ТЕХНОАТОМ

# Продолжение работы с ORM

Лекция 6

Алена Елизарова

# Еще немного про миграции

ТЕХНОАТОМ

```
# Data Migration
# python manage.py makemigrations --empty yourappname

from django.db import migrations

def do_smth():
    ...

class Migration(migrations.Migration):

    dependencies = [
        ('yourappname', '0001_initial'),
    ]

    operations = [
        migrations.RunPython(do_smth),
    ]
```

# Еще немного про миграции

ТЕХНОАТОМ

```
def do_smth(apps, shema_editor):
    SomeModel = apps.get_model('some_app', 'SomeModel')
```

# Django Admin

ТЕХНОАТОМ

```
# chats.admin.py

from django.contrib import admin
from chats.models import Chat

class ChatAdmin(admin.ModelAdmin):
    list_display = ('id', 'title')
    list_filter = ('is_active',)

admin.site.register(Chat, ChatAdmin)



# дока https://docs.djangoproject.com/en/2.2/ref/contrib/admin/
```

# Методы модели

ТЕХНОАТОМ

```
# chats.models.py

from django.db import models

class Chat(models.Model):
    title = models.CharField('Название', max_length=50)
    sorting_key = models.IntegerField('Ключ сортировки', default=0)

    def get_absolute_url(self):
        reverse('chat', kwargs={'chat_id': self.id})

    def __str__(self):
        return self.title

    class Meta:
        ordering = ['-sorting_key']
        verbose_name = 'Чат'
        verbose_name_plural = 'Чаты'
```

# Расширение AbstractUser

ТЕХНОАТОМ

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


class User(AbstractUser):
    bio = models.TextField('Биография', max_length=500, blank=True)
    location = models.CharField('Город', max_length=30, blank=True)
    birthday = models.DateField('Дата рождения', null=True, blank=True)


# в settings.py добавить AUTH_USER_MODEL = "users.YourUserModel"
```

# Шаблонизация в Django

ТЕХНОАТОМ

на самостоятельное изучение

https://docs.djangoproject.com/en/2.2/topics/templates/

# Продолжаем ORM

ТЕХНОАТОМ

```
# создание объекта без связей

post = Post()
post.title = ‘Test’
post.save()

post2 = Post.objects.create(title=’Test2’)


# создание объекта со связями

post3 = Post.objects.create(title=’Test3’)
post3.category_id = 2
post3.save()

category = Category.objects.create(title=’Test category’)
post4 = Post.objects.create(title=’Test 4’, category=category)
```

# Связи ManyToMany

ТЕХНОАТОМ

```
post = Post.objects.create(title='Test2')
tag = Tag.objects.create(slug='some_tag')
post.tags.add(tag)
```

# Загрузка объектов из БД

ТЕХНОАТОМ

```
# по ключу

try:
    post = Post.objects.get(id=5)
except Post.DoesNotExist:
    post = None

# по другому полю

try:
    post = Post.objects.get(title='Python')
except MultipleObjectsReturned:
    post = None
```

# Выборка нескольких объектов из БД

ТЕХНОАТОМ

```
all_posts = Post.objects.all()
first_three = Post.objects.all()[:3]

some_category = Categoty.objects.get(id=1)

category_posts = Post.objects.filter(category=some_category)
category_posts = Post.objects.filter(category_id=1)


css_posts = Post.objects.filter(title__contains='css')
css_posts = css_posts.order_by('-rating')
css_posts = css_posts[10:20]
```

**QuerySet** - объекты, представляющие собой запрос к базе данных (не результаты)

**QuerySet** - ленивые объекты

# Chaining

TEXHOATOM

```
posts = Post.objects.all()

posts = posts.filter(title__contains='CSS')

posts = posts.exclude(id=21)

posts = posts.order_by('-rating')

posts = posts.reverse() # [ QuerySet ]

posts = posts[:3] # [ POST ]
```

# Методы QuerySet (chaining)

ТЕХНОАТОМ

*filter, exclude* - фильтрация, WHERE в SQL

*order_by* - сортировка

*annotate* - выбор агрегатов, в SQL - JOIN и GROUP_BY

*values* - выбор отдельных колонок, а не объектов

*values_list* - то же, только без названия колонок

*distinct* - выбор уникальных значений

*select_related*, *prefetch_related* - выборка из нескольких таблиц

# Методы QuerySet (результат)

ТЕХНОАТОМ

*create* - создание нового объекта

*update* - обновление всех подходящих объектов

*delete* - удаление одного или нескольких объектов

*get_or_create* - выборка объекта или его создание

*count* - выборка количества COUNT(*)

# Синтаксис условий в filter и exclude

ТЕХНОАТОМ

