                      |             1 | 2019-06-18 23:48:46 |
| e4da8c6f-7c41-4c98-a52b-036ac86f35fb | 33333333-3333-3333-3333-333333333333 | luck      |                                      |             1 | 2019-06-18 23:48:46 |
| b0357bdb-a49f-43fb-91f8-1690623c9f27 | 11111111-1111-1111-1111-111111111111 | hellohello |                                      |             1 | 2019-06-18 23:50:21 |
| 5a1481fc-de81-4a83-b60a-e61a161f1f22 | 11111111-1111-1111-1111-111111111111 | hellohello | e4da8c6f-7c41-4c98-a52b-036ac86f35fb |             2 | 2019-06-18 23:51:45 |
+--------------------------------------+--------------------------------------+-----------+--------------------------------------+---------------+---------------------+
5 rows in set (0.01 sec)
```

1. GET /posts

このルーティングは全投稿の表示である。 仕様では、**{“posts”: [全投稿]}** という形で出力する必要があるが、作成したAPIサーバーはDB内のデータをjson形式にして表示する。そのためjsonがワンラインになった。また、最初の**"post":** を付けて表示していない。

```
TAKE-PC-6:~ take-pc-6$ curl http://localhost:8080/posts/
[{"comment_count":1,"text":"hi","parent_post_id":"","user_id":"22222222-2222-2222-2222-222222222222","id":"c7ca642d-ca86-441e-8128-67a327
bc6c0f","posted_at":"2019-06-18 23:48:46"},{"comment_count":1,"text":"hello","parent_post_id":"","user_id":"11111111-1111-1111-1111-11111
1111111","id":"abfd8036-8bb8-4962-8bc0-b7939d92a762","posted_at":"2019-06-18 23:48:46"},{"comment_count":1,"text":"luck","parent_post_id"
:"","user_id":"33333333-3333-3333-3333-333333333333","id":"e4da8c6f-7c41-4c98-a52b-036ac86f35fb","posted_at":"2019-06-18 23:48:46"}]TAKE-
```

2. POST /posts/create

このルーティングは投稿の作成にあたる。 仕様では成功時、**{“result”:”ok”}** となり、失敗時、**{“result”:”NG”, “message”:[特定のエラー]}** と表示する必要がある。作成したサーバーでは、成功、失敗の両方の場合において適切なメッセージを表示することができた。
今回はユーザーのIDを事前にDBに登録しておき、投稿がそのユーザーか否かを判断する必要がある。

```
TAKE-PC-6:~ take-pc-6$ curl -H "Content-type: application/json" -X POST -d '{"user_id": "11111111-1111-1111-1111-111111111111","text": "h
ellohello"}' http://localhost:8080/posts/create
{'result': 'OK'}
```

```
TAKE-PC-6:~ take-pc-6$ curl -H "Content-type: application/json" -X POST -d '{"user_id": "11111111-1111-1111-1111-11111111111a","text": "h
ellohello"}' http://localhost:8080/posts/create
{'result': 'NG', 'message': '該当するユーザーIDが登録されていません'}
```

3. GET /posts/:post_id/comments

このルーティングではクエリ上で指定したIDをもつ投稿を表示するというもの。 DBから該当する投稿を取得して表示することが必要になる。作成したサーバーでも適切に投稿を取得して表示することができた。

```
TAKE-PC-6:~ take-pc-6$ curl http://localhost:8080/posts/5a1481fc-de81-4a83-b60a-e61a161f1f22/comments
{"comment_count":2,"text":"hellohello","parent_post_id":"e4da8c6f-7c41-4c98-a52b-036ac86f35fb","user_id":"11111111-1111-1111-1111-1111111
11111","id":"5a1481fc-de81-4a83-b60a-e61a161f1f22","posted_at":"2019-06-18 23:51:45"}TAKE-PC-6:~ take-pc-6$
```

4. POST /posts/:post_id/comments/create

このルーティングではクエリ上で指定した投稿に対してのコメントを作成するというもの。 DBから該当する投稿があるかどうかを確認したあと、コメントを保存するということが求められる。またコメントをする投稿のcomment_countを増やすということも必要である。 作成したサーバーでも必要な動作を行うことができた。

```
TAKE-PC-6:~ take-pc-6$ curl -H "Content-type: application/json" -X POST -d '{"user_id": "11111111-1111-1111-1111-111111111111","text": "h
ellohello"}' http://localhost:8080/posts/e4da8c6f-7c41-4c98-a52b-036ac86f35fb/comments/create
{'result': 'OK'}
```

```
TAKE-PC-6:~ take-pc-6$ curl -H "Content-type: application/json" -X POST -d '{"user_id": "11111111-1111-1111-1111-111111111111","text": "h
ellohello"}' http://localhost:8080/posts/111/comments/create
{'result': 'NG', 'message': '該当する投稿IDが登録されていません'}
```

できなかった部分

1. 表示がうまくjson形式に整形できなかった。
   全投稿の表示はjsonとして表示できたが、postした時のメッセージやget,postのエラーメッセージがplain/text形式になってしまった。整形する方法を探したがうまく見つけることができなかった。

2. jsonの表示がうまくできなかった。
   全投稿の表示や該当のコメントを表示する際、DBからデータを取得することはできたがそれをそのまま出力するプログラムにしたため、要求される形で出力できなかった。