💛

Django Study

Tutorial1 - Serialization

Links

- Pygments : https://pygments.org/faq/
- **설명을 위한 도표**
- **용어 정리**
- Serializer.py

SQL related shell commands

🐬 SQL blah blah blah

Result Preview

`localhost:8000/snippets/`

```
(env) C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization\tutor
ial>http http://127.0.0.1:8000/snippets/
HTTP/1.1 200 OK
Content-Length: 352
Content-Type: application/json
Date: Sun, 12 Apr 2020 06:58:38 GMT
Server: WSGIServer/0.2 CPython/3.8.2
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

[
    {
        "code": "foo = \"bar\"\n",
        "id": 1,
        "language": "python",
        "linenos": false,
        "style": "friendly",
        "title": ""
    },
    {
        "code": "print(\"hello world\")\n",
        "id": 2,
        "language": "python",
        "linenos": false,
        "style": "friendly",
        "title": ""
    },
    {
        "code": "print(\"hello world\")",
        "id": 3,
        "language": "python",
        "linenos": false,
        "style": "friendly",
        "title": ""
    }
]
```

`localhost:8000/snippets/1/`

```
(env) C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization\tutor
ial>http http://127.0.0.1:8000/snippets/1/
HTTP/1.1 200 OK
Content-Length: 110
Content-Type: application/json
Date: Sun, 12 Apr 2020 06:59:06 GMT
Server: WSGIServer/0.2 CPython/3.8.2
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

{
    "code": "foo = \"bar\"\n",
    "id": 1,
    "language": "python",
    "linenos": false,
    "style": "friendly",
    "title": ""
}
```

Table of Contents

---

1. Setting up a new environment

프로젝트를 진행할 폴더를 만든다.

```
mkdir tutorial1-serialization
cd tutorial1-serialization
```

venv 를 사용해 가상 환경을 만든다.

```
python -m venv env
```

env 폴더 → Scripts 폴더 혹은 bin 폴더→ activate.bat 을 실행시켜 가상 환경을 기동한다.

📌 (Windows 의 경우)

```
env\Scripts\activate.bat
```

📌 (Mac OS 혹은 UNIX 의 경우)

```
source env/bin/activate
```

정상적으로 실행되었다면 왼쪽에 (env) 표시가 나타난다.

```
C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization>env\Scripts\activate.bat

(env) C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization>
```

가상 환경 안에서 필요한 패키지들을 다운받는다.

```
pip install django
pip install djangorestframework
pip install pygments
```

→ `pygments` 는 이번 투토리얼에서 사용하게될 파이썬 엔진 중 하나이다. 코드 하이라이팅을 쉽게 해주는 여러 클래스들을 제공해준다.

📌가상 환경을 종료할 경우 `env\Scripts\deactivate.bat` 을 실행해주면 된다.

```
#혹은 그냥
deactivate
#을 실행해 종료한다.
```

---

2. Getting Started

새로운 디장고 프로젝트를 생성한다.

```
django-admin startproject tutorial #tutorial 은 그냥 폴더 이름.

cd tutorial
```

→ 예시 코드의 import 문을 수정하고 싶지 않다면 폴더 이름을 tutorial 로 하는 것을 추천.

📌 현재 폴더에 프로젝트를 만들고 싶다면 맨 **뒤에** `.` **을 찍어준다.**

```
django-admin startproject tutorial .
```

여기까지 실행하면 다음과 같은 파일들이 tutorial 폴더 안에 생성될 것이다.

```
(env) C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization\tutorial>ls -al
total 137
drwx------+ 1 sarah sarah      0 Apr 12 15:33 .
drwx------+ 1 sarah sarah      0 Apr 12 15:19 ..
-rwx------+ 1 sarah sarah 135168 Apr 12 15:33 db.sqlite3
-rwx------+ 1 sarah sarah    649 Apr 12 15:19 manage.py
drwx------+ 1 sarah sarah      0 Apr 12 15:45 snippets
drwx------+ 1 sarah sarah      0 Apr 12 15:20 tutorial
```

프로젝트 폴더 안에 투토리얼에서 만들 **앱**을 또 생성해줘야 한다.

```
python manage.py startapp snippets #snippets 는 앱 이름
```

현재 우리는 tutorial 이라는 폴더 안에 있다. 그 밑에 자동으로 tutorial 이라는 또 다른 폴더가 생성되어 있을 것이다. 해당 폴더로 들어가 그 안의 `settings.py` 파일을 보면 아래와 같이 `INSTALLED_APPS` 가 보일것이다. 다음 두 가지 앱들을 추가해주자.

