[10기 파이썬반] dj-rest-auth 회원 라이브러리 커스터마이징

Django 회원 라이브러리 커스터마이징

- 장고의 회원 관련 라이브러리인 `dj-rest-auth` 및 `allauth` 를 사용할 때 유저 필드 커스터마이징 하는 방법에 대한 문서입니다.
- 라이브러리를 사용하는 동시에 `nickname` 등의 추가 필드를 사용할 때는 필수적으로 알아야 합니다.

준비사항

- 필요 라이브러리 설치

```
$ pip install django djangorestframework dj-rest-auth django-allauth
```

- accounts 앱 생성 및 등록

```
INSTALLED_APPS = [
    # APP
    'accounts',

    # DRF
    'rest_framework',
    'rest_framework.authtoken',

    # REST_AUTH
    'dj_rest_auth',
    'allauth',
    'allauth.account',

    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

- models.py 에 User 작성
  - **아래 필드는 예시입니다!!!**

```
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.
class User(AbstractUser):
    username = models.CharField(max_length=30, unique=True)
    nickname = models.CharField(max_length=255, blank=True, null=True)
    email = models.EmailField(max_length=254, blank=True, null=True)
    age = models.IntegerField(blank=True, null=True)
    money = models.IntegerField(blank=True, null=True)
    salary = models.IntegerField(blank=True, null=True)
    # 리스트 데이터 저장을 위해 Text 형태로 저장
    financial_products = models.TextField(blank=True, null=True)

    # superuser fields
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)

    USERNAME_FIELD = 'username'
```

- settings.py 설정

```
# DRF auth settings
# Token 인증을 기본으로 사용하도록 설정
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ]
}

# 사용자 수정
AUTH_USER_MODEL = 'accounts.User'

# ALL AUTH 수정
# dj-rest-auth 는 email 을 필수적으로 사용하도록 구현되어 있으므로, 해당 사항을 수정
ACCOUNT_EMAIL_REQUIRED = False
ACCOUNT_EMAIL_VERIFICATION = None

# django 인증 시스템에서 사용할 백엔드 클래스 지정
# 기본 인증 백엔드와 allauth 패키지에서 제공하는 인증 백엔드를 모두 사용하겠다는 설정.
AUTHENTICATION_BACKENDS = (
    # django 기본 인증 백엔드
    "django.contrib.auth.backends.ModelBackend",
    # django-allauth 패키지에서 제공하는 인증 백엔드 클래스.
    "allauth.account.auth_backends.AuthenticationBackend",
)

# MIDDLEWARE 에 아래 내용 추가
MIDDLEWARE = [
    ...
```

```
    # Add the account middleware:
    "allauth.account.middleware.AccountMiddleware",
]
```

dj-rest-auth 의 기본 로직

- `rest-auth` 의 기본 url 설정

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    ...
    path('dj-rest-auth/', include('dj_rest_auth.urls')),
    path('dj-rest-auth/registration/', include('dj_rest_auth.registration.urls')),
]
```

