

SIMPLE GIT GUIDE(command)

1. Git hub에서 파일을 다운로드 할 건데, 파일 다운로드 할 위치에서 cmd창을 켜다 start command prompt with ruby

```
cd /*local folder path*/
```

2. 로컬에서 작업한 후 git hub에 업데이트할 거니까 환경 설정하고 clone한다.

```
git init
```

```
git clone /*repository*/
```

**** Repository**

- Server : <https://github.com/zzokokeic/barogagi1gibateam.git>
- Fuse : <https://github.com/zzokokeic/barogagifuze>
- Heroku : <https://git.heroku.com/arcane-gorge-90774.git>

Heroku app : <https://arcane-gorge-90774.herokuapp.com>

3. 파일을 수정/업데이트한다!

이 때, 실제 파일 다운로드 는 /*local folder path*/ 안에 barogagi1gibateam 이름으로 완료됨

4. 파일 수정을 완료한 후 git hub에 push한다.

/*local folder path*/barogagi1gibateam 위치에서 git status 하면 modified / new / delete 등으로 확인 가능,

```
git add .
```

```
git commit -m "메세지내용 - (최대한 작업한 내용 자세히 써줍시다)"
```

```
git push origin master
```

DB Debugging

```
** rake db:drop:_unsafe
```

혹은 Schema.rb / dev.sqlite3 두개 파일 삭제

```
** rake db:migrate  
** rake db:seed
```

이 때 seeds.rb는 barogagi1gibateam/db/ 안에 존재해야함
seeds.rb 는 공용으로 사용할 DB 미리 넣어 둘 파일

```
C:\#BRGG\#BRGG_BAR_TEAM\#barogagi1gibateam>irb  
irb(main):001:0> require './controller.rb'  
=> true  
irb(main):002:0> h = Habit.first  
D, [2018-11-12T00:42:55.062106 #8520] DEBUG -- :   Habit Load (0.5ms)  SELECT "habits".*  
FROM "habits" ORDER BY "id" ASC LIMIT ?  [["LIMIT", 1]]  
=> #<Habit id: 1, mission: "habitname", category: "habitcategory">  
irb(main):003:0> h.category = "category_change_test"  
=> "category_change_test"  
irb(main):004:0> h.save  
D, [2018-11-12T00:43:25.267721 #8520] DEBUG -- :   (0.0ms)  begin transaction  
D, [2018-11-12T00:43:25.276209 #8520] DEBUG -- :   Habit Update (2.4ms)  UPDATE "habits"  
SET "id" = ?  [["category", "category_change_test"], ["id", 1]]  
D, [2018-11-12T00:43:25.286625 #8520] DEBUG -- :   (6.4ms)  commit transaction  
=> true
```

** irb 로 db update 하려면

```
irb  
require './controller.rb'
```

타이핑 후 테스트
/* DB TEST */
DB.save필수!

DB Debugging

Chrome에서 **postman**이라는 확장앱 사용해서 DB값 동작확인
 유저시나리오 대로 확인도 반드시 해야 하지만 예외처리를 위한 다른 값을 넣어보는 작업도 필수 -> controller.rb에 반영