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'snippets.apps.SnippetsConfig',
]
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'snippets.apps.SnippetsConfig',
]
```

---

3. Creating a model to work with

`Snippet` 이라는 모델 클래스를 작성한다. `snippets/models.py` 파일 안에 들어가서 아래의 코드를 붙여넣자.

📌 **모델 클래스 : 데이터베이스의 테이블을 정의한다고 생각하면 쉽다.**

📌 Object Relational Mapping?

https://wayhome25.github.io/django/2017/04/01/django-ep9-crud/

```
from django.db import models
from pygments.lexers import get_all_lexers
from pygments.styles import get_all_styles

LEXERS = [item for item in get_all_lexers() if item[1]]
LANGUAGE_CHOICES = sorted([(item[1][0], item[0]) for item in LEXERS])
STYLE_CHOICES = sorted([(item, item) for item in get_all_styles()])

#Model 로 만듦으로써 장고는 이 객체가 DB에 저장될 객체라는 것을 알 수 있다.
class Snippet(models.Model):
    created = models.DateTimeField(auto_now_add=True)
    title = models.CharField(max_length=100, blank=True, default='')
    code = models.TextField()
    linenos = models.BooleanField(default=False)
    language = models.CharField(choices=LANGUAGE_CHOICES, default='python', max_length=100)
    style = models.CharField(choices=STYLE_CHOICES, default='friendly', max_length=100)

    class Meta:
        ordering = ['created']
```

- class Meta:

  ordering = ['created'] #생성된 시점 순서대로 정렬해서 저장한다는 뜻.
- Django 의 Meta Options
- ORM ordering vs Model "Meta" ordering : 요약하면 객체가 항상 정렬되어 보관되어야 하는 경우는 Meta 오더링을 사용한다. 하지만 DB 에서 정렬 작업은 많은 비용이 들기 때문에 꼭 필요할때만 사용하자. 추후에 정렬을 해도 되는 경우 ORM 의 `Model.objects.order_by("user")` 메소드를 사용해 정렬하도록 하자.
- **Options.ordering** Documentation

새로운 모델을 생성하였으니 이에 맞는 DB 가 필요하다. 데이터베이스를 sync 해주자.

```
python manage.py makemigrations snippets #현재 앱에 대한 migration 수행
python manage.py migrate #모든 앱들에 걸쳐서 migration 수행
```

→ 이렇게 해주면 상위 폴더 밑에 dp.sqlite3 파일이 자동으로 생성될 것이다.

4. Creating a Serializer class

가장 처음 Web API 를 만드는 데 있어서 해야 할 것은 위의 모델 객체를 (Snippet instances) 직렬화, 역직렬화 하는 수단을 만드는 것이다. (영어로는 Serialization, Deserialization)

클라이언트는 모델 객체에 대한 정보들을 JSON 혹은 XML로 반환받아야 한다. 따라서 모델 객체를 파이썬의 native datatype 으로 직렬화 하여 JSON 혹은 XML로 만든 후 클라이언트에 제공한다.

`snippets/serializers.py` 를 만들자

```
cd snippets
touch serializers.py
```

`serializers.py` 파일 안에 들어가서 아래의 코드를 붙여넣자.