- `http://127.0.0.1:8000/dj-rest-auth/` 결과

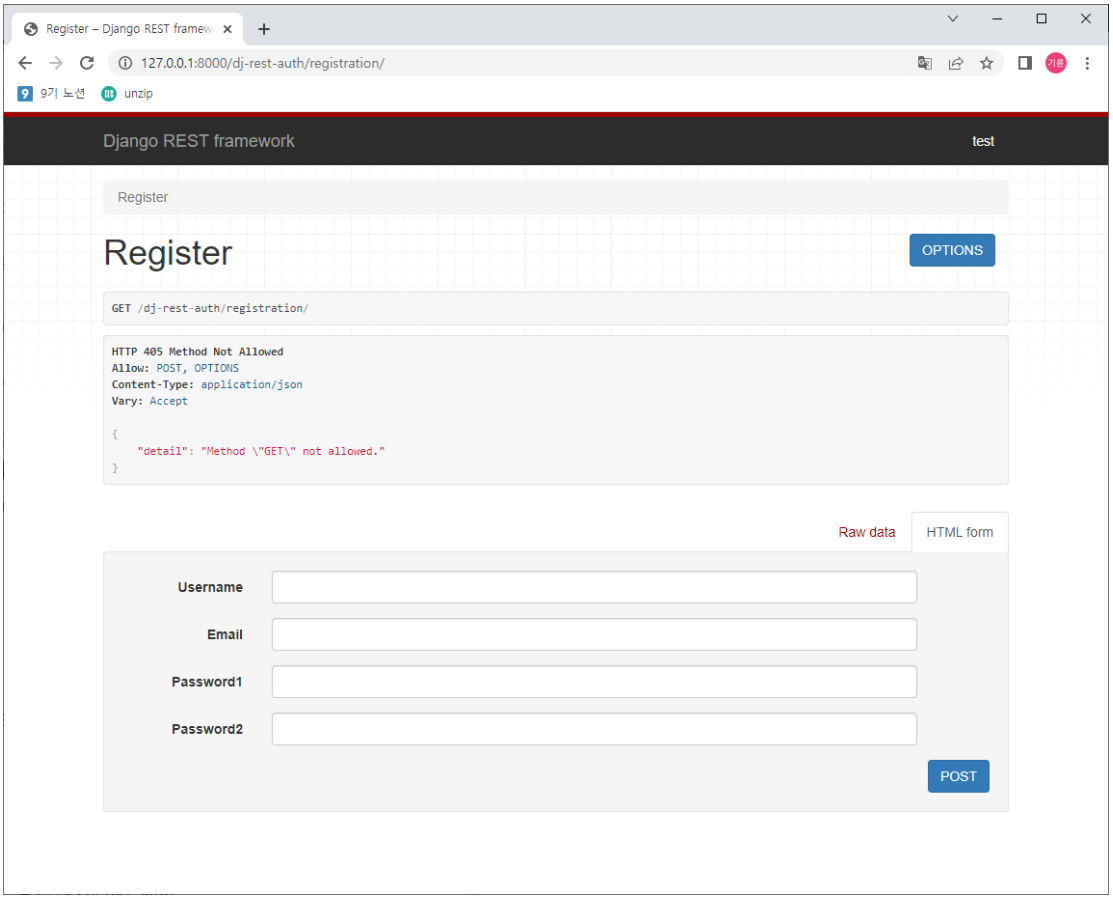
- dj-rest-auth 관련 기능은 다음과 같습니다.
  - 참고사이트 - dj-rest-auth docs

1. `dj-rest-auth/ ^password/reset/$ [name='rest_password_reset']`
   - 패스워드 초기화 (이메일로 전송)
2. `dj-rest-auth/ ^password/reset/confirm/$ [name='rest_password_reset_confirm']`
   - 패스워드 초기화 (이메일 확인 후 초기화 페이지)
3. `dj-rest-auth/ ^login/$ [name='rest_login']`

    - 로그인
4. `dj-rest-auth/ ^logout/$ [name='rest_logout']`
    - 로그아웃
5. `dj-rest-auth/ ^user/$ [name='rest_user_details']`
    - 유저 정보 반환
6. `dj-rest-auth/ ^password/change/$ [name='rest_password_change']`
    - 패스워드 변경
7. `dj-rest-auth/registration/`
    - 회원가입

- 프로젝트에서 사용할 주요 기능은 `3. 로그인`, `4. 로그아웃`, `5. 유저 정보 반환`, `7. 회원가입` 4가지입니다.

- 회원가입
    - `http://127.0.0.1:8000/dj-rest-auth/registeration/` 접속 시
    - 다음과 같이 회원가입 시 입력 받을 필드들이 출력됩니다.
    - 즉, REST 요청 시 아래 필드들에 대한 정보를 보내야 합니다.



models.py 에서 User 커스터마이징만 작성하면, 해당 필드들이 출력되지 않습니다.