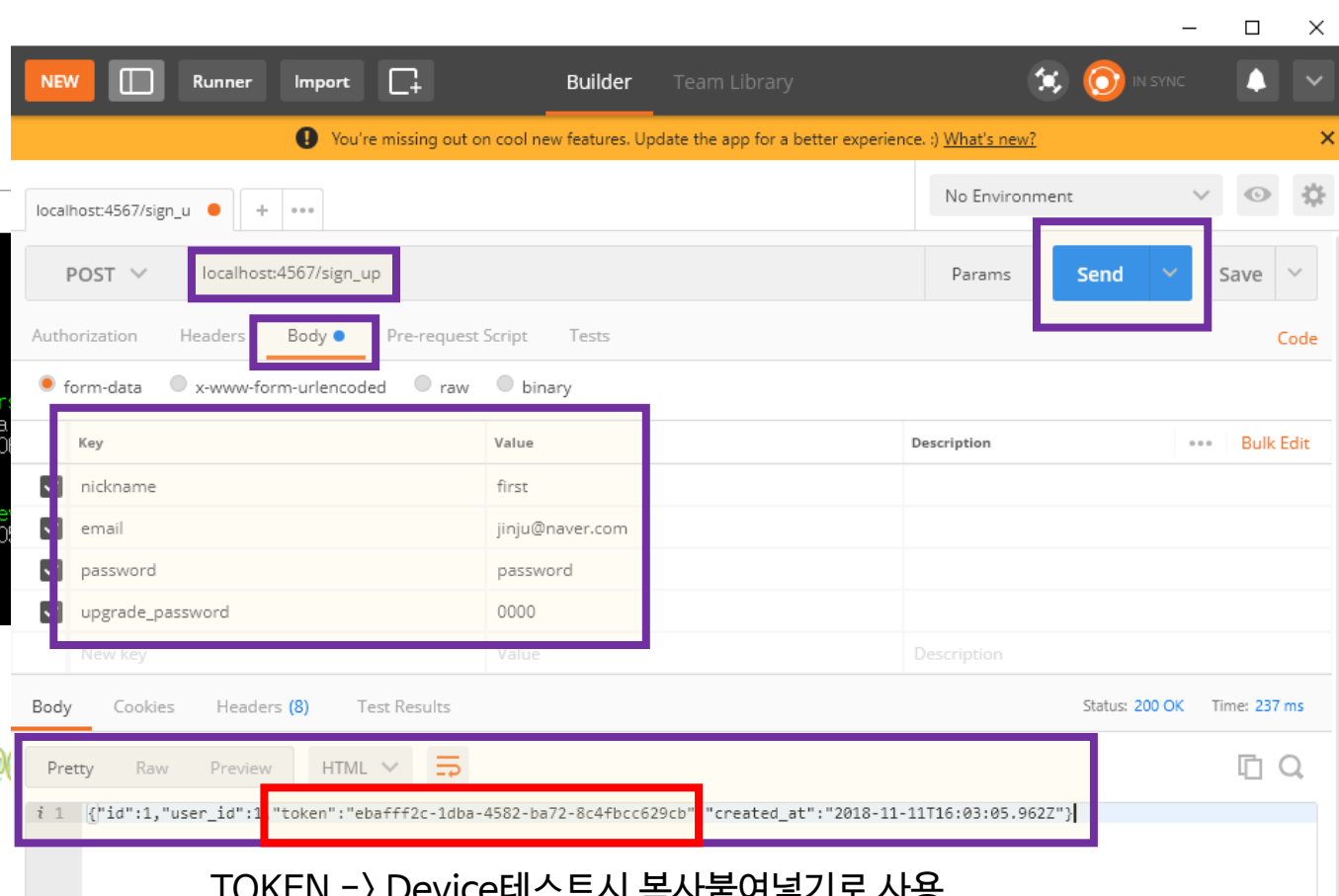
먼저 ruby **controller.rb** 커맨드로 local host 동작시키고
Postman 앱에서 params 값들을 지정
Send!

```
C:\> Start Command Prompt with Ruby - ruby controller.rb

C:\> #BRRGGWBRRGG_BAR_TEAM#barogagilgibateam>ruby controller.rb
[2018-11-12 01:03:03] INFO WEBrick 1.5.7
[2018-11-12 01:03:03] INFO ruby 2.4.4 (2018-03-28) [x64-mingw32]
== Sinatra (v2.0.4) has taken the stage on 4567 for development with backup from WEBrick
[2018-11-12 01:03:03] INFO WEBrick::HTTPServer#start: pid=22368 port=4567
D, [2018-11-12T01:03:05.881134 #22368] DEBUG -- : (0.5ms) begin transaction
D, [2018-11-12T01:03:05.928748 #22368] DEBUG -- : User Create (1.5ms) INSERT INTO "user" ("max_myanimal", "created_at") VALUES (?, ?, ?, ?, ?, ?) [["nickname", "first"], ["email", "0000"], ["current_coin", 0], ["max_myanimal", 3], ["created_at", "2018-11-11 16:03:05.888000"]]
D, [2018-11-12T01:03:05.935693 #22368] DEBUG -- : (5.5ms) commit transaction
D, [2018-11-12T01:03:05.961486 #22368] DEBUG -- : (0.0ms) begin transaction
D, [2018-11-12T01:03:05.964460 #22368] DEBUG -- : Device Create (1.0ms) INSERT INTO "device" ("token", "ebafff2c-1dba-4582-ba72-8c4fbcc629cb"), ["created_at", "2018-11-11 16:03:05.964460"]
D, [2018-11-12T01:03:05.974380 #22368] DEBUG -- : (8.4ms) commit transaction
: 1 - [12/Nov/2018:01:03:05 +0900] "POST /sign_up HTTP/1.1" 200 107 0.1428
: 1 - [12/Nov/2018:01:03:05 대한민국 표준시] "POST /sign_up HTTP/1.1" 200 107
```

아래처럼 controller.rb 에 puts를 넣어서 테스트해볼 수 도 있음

```
puts "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
puts params["upgrade_password"]
puts user.upgrade_password
```



TOKEN -> Device테스트시 복사붙여넣기로 사용

Heroku

나중에 DB도 결국 heroku에서 해야 하기 때문에 Server + Heroku 싱크를 맞추자

Heroku

```
git init
git clone /*repository*/

git add .
git commit - am "message"
git push heroku master
git pull heroku master
```

Heroku DB

```
heroku rake db:migrate
heroku rake db:drop
heroku rake db:seeds
```

Git이랑 비교 -

```
git init
git clone /*repository*/

git add .
git commit - am "message"
git push origin master
```

DB Oriented Development

```
get '/device' #로그인된 사람 가져오기
```

```
  user = USER.  
  device = user.device  
  device.to_json
```

```
end
```

```
post '/device' #로그인
```

```
  user = User.where(id: pwd)  
  device = Device.new  
  device.user = user  
  device.save  
  device.to_json
```

```
end
```

```
delete '/device' #로그아웃
```

```
end
```

```
get '/aminal'
```

```
  # options = params["option"]  
  Aniaml.where(option => params["option"]) GraphQL
```

```
end
```

RESTFUL 방식

: 미팅앱엔 이 방식으로

서버와 클라이언트의 로드를 분산하는 게 좋을 것 같다고 하심

서버에서 DB관련 method를 아래 방식으로 매칭시켜

최대한 로드를 줄임

- GET / POST / PUT / DELETE

대신 클라이언트에서 로직까지 함께 구현해야 함

(DB로직을 모두 알아야함 -> 옵션으로 바로 params전달)

Password 암호화

암호화하는 방식(BCrypt)

```
require 'bcrypt'  
a = BCrypt::Password.create("1234")  
BCrypt::Password.new(a) == "1234"
```

SHA-512

lrb에서 테스트한 내용인데

스트링 “1234”를 a에 저장한 것

이 때 좌항의 new method로 암호화했던 스트링을
또 다시 다른 암호화 방식으로 복호화해서

우항의 이 전에 암호화했던 스트링이랑 또 다시 비교

Fuse 동작방법

준비사항_

Server의 ruby controller.rb 실행

Fuse Project파일 다운받아서 FUSE 프로그램 실행