Django - Query set 쿼리 셋

Select 쿼리

- **Class.objects.all()**: 테이블의 모든 데이터 조회 -> <QuerySet [ ]>
- **Class.objects.get()**: 하나의 행(row)만 조회 -> <Object: ‘result’>
- **Class.objects.filter()**: 해당 조건에 맞는 데이터만 조회 -> <QuerySet [ ]>
- **Class.objects.exclude()**: 해당 조건을 제외한 데이터만 조회 -> <QuerySet [ ]>
- **Class.objects.count()**: 해당 조건을 가진 데이터 개수 -> int
- **Class.objects.exists()**: 조건에 해당하는 데이터 유무 -> True, False

```
Post.objects.all()
Post.objects.get(pk=pk)
Comment.objects.filter(post=post)
Comment.objects.exclude(post=post)
HashTag.objects.count()
HashTag.objects.filter(post=post).exists()
```

Django - Query set 쿼리 셋

F(): ORM을 사용할 때, 파이썬 메모리 효율을 위해 사용하는 메서드
데이터베이스에서 먼저 쿼리 동작 후 반환

Select 쿼리

- **Class.objects.values()**: 테이블의 값을 리스트 속 딕셔너리로 반환 -> <QuerySet [{'key':'value'}]>
- **Class.objects.values_list()**: 테이블의 값을 리스트 속 튜플로 반환 -> <QuerySet [(), ()]>
- **Class.objects.order_by()**: 특정 필드를 기준으로 정렬(-필드명: 내림차순) -> <QuerySet [ ]>
- **Class.objects.first(), last()**: 쿼리셋 결과 중 가장 첫 번째, 마지막 결과 -> <Object: 'result'>
- **Class.objects.aggregate()**: 집계 함수 적용 시 사용 (GROUP BY) -> <QuerySet [ ]>
- **Class.objects.annotate()**: 컬럼 별로 주석을 달아 집계 함수 적용시 사용 (as) -> <QuerySet [ ]>

Post.objects.values() -> <QuerySet [{'id': 1, 'title': 'title1', … }, …]>
Post.objects.values_list() -> <QuerySet [(1, 'title1', …), …]>
HashTag.objects.order_by('-name')
Comment.objects.first()
User.objects.aggregate(Avg('age'))
Post.objects.annotate(title=F("post__comment").values('title))

Django - Query set 쿼리 셋

Select 쿼리

- **filter(), exclude() 필터**
  - __startswith: 특정 문자로 시작
  - __endswith: 특정 문자로 끝남
  - __contains: 특정 문자를 포함 (대소문자 구분)
  - __icontains: 특정 문자를 포함 (대소문자 구분 x)
  - __gt: 특정 값보다 큼
  - __lt: 특정 값보다 작음
  - __in: 특정 문자열을 포함하고 있는 것
  - __isnull: True일 시 필드값이 null 인 것
  - __year: 특정 년도
  - __month: 특정 월
  - __day: 특정 일
  - __date: 특정 날짜(YY-MM-DD)

```
User.objects.filter(date_joined__year=”2023”)
```

Django - Query set 쿼리 셋

Select 쿼리

- **Chaining**

  Post.objects.filter(writer=user).count()

- **Slicing**

  Post.objects.all()[:3]

Django - Query set 쿼리 셋

Select 쿼리

- **filter()**
  - 논리 연산자
    - **AND**

      Comment.objects.filter(post=post) & Comment.objects.filter(pk=pk)
    - **OR**

      Comment.objects.filter(post=post) | Comment.objects.filter(pk=pk)
    - **NOT**

      !(Comment.objects.filter(post=post))
  - **Q()** 조건문
    - **AND**

      Comment.objects.filter(Q(post=post) & Q(pk=pk))
    - **OR**

Django - Query set 쿼리 셋

Insert 쿼리

- **Class.objects.create():** 하나의 객체 생성

  Post.objects.create(title='title1', writer='writer1', content='content1')
- **Class.objects.bulk_create([ ]):** 여러 개의 객체 생성

  Post.objects.bulk_create([Post(title='title1',writer='writer1',content='content1'), Post(...)])
- **Class.objects.get_or_create():** 조건에 맞는 데이터가 이미 있으면 **get** 없으면 **create**

  Post.objects.get_or_create(title='title1', writer='writer1', content='content1')

Django - Query set 쿼리 셋

Update 쿼리

- 업데이트할 객체를 변수에 저장해서 각 필드에 접근

```
post = Post.object.get(pk=pk)
post.title = ‘new title’
post.save()
```

- 업데이트할 객체에 직접 접근

```
Post.objects.get(pk=pk).update(content=’new content’)
```

Django - Query set 쿼리 셋

Delete 쿼리

- 업데이트할 객체를 변수에 저장해서 삭제

```
post = Post.objects.get(pk=pk)
post.delete()
```

Delete 실무 구현

- User: removed_user(flag) -> boolean, 0 or 1

Django - Query set 쿼리 셋

Django ORM: SQL을 파이썬 문법으로 사용 가능하게 도와주는 기능

Django QuerySet
- **get, all ..**
- **create, bulk_creat ..**
- **.update, .title='title'**
- **.delete()**

SQL
- **SELECT**
- **INSERT**
- **UPDATE**
- **DELETE**

-> Pagination

Django - Query set 쿼리 셋

쿼리 고도화 (JOIN)

단일 쿼리 -> 관계가 있는 데이터를 가지고 온다.

- select_related(JOIN): SQL 데이터베이스 상에서
- prefetch_related(JOIN): 파이썬 상에서

(정)참조, 역참조 -> 두 가지 다 표현 가능

Django REST FRAMEWORK

Server Side Rendering

REST: API

DRM

- 설치: python -m pip install djangorestframework
- settings.py 안에 INSTALL_APPS