위의 필드들을 입력 받도록 하는 기준은 github 코드를 참고해야 합니다.

rest-auth github

- 코드를 직접 살펴보자! ( )
  - github 의 코드 정의 따라가기 기능을 이용하면 훨씬 쉽게 추적이 가능합니다.
- github 의 `dj_rest_auth/registeration/` 부분이 회원가입과 관련이 된 부분입니다.
- urls.py 의 `path('', RegisterView.as_view(), name='rest_register')` 를 확인하면, `RegisterView` 로 연결된 것으로 확인됩니다.

```
# dj-rest-auth/views.py

class RegisterView(CreateAPIView):
    serializer_class = RegisterSerializer
    permission_classes = register_permission_classes()
    token_model = TokenModel
    throttle_scope = 'dj_rest_auth'

    ...
```

- Django 의 CreateAPIView 에서 serializer_class 로 지정한 `RegisterSerializer` 가 **회원가입 시 사용되는 serializer** 입니다.
- `RegisterSerializer` 코드

```
# dj-rest-auth/dj_rest_auth/registration/app_settings.py
from dj_rest_auth.registration.serializers import (
    RegisterSerializer as DefaultRegisterSerializer,
)

serializers = getattr(settings, 'REST_AUTH_REGISTER_SERIALIZERS', {})
RegisterSerializer = import_callable(serializers.get('REGISTER_SERIALIZER', DefaultRegisterSerializer))

# dj-rest-auth/dj_rest_auth/utils.py
def import_callable(path_or_callable):
    if hasattr(path_or_callable, '__call__'):
        return path_or_callable
    else:
        assert isinstance(path_or_callable, str)
        package, attr = path_or_callable.rsplit('.', 1)
        return getattr(import_module(package), attr)
```

- github 에서 따라가기를 해보면 위와 같이 `app_settings.py` 의 코드를 볼 수 있습니다.
  - `import_callable`: path_or_callable 파라미터가 `__call__` 속성을 가지고 있으면 그대로 반환합니다. 가지고 있지 않다면, 패키지에서 모듈을 가져옵니다.

    - `serializers`: settings.py 에 `REST_AUTH_REGISTER_SERIALIZERS` 속성을 가져옵니다. 없다면 비워둡니다.
    - `RegisterSerializer`: 앞의 코드에서 settings.py 에서 읽어왔다면 해당 serializer 를 지정하고, 비어있다면 `DefaultRegisterSerializer` 를 사용합니다.
        - 기본적으로 settings.py 에는 `REST_AUTH_REGISTER_SERIALIZERS` 가 없으니, `DefaultRegisterSerializer` 가 호출됩니다.

- `DefaultRegisterSerializer` 코드

```
class RegisterSerializer(serializers.Serializer):
    username = serializers.CharField(
        max_length=get_username_max_length(),
        min_length=allauth_settings.USERNAME_MIN_LENGTH,
        required=allauth_settings.USERNAME_REQUIRED,
    )
    email = serializers.EmailField(required=allauth_settings.EMAIL_REQUIRED)
    password1 = serializers.CharField(write_only=True)
    password2 = serializers.CharField(write_only=True)

    def validate_username(self, username):
        username = get_adapter().clean_username(username)
        return username

    def validate_email(self, email):
        email = get_adapter().clean_email(email)
        if allauth_settings.UNIQUE_EMAIL:
            if email and email_address_exists(email):
                raise serializers.ValidationError(
                    _('A user is already registered with this e-mail address.'),
                )
        return email

    def validate_password1(self, password):
        return get_adapter().clean_password(password)

    def validate(self, data):
        if data['password1'] != data['password2']:
            raise serializers.ValidationError(_("The two password fields didn't match."))
        return data

    def custom_signup(self, request, user):
        pass

    def get_cleaned_data(self):
        return {
            'username': self.validated_data.get('username', ''),
            'password1': self.validated_data.get('password1', ''),
            'email': self.validated_data.get('email', ''),
        }

    def save(self, request):
        adapter = get_adapter()
        user = adapter.new_user(request)
        self.cleaned_data = self.get_cleaned_data()
        user = adapter.save_user(request, user, self, commit=False)
        if "password1" in self.cleaned_data:
            try:
                adapter.clean_password(self.cleaned_data['password1'], user=user)
            except DjangoValidationError as exc:
                raise serializers.ValidationError(
```

```
                    detail=serializers.as_serializer_error(exc)
                )
            user.save()
            self.custom_signup(request, user)
            setup_user_email(request, user, [])
            return
```

- 기본적으로 `username`, `email`, `password1`, `password2` 4가지 필드가 설정되어 있습니다.
- 또한, 저장 시 사용되는 `save()` 함수와 `get_cleaned_data()` 함수를 보면, 기본적으로 설정된 필드만 저장 시 사용 가능 하다는 것을 확인할 수 있습니다.