```
from rest_framework import serializers
from snippets.models import Snippet, LANGUAGE_CHOICES, STYLE_CHOICES


class SnippetSerializer(serializers.Serializer):
    id = serializers.IntegerField(read_only=True)
    title = serializers.CharField(required=False, allow_blank=True, max_length=100)
    code = serializers.CharField(style={'base_template': 'textarea.html'})
    linenos = serializers.BooleanField(required=False)
    language = serializers.ChoiceField(choices=LANGUAGE_CHOICES, default='python')
    style = serializers.ChoiceField(choices=STYLE_CHOICES, default='friendly')

    def create(self, validated_data):
        """
        Create and return a new `Snippet` instance, given the validated data.
        """
        return Snippet.objects.create(**validated_data)

    def update(self, instance, validated_data):
        """
        Update and return an existing `Snippet` instance, given the validated data.
        """
        instance.title = validated_data.get('title', instance.title)
        instance.code = validated_data.get('code', instance.code)
        instance.linenos = validated_data.get('linenos', instance.linenos)
        instance.language = validated_data.get('language', instance.language)
        instance.style = validated_data.get('style', instance.style)
        instance.save()
        return instance
```

- `create()` 와 `update()` : `serializer.save()` 를 호출했을 때 객체들이 생성, 업데이트 되는 시점에 호출되는 함수들이다.
- 추후에 위의 `class SnippetSerializer(serializers.Serializer):` 대신 `ModelSerializer` 를 활용하여 클래스를 단축할 것이다.
- **data vs. validated data**

  ⇒ serializer 클래스를 통해서 확인하기.

(Github 코드에 들어가서 save() 함수 구현부의 validated_data 를 찾아보자)

❓ I AM A PYTHON NEWBIE ❓

`**` + `변수` **의 의미** : 함수 호출 시 `**validated_data` 라고 호출한 것은 딕셔너리를 키워드 매개변수로 unpack 하는 것을 뜻한다.

예를들어

```
>>> dic={'a': 10, 'b':20}
>>> functionA(**dic)  #it is similar to functionA(a=10, b=20)
```

→ https://stackoverflow.com/questions/11315010/what-do-and-before-a-variable-name-mean-in-a-function-signature 참고

---

5. Working with Serializers

위에서 우리가 만든 Snippet 클래스의 실제 객체들을 만들고 이 객체들을 Serialize 할 것이다.

`snippets` 폴더 아래에다가 아래의 코드를 복사해서 `samples.py` 라는 파일로 저장하자.

Snippet 객체들을 만들고 저장하는 코드이다.

```
from snippets.models import Snippet
from snippets.serializers import SnippetSerializer
from rest_framework.renderers import JSONRenderer
from rest_framework.parsers import JSONParser

snippet = Snippet(code='foo = "bar"\n')
snippet.save() #더미 데이터 저장

snippet = Snippet(code='print("hello, world")\n')
snippet.save() #더미 데이터 저장

serializer = SnippetSerializer(snippet) #직렬화 한 후

content = JSONRenderer().render(serializer.data) #JSON 객체로 만들고

print(serializer.data) #데이터를 확인한다
print(content) #JSON 으로 렌더링 된 결과를 확인한다
```

`tutorial` 폴더 아래로 경로를 이동한 후 아래의 커맨드를 쳐서 스크립트를 실행해보자.

```
C: ...\tutorial> python manage.py shell < snippets\samples.py
```

위의 print 문 안에 있는 내용들이 출력되어 나오는 것을 확인할 수 있다.

```
(env) C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization\tutorial>python mana
ge.py shell < snippets\samples.py
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win3
2
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> >>> >>> >>> >>> >>> >>> >>> >>> >>> >>> >>> >>> >>> >>> >>> {'id': 17, 'title': '', 'code
': 'print("hello, world")\n', 'linenos': False, 'language': 'python', 'style': 'friendly'}
>>> b'{"id":17,"title":"","code":"print(\\"hello, world\\")\\n","linenos":false,"language":"p
ython","style":"friendly"}'
>>>
now exiting InteractiveConsole...

(env) C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization\tutorial>
```

📌 **Deserialize 하는 부분은 script 로 실행하지 않고 shell에서 바로 실행해 봅시다.**

Shell 을 실행하자.

```
python manage.py shell
```

Deserialize 하는 과정도 Serialize 과정과 비슷하다. Byte Stream 을 파이썬 native datatype 으로 변환한다.

```
>>> import io

#위에 Serialization 과정에서 만들었던 바이너리 데이터
#content = JSONRenderer().render(serializer.data)

>>> stream = io.BytesIO(content) #바이너리 데이터들을 IO 라이브러리로 읽어온 후
>>> data = JSONParser().parse(stream) #원래의 값들로 파싱한다
```

data 변수는 python native datatype 인데, 이를 다시 모델 객체로 변환한다.