```
filed=valie - точное совпадние

field__contains=value - суффикс оператора LIKE

filf__isnull, field__gt, field__lte

relation__field=value - условие по связанной таблице

category__title__contains='Python'


# Названия полей и таблиц не могут содержать __
```

# ModelManager

ТЕХНОАТОМ

В модели содержатся методы для работы с одним объектом (одной строкой).

В **ModelManager** содержатся объекты для работы с множеством объектов.

**ModelManager** по умолчанию содержит все те же методы, что и QuerySet и используется для создания QuerySet объектов, связанных с моделью.

# Свой ModelManager

ТЕХНОАТОМ

```
class PostManager(models.Manager):
    def best_posts(self):
       return self.filter(rating__gte=50)

    def published(self):
        return self.filter(status=Post.IS_PUBLISHED)


class Post(models.Model):
    title = ...

    ...

    objects = PostManager()

)
```

# Методы RelatedManager

ТЕХНОАТОМ

```
create(**kwargs) - создание новой категории, связанной с постом

add(category) - привязка существующей категории к посту

remove(category) - отвязка

clear() - очистка списка категорий у текущего поста
```

# Что еще?

ТЕХНОАТОМ

```
Post.objects.get()

Post.objects.first() # обычно после filter

Post.objects.last() # обычно после filter

Post.objects.none()

Post.objects.filter(id=2).exists()
```

# Усложняем запросы

ТЕХНОАТОМ

```
from django.db.models import Count
from django.db.models import Max
from django.db.models.functions import Length

posts = Post.objects.annotate(Count('tags')) # tags__count
posts = Post.objects.all().annotate(tag_count=Count('tags'))

posts = posts.filter(tag_count__gte=3)

posts = Post.objects.all().annotate(length=Length('title'))

users = User.objects.all().aggregate(age_max=Max(age))
```

# Почитать


https://docs.djangoproject.com/en/2.2/misc/design-philosophies/

https://docs.djangoproject.com/en/2.2/topics/db/models/

https://docs.djangoproject.com/en/2.2/ref/models/querysets/

# Квиз №4

ТЕХНОАТОМ

Маршрутизация URL в терминах Django это?

1) Процесс определения лучшего пути http-запроса
2) Проксирование запроса на Application Server
3) Нахождение подходящего контроллера по пути
4) Нахождение подходящего шаблона html по пути

# Квиз №4

ТЕХНОАТОМ

В Django согласно паттерну MVC роль компонента  View выполняет

1) urls.py

2) views.py

3) /templates

4) /static

# Квиз №4

ТЕХНОАТОМ

В Django приложение - это

1) Сам проект

2) Логическая часть проекта

3) wsgi-приложение

# Квиз №4

ТЕХНОАТОМ

Какая настройка может отключить traceback ошибки в браузере

1) ASYNC

2) TESTING

3) ALLOWED_HOSTS

4) DEBUG

# Квиз №4

ТЕХНОАТОМ

Какая БД идет "в комплекте" с Django по умолчанию

1) sqlite2

2) sqlite3

3) postgresql

4) Никакая

# Квиз №4

ТЕХНОАТОМ

Зачем нужен файл local_settings.py

1) Для переопределения любых переменных окружения внутри wsgi-приложения

2) Для того, чтобы не возникали конфликты в системе контроля версий

3) Для переопределения конкретных значений в настройках проекта

4) Для переопределения переменной DEBUG

# Квиз №4

ТЕХНОАТОМ

Django views первым параметром принимают

1) Ничего

2) Не важно что, главное чтобы одним из позиционных аргументов был объект запроса

3) Объект запроса

4) GET-параметры

# Квиз №4

ТЕХНОАТОМ

Что содержится в request.GET?

1) Мета-информация о запросе

2) Параметры из querystring

3) Параметры, прокинутые из url, по которому было найдено совпадение

4) Параметры из тела запроса

# Квиз №5

ТЕХНОАТОМ

https://forms.gle/vTWv3UuzHx2ouqPo9

# Домашнее задание

Реализовать методы для:
поиска пользователей
создания персонального чата
получения списка чатов

(пишем код во вьюхах, можем пока возвращать пустой http-response)

ТЕХНОАТОМ

# Спасибо за внимание!