따라서, Model 만 재정의 해서는 rest-auth 에서 회원가입 시 재정의 한 필드를 사용할 수 없습니다.

RegisterSerializer 재정의

- accounts/serializer.py 에 다음과 같이 회원가입 시 사용할 serializer 를 재정의합니다.
  - 기존 RegisterSerializer 를 상속받아 재정의
  - 유저 기본 정보 + `nickname`, `age`, `money`, `salary`, `financial_products` 필드를 추가
  - serializer 의 필드와 저장 시 사용되는 `get_cleaned_data` 함수 부분 수정

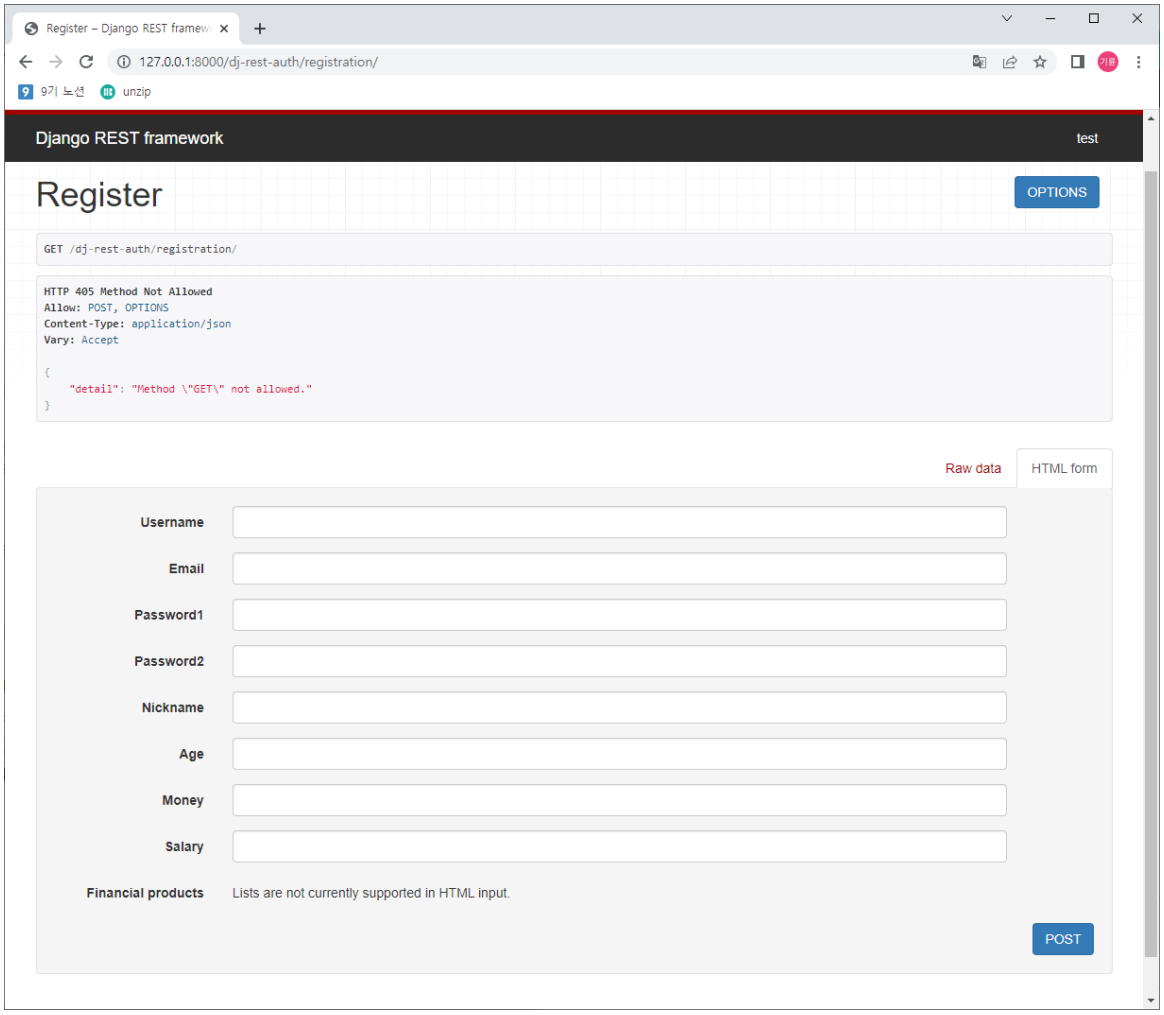
```
from rest_framework import serializers
from allauth.account import app_settings as allauth_settings
from allauth.utils import get_username_max_length
from allauth.account.adapter import get_adapter
from .models import User
from dj_rest_auth.registration.serializers import RegisterSerializer

class CustomRegisterSerializer(RegisterSerializer):
    # 추가할 필드들을 정의합니다.
    nickname = serializers.CharField(
      required=False,
      allow_blank=True,
      max_length=255
    )
    age = serializers.IntegerField(required=False)
    money = serializers.IntegerField(required=False)
    salary = serializers.IntegerField(required=False)
    financial_products = serializers.ListField(child=serializers.IntegerField(), required=False)

    def get_cleaned_data(self):
        return {
            'username': self.validated_data.get('username', ''),
            'password1': self.validated_data.get('password1', ''),
            'nickname': self.validated_data.get('nickname', ''),
            'age': self.validated_data.get('age', ''),
            'money': self.validated_data.get('money', ''),
            'salary': self.validated_data.get('salary', ''),
            'financial_products': self.validated_data.get('financial_products', ''),
        }

    def save(self, request):
      adapter = get_adapter()
      user = adapter.new_user(request)
```

```
        self.cleaned_data = self.get_cleaned_data()
        adapter.save_user(request, user, self)
        self.custom_signup(request, user)
        return user
```