```
>>> serializer = SnippetSerializer(data=data) #파싱된 값들을 Snippet 객체로 변환한다.
>>> serializer.is_valid()
# True

>>> serializer.validated_data
# OrderedDict([('title', ''), ('code', 'print("hello, world")\n'), ('linenos', False), ('language', 'python'), ('style', 'frien

>>> serializer.save()
# <Snippet: Snippet object>
```

모델 객체 대신 queryset 들을 직렬화 할 수 도 있다. `many=True` 플래그를 Serializer 매개변수에 지정해주자.

```
>>> serializer = SnippetSerializer(Snippet.objects.all(), many=True)

>>> serializer.data
# [OrderedDict([('id', 1), ('title', ''), ('code', 'foo = "bar"\n'), ('linenos', False), ('language', 'python'), ('style', 'fri
```

객체를 추가하고 확인했으니 잠시 Shell 을 종료하자.

```
#Shell 에다가
>>> quit()

#혹은 exit()
#혹은 Ctrl + D
```

---

6. Using ModelSerializers

위의 SnippetSerializer 는 그냥 **일반적인 Serializer 클래스를 상속받았는데, 이 경우 중복해서 필드를 작성해야 하는 불편함이 있다.** 따라서 Model 클래스를 쉽게 직렬화 할 수 있도록 하는 ModelSerializer 를 상속받는 방식으로 코드를 수정하자.

`snippets/serializers.py` 파일을 아래와 같이 수정하자.

```
class SnippetSerializer(serializers.ModelSerializer):
    class Meta:
        model = Snippet
        fields = ['id', 'title', 'code', 'linenos', 'language', 'style']
```

쉘을 다시 실행하여 수정한 Serializer 클래스를 테스트 해보자.

```
from snippets.serializers import SnippetSerializer
serializer = SnippetSerializer()
print(repr(serializer))

# SnippetSerializer():
#    id = IntegerField(label='ID', read_only=True)
#    title = CharField(allow_blank=True, max_length=100, required=False)
#    code = CharField(style={'base_template': 'textarea.html'})
#    linenos = BooleanField(required=False)
#    language = ChoiceField(choices=[('Clipper', 'FoxPro'), ('Cucumber', 'Gherkin'), ('RobotFramework', 'RobotFramework'), ('ab
#    style = ChoiceField(choices=[('autumn', 'autumn'), ('borland', 'borland'), ('bw', 'bw'), ('colorful', 'colorful')...
```

→ ModelSerializer 클래스는 자동적으로 필드들을 생성해주고, 이에 상응하는 create(), update() 메서드를 생성해준다는 편리함이 있다.

📌 하지만 필요한 필드를 자동으로 생성해주지 않는 경우에는 따로 선언하거나, 그냥 Serializer 클래스를 사용해야 함에 주의하자.

❓ I AM A PYTHON NEWBIE ❓

`Object.repr(self)` : 객체를 String 값으로 출력해주는 함수이다. str() 과 다른 점은 repr() 은 보통 디버깅시에 많이 사용되는 경우가 많다는 점이다. 사실상 기능은 같지만 repr 을 정의하는 경우 더 많은 정보들을 상세히 출력하도록 하는 것이 일반적이다. *(information rich and unambiguous)*

---

7. Writing regular Django views using our Serializer

API 뷰를 작성해보자.

`snippets/views.py` 파일을 아래와 같이 수정하자.

```
from django.http import HttpResponse, JsonResponse
from django.views.decorators.csrf import csrf_exempt
from rest_framework.parsers import JSONParser
from snippets.models import Snippet
from snippets.serializers import SnippetSerializer
```

API의 루트는 (localhost:8000/snippets 에 접속했을 때 뜨는 것) 코드 스닙펫의 리스트를 보여주는 페이지이다. 아래와 같은 메서드가 해당 요청을 수행한다.

