-auth의 views에 PasswordChange와 관련된 View,Form이 있다 이를 사용해 비밀 번호 변경 페이지 구현가능

-포스팅 쓰기는 instagram app에서 구현(생성함), 사진 한장과 caption,tag 만들기

-HTML에서 url reverse사용시 {% url '앱이름:view 이름 % '} 이렇게 하면 되는듯?

-Model을 만들고 나면 Admin.py에서 @admin.register를 사용해 admin 단에서 확인이 가능하다.

-re.findall(r"#[a-zA-Z\dㄱ-힇]+", post.caption) 이거 정규 표현식인데 #이들어간 a-z 즉 모든 알파뱃과 한글을 post.caption에서 가져온다란 의미이다.

===

<User Page 만들기>

-이미지를 동그랗게 만들기 위해 getbootstarp.com의 문서에서 검색 가능

-img의 class에 "rounded-circle" 추가하면된다.

-User Page의 핵심 ==> url이다 정규 표현식을 통해 username과 관련된 url이 들어와야 이동시키게 하는것이 중요하다.

re\_path(r'^(?P<username>[\w.@+-]+)/$',views.user\_page, name='user\_page') 이렇게 하면된다.

-중요한거는 $ 부분인데 달러표시를 해줘야 username만 들어왔을때 동작을 하게 한다.

======================

-API 서버? ==>앱/웹 서비스를 만드는 개발자들이 이용하는 data 위주의 서비스 시간이 지나도 호환성 유지가 필요로 한다.

데이터는 주로 json을 사용하며(모든건 x) 웹 서비스를 이용하는 경우엔 항상 최신버전을 사용

-REST==> Representational State Transfer의 약자로 대표 상태 전달? 이란 느낌으로 해석하면 된다.

아키텍처 스타일을 REST라고 부르며, 프로토콜에 독립적이다. 일반적인 REST 구현에서는 HTTP를 사용!

-RESTful API 디자인 원칙(by MS Azure 아키텍처 디자인)

1)리소스 중심으로 디자인(리소스? ==> 장고 모델 같은것)

2)클라이언트에서 액세스할 수 있는 모든 종류의 개체/서비스가 리소스에 포함

3)리소스마다 해당 리소스를 고유하게 식별하는 식별자

4)요청/응답 format으로 주로 json을 사용한다

5)균일한 인터페이스를 적용, 리소스 표준 HTTP 동사(GET,PUT등)를 사용한다

-리소스 중심으로 API 구성! 심플한 URI! 가중요하다

-기본 HTTP Method 기준으로 작업의 정의?

1)GET: 리소스의 표현, 응답 본문의 리소스의 세부정보를 보여준다

2)POST: 새 리소스 생성 요청, 응답 본문에 새 리소스 세부정보를 제공(멱등성 미보장)

즉 POST하면서 세부정보 제공 필요!

3)PUT: 기본 리소스를 대체(업데이트 개념), 요청 본문에 갱싱할 정보 제공(멱등성 보장되야됨)

멱등성?==>몇번을 하든 서버상에서는 같은 결과가 보장되는 개념?

4)PATCH: 기본 리소스를 부분 대체한다. (멱등성 미보장)

5)DELETE: 지정 리소스 제거

-Content-Type 해더:요청시 처리를 원하는 형식 지정하면 서버에서는 형식대로 응답한다. 서버측에서 형식을 지원하지 않으면 415 코드 반환

-HTTP METHOD 별 다양한 상태 코드

1)GET: 성공시 200, 리소스 못찾으면 404

2)POST: 성공시201응답, 새 리소스의 URI는 응답의 Location 해더로, 200,204도 반환할 수 있다 잘못된

데이터 요청시 400에러!

3)PUT: 기존 리소스를 업데이트할 경우 200(OK)또는 204(내용없음) 반환, 상황에 따라 업데이트할 수 없는 경우 409 충돌 반환

4)DELETE: 성공시 204응답(내용없음) 리소스가 없는 경우 404 응답

5)비동기 작업: 작업 완료에 시간이 오래 걸릴 경우 다른 TASK Queue를 통해 비동기 처리 할 수 있다. 이 때 요청은 수락되었지만 완료되지 않았는 경우 202 응답(수락됨 응답)

-Django-Rest\_framework: 장고가 아닌 별도의 third-party 애플리케이션이다. 즉 장고 API에서 널리 쓰여지기 때문에공부하도록 한다!