- settings.py 에 rest-auth 가 회원 가입 시 위에서 구현한 serializer 를 호출하도록 설정

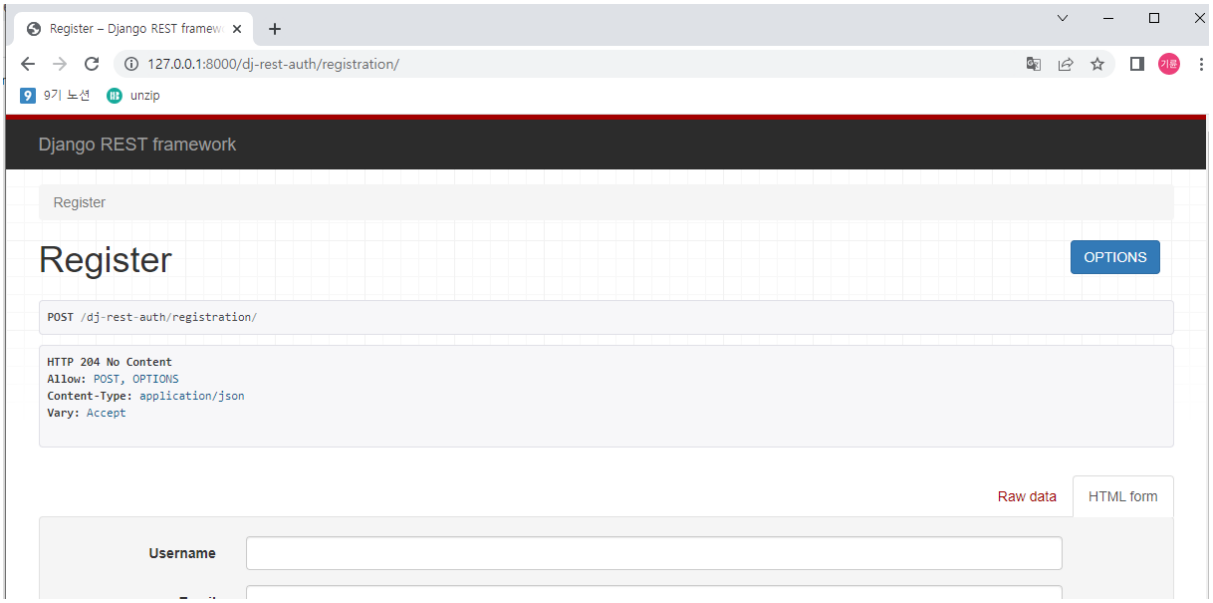
```
# REST-AUTH 회원가입 기본 Serailizer 재정의
REST_AUTH = {
    'REGISTER_SERIALIZER': 'accounts.serializers.CustomRegisterSerializer',
}
```

- 결과
  - 입력 필드가 정상적으로 추가되었습니다.