```
@csrf_exempt
def snippet_list(request):
    """
    List all code snippets, or create a new snippet.
    """
    if request.method == 'GET':
        snippets = Snippet.objects.all()
        serializer = SnippetSerializer(snippets, many=True)
        return JsonResponse(serializer.data, safe=False)

    elif request.method == 'POST':
        data = JSONParser().parse(request)
        serializer = SnippetSerializer(data=data)
        if serializer.is_valid():
            serializer.save()
            return JsonResponse(serializer.data, status=201)
        return JsonResponse(serializer.errors, status=400)
```

CSRF 토큰이 없는 클라이언트들도 이 뷰에 POST 할 수 있도록 동작하길 원하기 때문에 `csrf_exempt` 어노테이션을 달아준다. 일반적으로는 이 작업을 할 일이 없는데, 지금으로선 우리의 목적을 달성하기엔 충분하다.

→ CSRF 토큰이 뭔가요?

API 루트에서 primary key 값을 path 에 추가하면 해당 snippet 도 반환될 수 있도록 하는 메서드도 작성하자.
(localhost:8000/snippets/1/ 과 같이 뒤에 번호를 명시해 요청하는 경우)

```
@csrf_exempt
def snippet_detail(request, pk):
    """
    Retrieve, update or delete a code snippet.
    """
    try:
        snippet = Snippet.objects.get(pk=pk)
    except Snippet.DoesNotExist:
        return HttpResponse(status=404)

    if request.method == 'GET':
        serializer = SnippetSerializer(snippet)
        return JsonResponse(serializer.data)

    elif request.method == 'PUT':
        data = JSONParser().parse(request)
        serializer = SnippetSerializer(snippet, data=data)
        if serializer.is_valid():
            serializer.save()
            return JsonResponse(serializer.data)
        return JsonResponse(serializer.errors, status=400)

    elif request.method == 'DELETE':
        snippet.delete()
        return HttpResponse(status=204)
```

views.py 파일에서 작성한 위의 메서드들을 실제 주소와 매핑하자.

`snippets/urls.py` 에 들어가서 아래와 같이 코드를 붙여넣자.

```
from django.urls import path
from snippets import views

urlpatterns = [
    path('snippets/', views.snippet_list),
    path('snippets/<int:pk>/', views.snippet_detail),
]
```

루트 URLconf 또한 설정을 해줘야 한다. `tutorial/urls.py` 파일에 들어가서 우리가 만든 snippet 앱의 URL 을 인식할 수 있도록 코드를 수정하자.

```
from django.urls import path, include

urlpatterns = [
    path('', include('snippets.urls')),
]
```

8. Testing our first attempt at a Web API

쉘에서 아직 나오지 않았다면 `quit()` 으로 빠져나오자.

```
quit()
```

... 그리고 디장고의 개발 서버를 작동시키자.

```
python manage.py runserver
```

→ localhost:8000/snippets/ 주소로 접속하면 위에서 추가한 Snippet 객체들의 정보가 JSON 으로 반환된 것을 확인할 수 있다.

터미널을 하나 더 켜서 우리의 프로젝트 폴더 밑으로 들어오자. `env\Scripts\activate.bat` 까지 했다면 아래와 같이 `httpie` 를 설치해보자.

```
pip install httpie
```

그리고 `http` 명령어를 이용해 로컬호스트에 접속해보자

```
http http://127.0.0.1:8000/snippets/
```

아래와 같이 응답이 오는 것을 확인할 수 있다.

```
C:\Users\sarah\Desktop\pg\6th-django-study\tutorial1-serialization\tutorial>http http://127.0.0.1:8000/snippets/
HTTP/1.1 200 OK
Content-Length: 352
Content-Type: application/json
Date: Sun, 12 Apr 2020 09:53:02 GMT
Server: WSGIServer/0.2 CPython/3.8.2
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

[
    {
        "code": "foo = \"bar\"\n",
        "id": 1,
        "language": "python",
        "linenos": false,
        "style": "friendly",
        "title": ""
    },
    {
        "code": "print(\"hello world\")\n",
        "id": 2,
        "language": "python",
        "linenos": false,
        "style": "friendly",
        "title": ""
    },
    {
        "code": "print(\"hello world\")",
        "id": 3,
        "language": "python",
        "linenos": false,
        "style": "friendly",
        "title": ""
    }
]
```