즉 REST API 컨셉을 만들 수 있게 도와주는게 DRF이다.

<DRF의 주요 기능들>

1)Serializer/ModelSerializer를 통한 데이터 유효성 검증및 데이터 직렬화

2)각종 Parser를 통한 데이터 처리

3)APIView/Generic/ViewSet/ModelViewSets를 통한 요청 처리

4)Renderer를 통한 응답 포맷 지원

5)인증 및 권한 체계==> JWT지원

6)Thorottling: 최대 호풀 횟수 제한

-CRUD란? ==> C는 Create(새 레코드 생성), R은 Read/Rerieve(레코드 목록 조회), U는 Update(특정 레코드 수정), D는 Delete(특정 레코드 삭제)란 뜻이다

CRUD는 리소스에 대한 대표적 동작일뿐 API의 전부는 아니다.

=====

-한 post model에 대한 API 서비스 제공시 URL 설계?

1)새 포스팅 내용을 받아 등록하고 확인 응답-->/post/new/ 주소로 POST 요청

2)포스팅 목록 및 검색응답-->/post/ 주소로 GET 요청

3)10번 포스팅 내용 응답--> /post/10 주소로 GET 요청

-REST API로 설계시 5개의 뷰 처리가 필요로 한다.

컨셉 코드?

-view.py에서 JsonResponse를 통해 응답을 준다(render가 아니고!)

===============

<다양한 HTTP 클라이언트 프로그램>

-GUI로는 postman(Powerful API Client)가 있고 CLI 프로그램은 HTTPie, cURL이 있다. 라이브러리는 request를 사용한다.

-여기서는 HTTPie를 사용한다. pip로 다운 받고 'http (원하는 url)'을 입력하면 해당 url의 JSON응답을 받는다.

-HTTP 테스트용 서버는 httpbin.org이다!

-JSON으로는 POST를 받을 수 없고 body에 unicode로 메시지가 전달되고 --form을 사용하면 request.post에 메시지가 들어온다.

=============

<JSON 직렬화>

-직렬화란? ==>프로그래밍 언어의 통신에서 데이터는 필히 문자열로 표현되어야함

즉 직렬화란 송신자가 객체를 문자열로 변환하는 작업을 말한다(Serialize)/ 수신자는 비직렬화를 한다.

따라서 직렬화에 대한 비 직렬화가 있어야 한다!

-공통으로 지원하는 직렬화 포맷은 JSON,XML이 있고 특정 언어에서만 지원하는 포맷이 있다(파이썬은 Pickle이 있다)

-보통의 웹 세상에서는 GET에 대해 HTML 포멧으로 응답, POST에 대해서는 application/x-www-form-urlencoded 인코딩 또는 multipart/form-data 인코딩으로 요청하고 HTML 포맷으로 응답

API 서버에서는 json 인코딩된 요청/응답을 한다!

-JSON은 **다른 언어/플랫폼과 통신할때 주로 사용한다! (자바랑 통신도 가능)**, 표준 라이브러리 json제공을 한다(즉 파이썬 기본 데이터 타입에 대해서는 지원하지만 장고 모델or 우리가 직접 만든 model에 대해서는 직렬화를 직접 커스텀 해야된다. 여기가 중요한 부분임)

JSON을 import하고 **json.dumps()**로 직렬화 한다

-pickle에서는 pickle.dumps로 하면 된다. (저장 타입은 Byte 타입이다.)

-핵심은 말했듯이 **JSON/Pickle 둘 다 “파이썬” 라이브러리지 “Django”라이브러리가 아니기 때문에 추가적인 Rule 설정이 필요로 하다.**

-추가적인 Rule은 DjangoJSONEncoder를 import해주면 된다. (from Django.core.serializers.json) 하지만 이 경우도 오류가 발생할 수 있다

🡺해결 방법 중 제일 간단한 거는 그냥 직접 변환 때리는 방법이 있다.

Data=[{‘id’: post.id, ‘title’: post.title, ‘content’: post.content} for post in Post.objects.all()] 이렇게!

-json.dumps의 **ensure\_ascii** 옵션, 기본은 True 값이고 이는 한글 문자열을 unicode로 바꿔준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

🡪\\u라는글자가 유니코드라는 뜻

-근데 위의 방식대로 직접 하는 것은 매우 귀찮다. 따라서 **나만의 Rule을 class로 커스텀하는 방법**이 좋다.