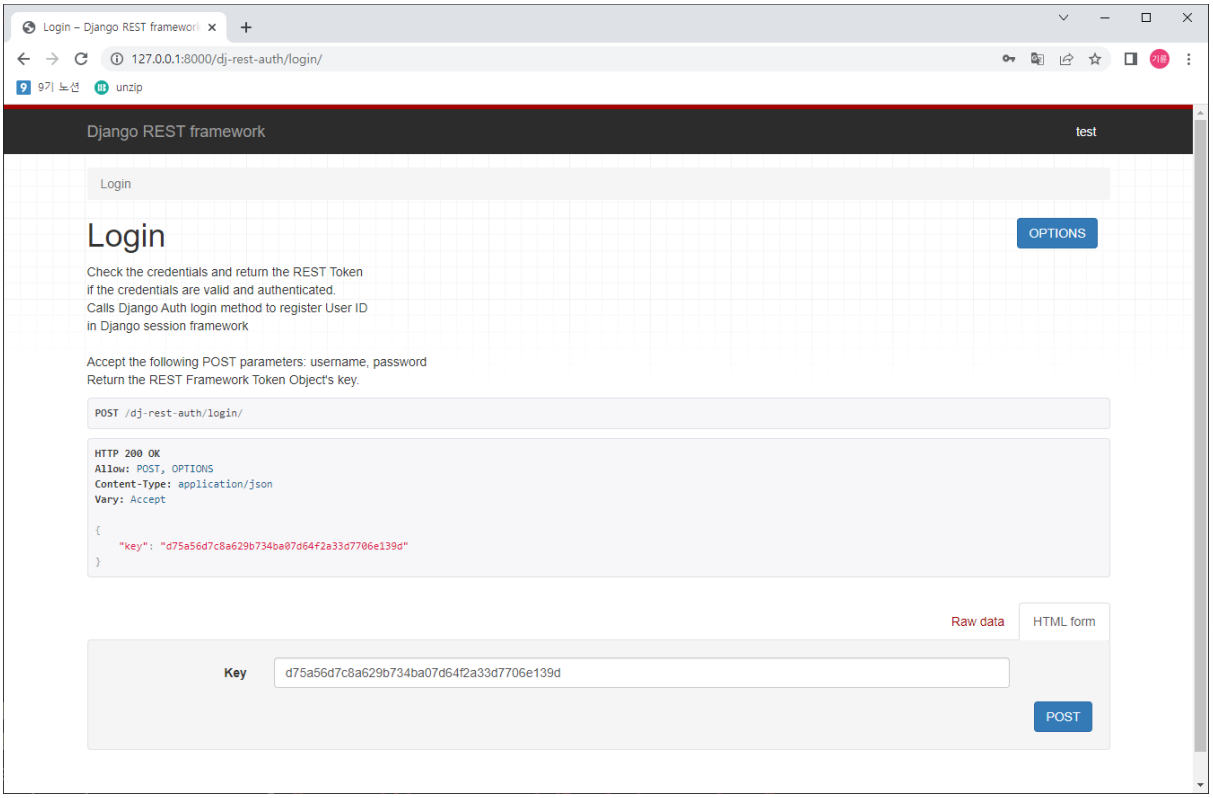


- 모든 데이터를 넣고 회원가입을 시도하면 가입이 진행됩니다.

