소프트웨어 개발의 원리 및 실제

Mar. 29

과제 - Django REST 를 이용한 약속 관리 기록 백엔드 서버 개발

사람들 사이의 약속 관리를 기록하는 백엔드 서버를 Django REST 로 구현한다.

0) 환경

권장 사항 : Ubuntu 16.04.3 LTS / Python 3.6.0

user name 은 swpp 여야 함!!

1) **설명**

하나의 약속은 <식별자, 생성날짜, 약속 시작 시간, 약속이 끝나는 시간, 약속 참여자 1, 약속 참여자 2> 로 나타난다. JSON 양식은 아래와 같다.

{

“id” : 식별자(숫자, Django 에서 model 을 정의할 때 자동으로 생성해준다.)

“created” : 생성날짜(DateTime 타입, 약속 생성시 자동으로 채워지도록 설정한다.)

“sinceWhen” : 약속 시작 시간(DateTime 타입)

“tilWhen” : 약속이 끝나는 시간(DateTime 타입)

“user1” : 약속 참여자 1(유저의 id 를 Foregin key 로 저장)

“user2” : 약속 참여자 2(유저의 id 를 Foregin key 로 저장)

}

User1, user2 를 표현할 때에는 Django 에서 기본적으로 제공하는 User 클래스를 활용한다.

DjangoRESTtutorial4(http://www.django-rest-framework.org/tutorial/4-authentication-and-permissions/) 참조

View 를 구현해야 하는 url 은 6 가지이다. 각 Url 에서 구현해야하는 기능은 아래 테이블과 같다.

url / 기능	GET	POST	PUT	DELETE
promises	약속전체 리스트 가져오기	SinceWhen 과 tilWhen 과 user2 를 입력받아 새로운약속 생성	-	-
promises/id	id 의 약속 가져오기	-	SinceWhen 과 tilWhen 을 받아서약속 관계의내용을 수정.	약속 삭제
users	전체 유저 목록	-	-	-
users/id	id 의유저 가져오기	-	-	-
userall	전체 user 를 가져오고 각 user 의 모든 약속들의 id 목록	-	-	-
userall/id	특정 유저만 get	-	-	-

주의 사항

- 임의의 인증되지 않은 유저가 접근해서 새로운 약속관계를 추가할 수 없으며, 인증되지 않은 유저가 기존의 약속관계를 수정 및 삭제할 수 없다. 또한 약속 당사자가 아닌 유저는 promises/id url 에 접근할 수 없다.
- ‘users/’ 와 ‘userall/’ 이 반환해야 할 정보는 다른 형태를 띈다.
    - 'user/’ 에서 나열한 유저 정보는 세부적으로 어떤 약속을 가지고 있는지(자신이

약속을 잡았는지(promises-as-inviter) 혹은 자신이 약속을 잡혔는지(promises-as-invitee)) 를 명시해야한다.

  - ‘userall/’에서 나열한 유저 정보는 자신이 가지고 있는 모든 약속의 id 들을 출력해야 한다.

- promises/ url 의 post method 를 통해서 약속을 생성할때는 sinceWhen 이 tilWhen 보다 이른 시간이어야 하며 같은 id 와의 약속을 만들 수 없다(400 bad request 로 처리).
- promises/id/ url 의 put method 를 통해서 약속 관계를 수정할 때도 sinceWhen 이 tilWhen 보다 이른 시간이어야 한다(400 bad request 로 처리).
- 정확한 형태는 문서 뒤의 스크린샷을 참조하자.

2) **제출 및 채점**

제출은 4 월 8 일 일요일 오후 11 시 59 분까지 swpp@sf.snu.ac.kr 로 제출한다.

제출 파일의 이름은 HW2_SWPP_학번.zip 으로 한다.

(예를 들면, HW2_SWPP_2019-22222.zip)

채점은 자동 채점 방식으로 이루어질 예정이다.