1)DjangoJSONEncoder를 상속받는 클래스 만들기

2) default라는 함수 하나만들고 obj 인자를 받는다 obj인자는 객체를 어떻게 변환하는지에 관한 인자이다.

텍스트이(가) 표시된 사진

자동 생성된 설명 <예시>

위의 예시에서 if문으로 obj가 QuerySet이면 튜플로 바꾸고 재귀적으로 다시 올려 보내다. 뭐 이런 느낌이다 if문은 내 마음대로 커스텀 할 수 있다.

-\*\*최종적으로 **json.dumps(직렬화 할 내용, cls=”Json 엔코더”, ensure\_ascii=)** 이렇게 구현한다

<DRF에서 지원하는 **rest\_framework.renderer.JSONRender**>

-장고의 DjangoJSONEncoder를 상속받지 않고 기본 json.JSONEncoder를 상속받아 구현한다.

-**Model 타입은 미지원**🡪ModelSerializer를 통해 변환해라

<DRF에서 지원하는 **rest\_framework.renderer.JSONRenderer**>(er이 붙음)

-json.dumps에 대한 래핑 클래스로 보다 편리한 직렬화 지원

-utf-8인코딩도 추가 수행

data = Post.objects.all() JSONRenderer().render(data) 이렇게 사용

-당연히 모델에 대한 변환은 되지 않는다.

**<장고 Form&ModleForm VS DRF Serializer/ModelSerializer>**

-장고의 form과 DRF의 Serializer는 서로 유사하다. 코드 구성도 그렇고 둘 다 Form 필드 지정 또는 Model로부터 데이터를 읽어 오기 때문+ 유효성 검사 및 획득도 한다.

-다만 차이점은 장고는 Form 태그가 포함된 HTML을 생성하지만 DRF는 Form데이터가 포함된 JSON 문자열을 생성한다.

-ModelSerializer를 사용할 때 Post.objects.all()처럼 쿼리 set을 넘겨준다면 Many=True 옵션을 넣어주고 하나의 쿼리만 넘겨준다면 그냥 사용해도 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 사진이 예시다

-여기서 .data의 타입은 **ReturnDict** 타입이다.

<ReturnDict 타입?>

-Serializer.data의 데이터 타입으로 OrderedDict를 상속받으며 serializer를 키워드 인자로 삼는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

-위처럼 만들어진 serializer.data는 json.dumps or JSONRenderer로 직렬화가 가능하다. WHY? 🡺객체이기 때문에! 파이썬 기본 자료 구조이다.

**<장고 View에서의 JSON 응답>**

-View는 기본적으로 **HttpResponse 타입의 응답을 필요로 한다**. 2가지 방법이 있다

1)직접 json.dumps를 통해 직렬화 된 문자열을 획득하여 HttpResponse를 통해 응답

2)1번을 정리한 **JsonResponse** 지원🡪내부적으로 json.dumps를 사용해 DjangoJSONEncoder가 디폴트 지정되어 있다.

즉 2번을 사용하면 DjangoJSONEncoder에서만 지원하는 옵션을 위주로 사용해라

-DjangoJSONEncoder를 상속받은 **커스텀 Rule 인코더로도 Queryset을 직렬화 가능**하다.

**<DRF를 통한 JSON 응답>**

-DRF Response 활용하는 방법

From res\_framework.response import Response를 import해 Response(serializer.data)로 JSON 직렬화가 가능하다.



-Response와 APIView: 모든 DRF의 뷰는 APIView를 상속받고 APIView를 통해 Response에 다양한 속성이 지정됨 🡺Response는 API View를 달고 산다.

**<실제 활용 예제>**

-간결하게 사용한다

From rest\_framework import **generics**

Class PostListAPIView(generics.**ListAPIView**): #이건 List기능만 지원한다.

Qs=post.objects.all()

Serializer\_class=PostModelSerializer

Post\_list=PostListAPIView.as\_view()

-포스팅 조회 응답에 username을 포함시키려면?

1) serializer.py에 가서serializers.ReadOnlyField의 source=’author.username’ 옵션을 추가 당연히 models에 author는 추가해주기

2)중첩된 serializer를 통해 구현하기

텍스트이(가) 표시된 사진

자동 생성된 설명