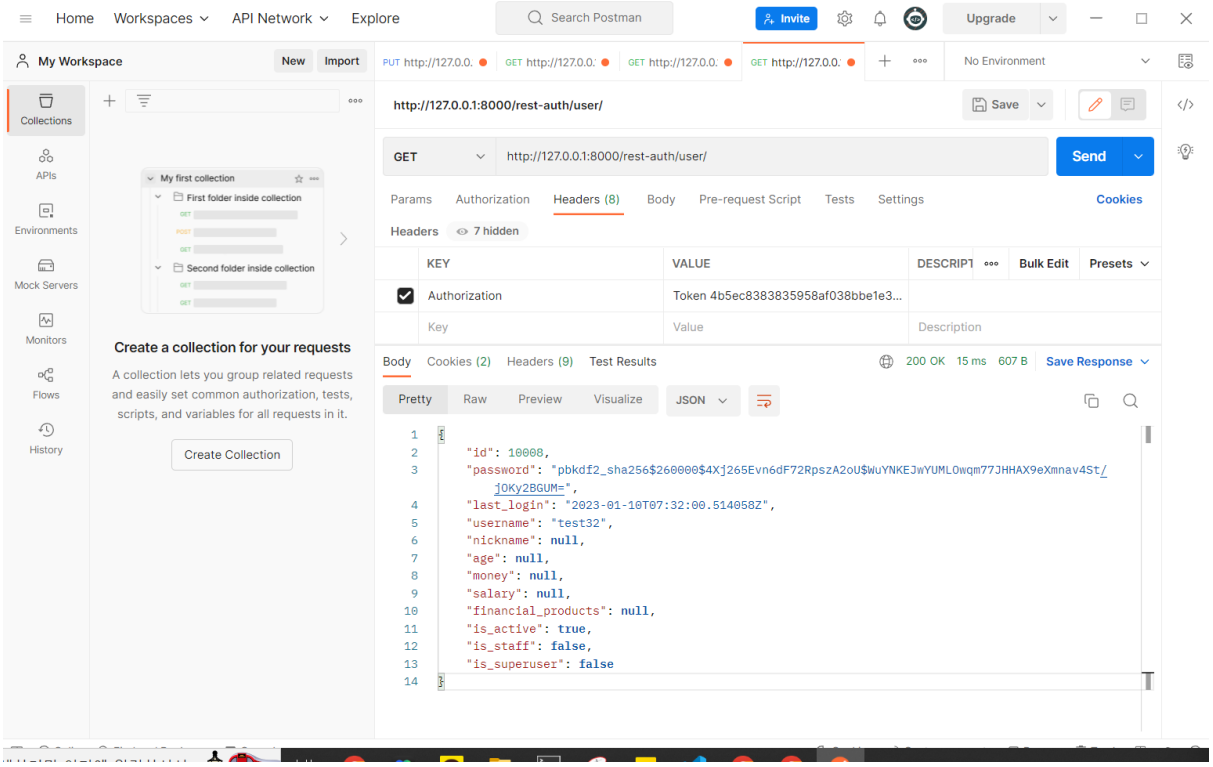
- 이 때, 다음과 같이 `204 STATUS` 를 반환합니다.



- 이는 dj_rest_auth 가 사용하는 `RegisterView` 를 재정의하여 하여 해결 할 수 있으나, 진도 외의 내용이므로 자동 로그인이 아닌 회원 가입 후 추가적으로 로그인 과정을 해주도록 합니다.
- http://127.0.0.1:8000/dj-rest-auth/login/ 주소로 접속하여 로그인을 시도합니다.
  - 로그인 성공 시 아래와 같이 key 를 반환해 줍니다.



- Postman 으로 유저의 데이터를 조회하면, 아래와 같은 내용이 출력됩니다.
  - 경로 : `/dj-rest-auth/user/`
    - Headers 다음과 같은 정보를 추가합니다.
      - KEY : Authorization
      - VALUE : <login 시 전달받은 key>
  - 회원 가입은 되는데, **데이터가 정상적으로 저장되지 않습니다.**



이유는 아래와 같습니다.

allauth 의 adapter

- RegisterSerializer 의 save 함수를 다시 확인해보면 아래와 같습니다.

```
from allauth.account.adapter import get_adapter

...

def save(self, request):
      adapter = get_adapter()
      user = adapter.new_user(request)
      self.cleaned_data = self.get_cleaned_data()
```

```
        adapter.save_user(request, user, self)
        self.custom_signup(request, user)
        return user
```

