장고 모델폼의 강력함!

DjangoCon 2016 - Modelform Lightning Talk
https://youtu.be/aNHBr7q-KVw?t=6m35s

M2M - ManyToManyField

(다대다 관계)

다대다 관계

예) 포스트와 태그, 지하철 노선과 역

정의하기

```
class Post(models.Model):
        title = ...
        content = ...
        tag_set = models.ManyToManyField('Tag', blank=Tru

class Tag(models.Model):
        name = ...
```

db.sqlite3 구조

- 두개의 테이블을 id로 묶은 새로운 테이블이 생성됨

1. 필드 정의는 어느 쪽에?
    - ManyToManyField 필드 정의는 어느 쪽에 해도 상관없다!
    - 단, admin.py 에서 관리해줄 때 편한 쪽으로 설계하자! (Post 안에 Tag 가 들어가 있는 것이 자연스러움)
2. 형식
    - 다대다 필드명은 복수형으로 써주거나 related manager 형식(ex. tag_set)으로 쓰는게 일반적
3. 왜 'Tag' 따옴표 안에?
    - Tag 클래스가 정의되기 전에 쓰였기 때문에. 문자열 Tag 로 찾으라고 알려주는 것
    - 물론 Tag 클래스를 Post 보다 먼저 선언해도 됨. 그러나 Post가 더 중요한 클래스이므로 위와 같은 방법 추천

```
$ python manage.py shell
($ python manage.py shell_plus)
>>> from video.models import Video, Tag
>>> ...
```

** django_extension (shell_plus)

QuerySet 활용

```
>>> p = Video.objects.last()     # id 로 접근해도 상관없음
>>> p
<Post: 파이썬 A to Z>

>>> t = Tag.objects.first()      # id 로 접근해도 상관없음
>>> p.tag_set.add(t)             # .add()로 tag_set에 tag 추

>>> p.tag_set.all()
<QuerySet [<Tag: Python>]>
```

```
>>> t1 = Tag.objects.create(name='starbucks')
>>> t1
<Tag: starbucks>

>>> p.tag_set.add(t1)
>>> p.tag_set.all()
<QuerySet [<Tag: Python>, <Tag: starbucks>]>

>>> p.tag_set.create(name='sandwich')
<Tag: sandwich>
>>> p.tag_set.all()
<QuerySet [<Tag: Python>, <Tag: starbucks>, <Tag: sandwic
```

Through Field

- 두 모델간의 관계가 추가적인 데이터를 가지는 경우
  -> 다대다 관계를 가지는 두 개의 모델 이외에 추가 데이터를 저장할 또다른 모델 - 중간(intermediate) 모델 - 을 만들어 준다.
- 예) `멤버`와 `그룹` 간의 다대다 관계에서 - 멤버가 그룹에 `가입한 날짜`나 `가입 이유` 등을 지정해주어야할 때

```
from django.db import models

class Person(models.Model):
    name = models.CharField(max_length=128)

    def __str__(self):
        return self.name

class Group(models.Model):
    name = models.CharField(max_length=128)
    members = models.ManyToManyField(Person, through='Meml

    def __str__(self):
        return self.name

class Membership(models.Model):
    person = models.ForeignKey(Person)
    group = models.ForeignKey(Group)
    date_joined = models.DateField()
    invite_reason = models.CharField(max_length=64)
```

1. M2M 필드 지정할 때, through 속성을 명시한다.
2. 중간 모델을 각각의 M2M 모델과 ForeignKey로 연결한다.

```
>>> ringo = Person.objects.create(name="Ringo Starr")
>>> paul = Person.objects.create(name="Paul McCartney")
>>> beatles = Group.objects.create(name="The Beatles")

>>> m1 = Membership(person=ringo, group=beatles,
...     date_joined=date(1962, 8, 16),
...     invite_reason="Needed a new drummer.")
>>> m1.save()

>>> beatles.members.all()
[<Person: Ringo Starr>]
>>> ringo.group_set.all()
[<Group: The Beatles>]

>>> m2 = Membership.objects.create(person=paul, group=bea
...     date_joined=date(1960, 8, 1),
...     invite_reason="Wanted to form a band.")
>>> beatles.members.all()
[<Person: Ringo Starr>, <Person: Paul McCartney>]
```

1. QuerySet 에서 바로 멤버를 그룹에 add해주지 못하고 중간 모델을 정의해주어야 한다.
2. `__gt`

```
>>> Person.objects.filter(
...     group__name='The Beatles',
...     membership__date_joined__gt=date(1961,1,1))
[<Person: Ringo Starr]
```

3. `__icontains`

```
>>> Person.objects.filter(group__name__icontains='Beatles
>>> print(q)
<QuerySet [<Person: Ringo Starr>]>
>>> q.delete()
```

```
from django.db import models

class Person(models.Model):
    name = models.CharField(max_length=50)

class Group(models.Model):
    name = models.CharField(max_length=128)
    members = models.ManyToManyField(
        Person,
        through='Membership',
        through_fields=('group', 'person'),
    )

class Membership(models.Model):
    group = models.ForeignKey(Group, on_delete=models.CAS
    person = models.ForeignKey(Person, on_delete=models.C
    inviter = models.ForeignKey(
        Person,
        on_delete=models.CASCADE,
        related_name="membership_invites",
    )
    invite_reason = models.CharField(max_length=64)
```

1. through_fields
    - 한개의 모델에서 여러개 필드(Person-person, inviter)를 가져와 쓸 때, 다대다 관계는 Person 과 Group임을 명시