원활한 자동 채점을 위해, 초기 환경은 다음과 같이 세팅한다.

```
swpp@ip-172-32-30-158:~$ pwd
/home/swpp/(이 경로가 중요함.)
swpp@ip-172-32-30-158:~$ virtualenv -p python3 env
...(설치 메세지)
swpp@ip-172-32-30-158:~$ source env/bin/activate
(env) swpp@ip-172-32-30-158:~$ pip3 install django djangorestframework
...(설치 메세지)
(env) swpp@ip-172-32-30-158:~$ django-admin.py startproject homeworktwo
(env) swpp@ip-172-32-30-158:~$ cd homeworktwo/
(env) swpp@ip-172-32-30-158:~/homeworktwo$ python3 manage.py startapp promises
(env) swpp@ip-172-32-30-158:~/homeworktwo$ ls
homeworktwo manage.py promises
(env) swpp@ip-172-32-30-158:~/homeworktwo$
```

제출은 자신의 ~/homeworktwo 디렉토리를 통째로 복사한 뒤, HW2_SWPP_학번 디렉토리에 넣고 압축하여 제출한다.

```
(env) swpp@ip-172-32-30-158:~/HW2_SWPP_2019-22222$ cp -r ../homeworktwo/ .
(env) swpp@ip-172-32-30-158:~/HW2_SWPP_2019-22222$ ls homeworktwo/
homeworktwo manage.py promises ...(다른 파일들)
```

채점에 사용할 스크립트는 다음 주 중에 공지할 예정.

APPENDIX. Django REST 과제 구현 예시.

127.0.0.1:8000/promises	127.0.0.1:8000/promises/id

Promise List

```
GET /promises/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```


```
[
    {
        "id": 1,
        "created": "2018-03-29T03:50:54.352653Z",
        "sinceWhen": "2018-03-28T08:18:36.959885Z",
        "tilWhen": "2019-03-28T08:18:36.959885Z",
        "user1": 1,
        "user2": 2
    },
    {
        "id": 2,
        "created": "2018-03-29T03:51:14.466410Z",
        "sinceWhen": "2020-08-28T08:18:36.959885Z",
        "tilWhen": "2021-03-28T08:18:36.959885Z",
        "user1": 1,
        "user2": 3
    },
    {
        "id": 3,
        "created": "2018-03-29T03:52:24.991506Z",
        "sinceWhen": "2030-01-27T09:05:57.579991Z",
        "tilWhen": "2030-03-22T09:05:57.579991Z",
        "user1": 2,
        "user2": 1
    },
    {
        "id": 4,
        "created": "2018-03-29T03:52:36.765527Z",
        "sinceWhen": "2010-01-27T09:05:57.579991Z",
        "tilWhen": "2012-03-22T09:05:57.579991Z",
        "user1": 2,
        "user2": 3
    },
    {
        "id": 5,
        "created": "2018-03-29T03:53:51.618728Z",
        "sinceWhen": "1999-03-28T08:18:36.959885Z",
        "tilWhen": "2000-03-30T14:35:28.488304Z",
        "user1": 3,
        "user2": 1
    },
    {
        "id": 6,
        "created": "2018-03-29T03:54:22.217951Z",
        "sinceWhen": "1996-03-28T08:18:36.959885Z",
        "tilWhen": "1997-03-30T14:35:28.488304Z",
        "user1": 3,
        "user2": 2
    }
]
```


Promise Detail

```
GET /promises/5/
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```


```
{
    "id": 5,
    "created": "2018-03-29T03:53:51.618728Z",
    "sinceWhen": "1999-03-28T08:18:36.959885Z",
    "tilWhen": "2000-03-30T14:35:28.488304Z",
    "user1": 3,
    "user2": 1
}
```

127.0.0.1:8000/users	127.0.0.1:8000/users/id

User List

```
GET /users/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```


```
[
    {
        "id": 1,
        "username": "aa",
        "promises_as_inviter": [
            1,
            2
        ],
        "promises_as_invitee": [
            3,
            5
        ]
    },
    {
        "id": 2,
        "username": "bb",
        "promises_as_inviter": [
            3,
            4
        ],
        "promises_as_invitee": [
            1,
            6
        ]
    },
    {
        "id": 3,
        "username": "cc",
        "promises_as_inviter": [
            5,
            6
        ],
        "promises_as_invitee": [
            2,
            4
        ]
    }
]
```


User Detail

```
GET /users/3/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```


```
{
    "id": 3,
    "username": "cc",
    "promises_as_inviter": [
        5,
        6
    ],
    "promises_as_invitee": [
        2,
        4
    ]
}
```

127.0.0.1:8000/userall	127.0.0.1:8000/userall/id