- `adater = get_adapter()` 코드를 확인할 수 있습니다.
  - `get_adapter` 코드를 따라가면 `allauth` 의 코드가 나옵니다.
    - dj-rest-auth 에서 allauth 를 사용하는 것을 확인할 수 있습니다.
    - allauth github

```
# django-allauth/allauth/account/adapter.py

def get_adapter(request=None):
    return import_attribute(app_settings.ADAPTER)(request)
```

- `app_settings.ADAPTER` 를 github 를 이용하여 따라가보면, 다음 코드를 볼 수 있습니다.

```
# django-allauth/allauth/account/app_settings.py

@property
def ADAPTER(self):
    return self._setting("ADAPTER", "allauth.account.adapter.DefaultAccountAdapter")
```

- 기본 값으로 `DefaultAccountAdapter` 를 사용한다는 것을 알 수 있습니다.
- `DefaultAccountAdapter` 코드

```
# django-allauth/allauth/account/adapter.py

class DefaultAccountAdapter(object):
    ...

    def save_user(self, request, user, form, commit=True):
        """
        Saves a new `User` instance using information provided in the
        signup form.
        """
        from .utils import user_email, user_field, user_username

        data = form.cleaned_data
        first_name = data.get("first_name")
        last_name = data.get("last_name")
        email = data.get("email")
        username = data.get("username")
        user_email(user, email)
        user_username(user, username)
        if first_name:
            user_field(user, "first_name", first_name)
        if last_name:
            user_field(user, "last_name", last_name)
        if "password1" in data:
            user.set_password(data["password1"])
        else:
```

```
            user.set_unusable_password()
        self.populate_username(request, user)
        if commit:
            # Ability not to commit makes it easier to derive from
            # this adapter by adding
            user.save()
        return
```

- 위 코드를 보면 저장 시 사용하는 `save_user` 함수에서도 필드들이 고정이 되어있습니다.
- 여기서 **Adapter 또한 커스터마이징을 해야한다는 것을 알 수 있습니다.**

Adapter 커스터마이징

- 기본적으로 사용하는 `DefaultAccountAdapter` 를 상속받아, 우리가 수정이 필요한 `save_user` 함수만 오버라이딩 하면 쉽게 사용할 수 있습니다.

```
# accounts/models.py

from allauth.account.adapter import DefaultAccountAdapter

class CustomAccountAdapter(DefaultAccountAdapter):
    def save_user(self, request, user, form, commit=True):
        """
        Saves a new `User` instance using information provided in the
        signup form.
        """
        from allauth.account.utils import user_email, user_field, user_username

        # 기존 코드를 참고하여 새로운 필드들을 작성해줍니다.
        data = form.cleaned_data
        first_name = data.get("first_name")
        last_name = data.get("last_name")
        email = data.get("email")
        username = data.get("username")
        nickname = data.get("nickname")
        age = data.get("age")
        money = data.get("money")
        salary = data.get("salary")
        financial_product = data.get("financial_products")

        user_email(user, email)
        user_username(user, username)
        if first_name:
            user_field(user, "first_name", first_name)
        if last_name:
            user_field(user, "last_name", last_name)
        if nickname:
            user_field(user, "nickname", nickname)
        if age:
            user.age = age
        if money:
            user.money = money
        if salary:
            user.salary = salary
        if financial_product:
            financial_products = user.financial_products.split(',')
            financial_products.append(financial_product)
```

```
            if len(financial_products) > 1:
                financial_products = ','.join(financial_products)
            user_field(user, "financial_products", financial_products)
        if "password1" in data:
            user.set_password(data["password1"])
        else:
            user.set_unusable_password()
        self.populate_username(request, user)
        if commit:
            # Ability not to commit makes it easier to derive from
            # this adapter by adding
            user.save()
        return user
```

- 추가적인 입력이 필요한 필드들을 추가해줍니다.
- `financial_products`의 경우 문자열 입력을 받아 리스트 형태로 저장을 해야 되기 때문에, 저장하기 전에 리스트로 바꿔주는 작업을 추가해줍니다.

- 회원 가입 시 사용하는 `allauth.account.adapter.DefaultAccountAdapter` 를 위에서 구현한 adapter 로 설정해주어야 합니다.
    - settings.py 에 아래 내용을 추가한다.

```
ACCOUNT_ADAPTER  = 'accounts.models.CustomAccountAdapter'
```